NIST Special Publication 500–215

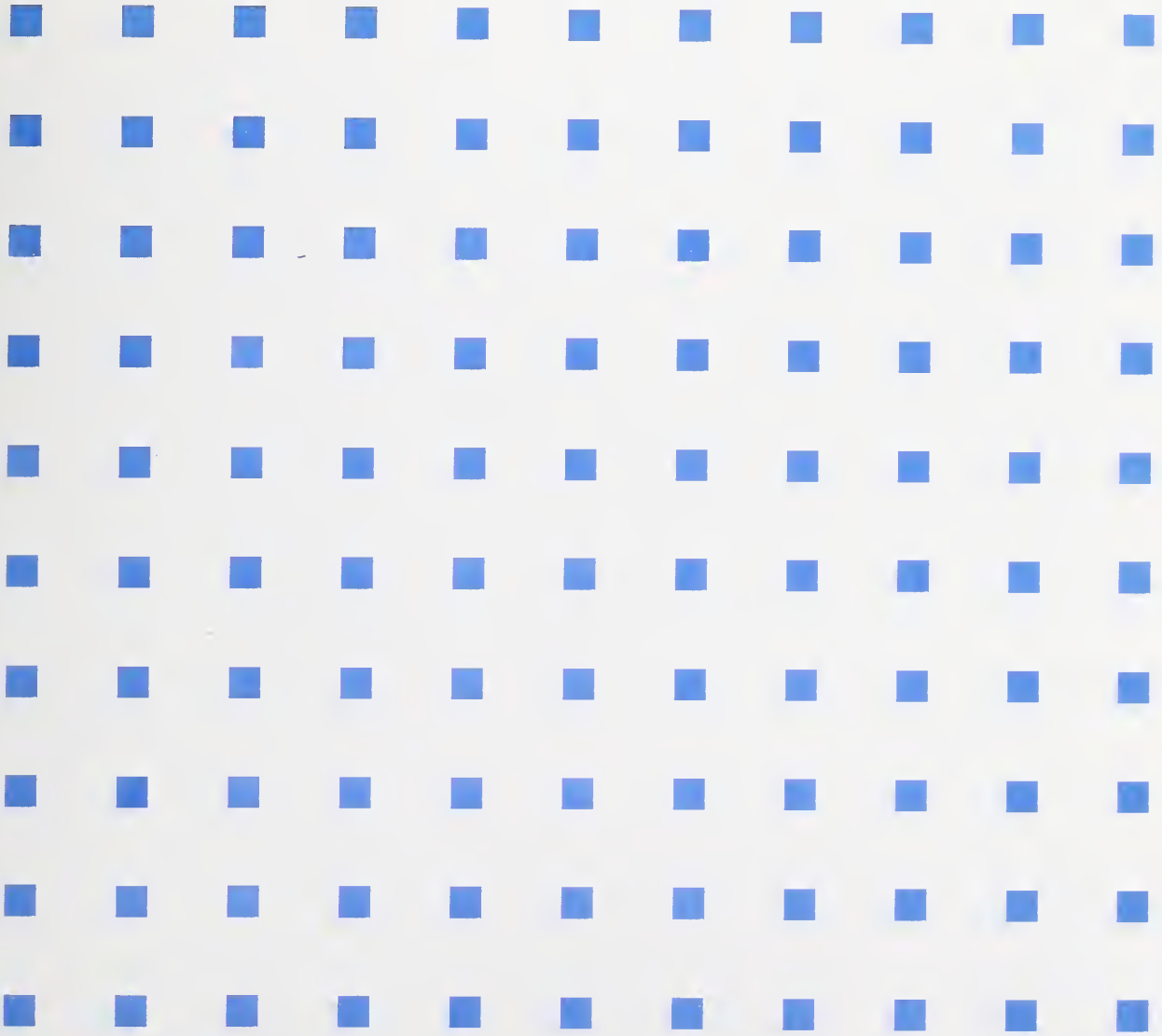
Computer Systems Technology

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards and Technology

NIST

The Second Text REtrieval Conference (TREC–2)

D. K. Harman, Editor



The National Institute of Standards and Technology was established in 1988 by Congress to "assist industry in the development of technology . . . needed to improve product quality, to modernize manufacturing processes, to ensure product reliability . . . and to facilitate rapid commercialization . . . of products based on new scientific discoveries."

NIST, originally founded as the National Bureau of Standards in 1901, works to strengthen U.S. industry's competitiveness; advance science and engineering; and improve public health, safety, and the environment. One of the agency's basic functions is to develop, maintain, and retain custody of the national standards of measurement, and provide the means and methods for comparing standards used in science, engineering, manufacturing, commerce, industry, and education with the standards adopted or recognized by the Federal Government.

As an agency of the U.S. Commerce Department's Technology Administration, NIST conducts basic and applied research in the physical sciences and engineering and performs related services. The Institute does generic and precompetitive work on new and advanced technologies. NIST's research facilities are located at Gaithersburg, MD 20899, and at Boulder, CO 80303. Major technical operating units and their principal activities are listed below. For more information contact the Public Inquiries Desk, 301-975-3058.

Technology Services
- Manufacturing Technology Centers Program
- Standards Services
- Technology Commercialization
- Measurement Services
- Technology Evaluation and Assessment
- Information Services

Electronics and Electrical Engineering Laboratory
- Microelectronics
- Law Enforcement Standards
- Electricity
- Semiconductor Electronics
- Electromagnetic Fields[1]
- Electromagnetic Technology[1]

Chemical Science and Technology Laboratory
- Biotechnology
- Chemical Engineering[1]
- Chemical Kinetics and Thermodynamics
- Inorganic Analytical Research
- Organic Analytical Research
- Process Measurements
- Surface and Microanalysis Science
- Thermophysics[2]

Physics Laboratory
- Electron and Optical Physics
- Atomic Physics
- Molecular Physics
- Radiometric Physics
- Quantum Metrology
- Ionizing Radiation
- Time and Frequency[1]
- Quantum Physics[1]

Manufacturing Engineering Laboratory
- Precision Engineering
- Automated Production Technology
- Robot Systems
- Factory Automation
- Fabrication Technology

Materials Science and Engineering Laboratory
- Intelligent Processing of Materials
- Ceramics
- Materials Reliability[1]
- Polymers
- Metallurgy
- Reactor Radiation

Building and Fire Research Laboratory
- Structures
- Building Materials
- Building Environment
- Fire Science and Engineering
- Fire Measurement and Research

Computer Systems Laboratory
- Information Systems Engineering
- Systems and Software Technology
- Computer Security
- Systems and Network Architecture
- Advanced Systems

Computing and Applied Mathematics Laboratory
- Applied and Computational Mathematics[2]
- Statistical Engineering[2]
- Scientific Computing Environments[2]
- Computer Services[2]
- Computer Systems and Communications[2]
- Information Systems

[1] At Boulder, CO 80303.
[2] Some elements at Boulder, CO 80303.

NIST Special Publication 500–215

The Second Text REtrieval Conference (TREC–2)

D. K. Harman, Editor

Computer Systems Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899

March 1994

U.S. Department of Commerce
Ronald H. Brown, Secretary

Technology Administration
Mary L. Good, Under Secretary for Technology

National Institute of Standards and Technology
Arati Prabhakar, Director

Reports on Computer Systems Technology

The National Institute of Standards and Technology (NIST) has a unique responsibility for computer systems technology within the Federal government. NIST's Computer Systems Laboratory (CSL) develops standards and guidelines, provides technical assistance, and conducts research for computers and related telecommunications systems to achieve more effective utilization of Federal information technology resources. CSL's responsibilities include development of technical, management, physical, and administrative standards and guidelines for the cost-effective security and privacy of sensitive unclassified information processed in Federal computers. CSL assists agencies in developing security plans and in improving computer security awareness training. This Special Publication 500 series reports CSL research and guidelines to Federal agencies as well as to organizations in industry, government, and academia.

National Institute of Standards and Technology Special Publication 500-215
Natl. Inst. Stand. Technol. Spec. Publ. 500-215, 479 pages (March 1994)
CODEN: NSPUE2

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1994

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402

Preface

This report constitutes the proceedings of the second Text REtrieval Conference (TREC-2) held in Gaithersburg, Maryland, August 31-September 2, 1993. The conference was co-sponsored by the National Institute of Standards and Technology (NIST) and the Advanced Research Projects Agency (ARPA), and was attended by 150 people involved in the 31 participating groups. The conference was the second in an on-going series of workshops to evaluate new technologies in text retrieval.

The workshop included plenary sessions and twelve discussion groups. Because the participants in the workshop drew on their personal experiences, they sometimes cited specific vendors and commercial products. The inclusion or omission of a particular company or product does not imply either endorsement or criticism by NIST.

The sponsorship of the Software and Intelligent Systems Technology Office of the Advanced Research Projects Agency is gratefully acknowledged, along with the tremendous work of the program committee.

Donna Harman
February 20, 1994

TREC-2 Program Committee

Donna Harman, NIST, chair
Chris Buckley, Cornell University
Susan Dumais, Bellcore
Darryl Howard, U.S. Department of Defense
David Lewis, AT & T Bell Labs
John Prange, U.S. Department of Defense
Alan Smeaton, Dublin City University, Ireland
Richard Tong, Advanced Decision Systems
Steve Walker, City University, UK

TABLE OF CONTENTS

REPORTS OF DISCUSSION GROUPS

APPENDICES

Abstract

This report constitutes the proceedings of the second Text REtrieval Conference (TREC-2) held in Gaithersburg, Maryland, August 31-September 2, 1993. The conference was co-sponsored by the National Institute of Standards and Technology (NIST) and the Advanced Research Projects Agency (ARPA), and was attended by 150 people involved in the 31 participating groups.

The goal of the conference was to bring research groups together to discuss their work on a new large test collection. There was a wide variation of retrieval techniques reported on, including methods using automatic thesaurii, sophisticated term weighting, natural language techniques, relevance feedback, and advanced pattern matching. As results had been run through a common evaluation package, groups were able to compare the effectiveness of different techniques, and discuss how differences between the systems affected performance.

The conference included paper sessions and discussion groups. This proceedings includes papers from most of the participants (several poster groups did not submit papers), along with reports from some of the discussion groups.

Overview of the Second Text REtrieval Conference (TREC-2)

Donna Harman

National Institute of Standards and Technology
Gaithersburg, MD. 20899

1. Introduction

In November of 1992 the first Text REtrieval Conference (TREC-1) was held at NIST [Harman 1993]. The conference, co-sponsored by ARPA and NIST, brought together information retrieval researchers to discuss their system results on a new large test collection (the TIPSTER collection). This was the first time that such groups had ever compared results on the same data using the same evaluation methods and represented a breakthrough in cross-system evaluation in information retrieval. It was also the first time that most of these groups had used such a large test collection and therefore required a major effort by all groups to scale up their retrieval techniques.

The overall goal of the TREC initiative is to encourage research in information retrieval using large-scale test collections. It is hoped that by providing a very large test collection, and encouraging interaction with other groups in a friendly evaluation forum, new momentum in information retrieval will be generated. Because of the NIST involvement, groups with commercial retrieval products have participated in TREC, leading to increased technological transfer between the research labs and the commercial products. TREC has also provided a state-of-the-art showcase of retrieval methods for ARPA clients.

Whereas the TREC-1 conference demonstrated a wide range of different approaches to the retrieval of text from large document collections, the results should be viewed as very preliminary. Not only were the deadlines for results very tight, but the huge increase in the size of the document collection required significant system rebuilding by most groups. Much of this work was a system engineering task: finding reasonable data structures to use, getting indexing routines to be efficient enough to index all the data, finding enough storage to handle the large inverted files and other structures, etc. Still, the results showed that the systems did the task well, and that automatic construction of queries from the topics did as well as, or better than, manual construction of queries.

The second TREC conference (TREC-2) occurred in August of 1993, less than 10 months after the first conference. In addition to 22 of the TREC-1 groups, nine new groups took part, bringing the total number of participating groups to 31. Many of the original TREC-1 groups were able to "complete" their system rebuilding and tuning, and in general the TREC-2 results show significant improvements over the TREC-1 results.

This paper provides an overview of the TREC-2 conference, including a review of the TREC task, a brief description of the test collection being used, and an overview of the results. The papers from the individual groups should be referred to for more details on specific system approaches.

2. The TREC Task

2.1 Introduction

TREC is designed to encourage research in information retrieval using large data collections. Two types of retrieval are being examined -- retrieval using an "adhoc" query such as a researcher might use in a library environment, and retrieval using a "routing" query such as a profile to filter some incoming document stream. The TREC task is not tied to any given application, and is not primarily concerned with interfaces or optimized response time for searching. However it is helpful to have some potential user in mind when designing or testing a retrieval system. The model for a user in TREC is a dedicated searcher, not a novice searcher, and the model for the application is one needing monitoring of data streams for information on specific topics (routing), and the ability to do adhoc searches on archived data for new topics. It should be assumed that the users need the ability to do both high precision and high recall searches, and are willing to look at many documents and repeatedly modify queries in order to get high recall. Obviously they would like a system that makes this as easy as possible, but this ease should be reflected in TREC as added intelligence in the system rather than as special interfaces.

Since TREC has been designed to evaluate system performance both in a routing (filtering or profiling) mode, and in an adhoc mode, both functions need to be tested.

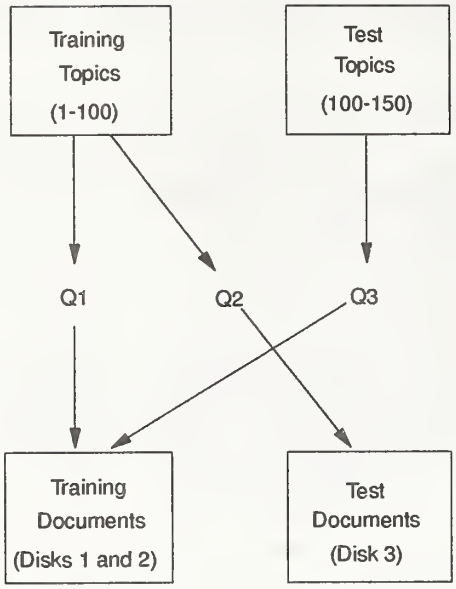


Figure 1. The TREC Task.

The test design was based on traditional information retrieval models, and evaluation used traditional recall and precision measures. The above diagram of the test design shows the various components of TREC (fig. 1).

This diagram reflects the four data sets (2 sets of topics and 2 sets of documents) that were provided to participants. These data sets (along with a set of sample relevance judgments for the 100 training topics) were used to construct three sets of queries. Q1 is the set of queries (probably multiple sets) created to help in adjusting a system to this task, to create better weighting algorithms, and in general to train the system for testing. The results of this research were used to create Q2, the routing queries to be used against the test documents. Q3 is the set of queries created from the test topics as adhoc queries for searching against the training documents. The results from searches using Q2 and Q3 were the official test results sent to NIST.

2.1 Specific Task Guidelines

Because the TREC participants used a wide variety of indexing/knowledge base building techniques, and a wide variety of approaches to generate search queries, it was important to establish clear guidelines for the evaluation task. The guidelines deal with the methods of indexing/knowledge base construction, and with the methods of generating the queries from the supplied topics. In general, they were constructed to reflect an actual operational environment, and to allow as fair as possible a separation among the diverse query construction approaches.

There were guidelines for constructing and manipulating the system data structures. These structures were defined to consist of the original documents, any new structures built automatically from the documents (such as inverted files, thesauri, conceptual networks, etc.), and any new structures built manually from the documents (such as thesauri, synonym lists, knowledge bases, rules, etc.). The following guidelines were developed for the TREC task.

1. System data structures should be built using the initial training set (documents from disks 1 and 2, training topics 1-100, and the relevance judgments). They may be modified based on the test documents from disk 3, but not based on the test topics.

2. There are parts of the test collection, such as the Wall Street Journal and the Ziff material, that contain manually assigned controlled or uncontrolled index terms. These fields are delimited by SGML tags, as specified in the documentation files included with the data. Since the primary focus is on retrieval and routing of naturally occurring text, these manually indexed terms should not be used.

3. Special care should be used in handling the routing task. In a true routing situation, a single document would be indexed and compared against the routing topics. Since the test documents are generally indexed as a complete set, routing should be simulated by not using any information based on the full set of test documents (such as weighting based on the test collection, total frequency based on the test collection, etc.) in the searching. It is permissible to use training-set collection information however.

Additionally there were guidelines for constructing the queries from the provided topics. These guidelines were considered of great importance for fair system comparison and were therefore carefully constructed. Three generic categories were defined, based on the amount and kind of manual intervention used.

1. AUTOMATIC (completely automatic initial query construction)

   adhoc queries -- The system will automatically extract information from the topic to construct the query. The query will then be submitted to the system (with no manual modifications) and the results from the system will be the results submitted to NIST. There should be no manual intervention that would affect the results.

   routing queries -- The queries should be

constructed automatically using the training topics, the training relevance judgments and the training documents. The queries should then be submitted to NIST before the test documents are released and should not be modified after that point. The unmodified queries should be run against the test documents and the results submitted to NIST.

2. MANUAL (manual initial query construction)

   adhoc queries -- The query is constructed in some manner from the topic, either manually or using machine assistance. Once the query has been constructed, it will be submitted to the system (with no manual intervention), and the results from the system will be the results submitted to NIST. There should be no manual intervention after initial query construction that would affect the results. (Manual intervention is covered by the category labelled FEEDBACK.)

   routing queries -- The queries should be constructed in the same manner as the adhoc queries for MANUAL, but using the training topics, relevance judgments, and training documents. They should then be submitted to NIST before the test documents are released and should not be modified after that point. The unmodified queries should be run against the test documents and the results submitted to NIST.

3. FEEDBACK (automatic or manual query construction with feedback)

   adhoc queries -- The initial query can be constructed using either AUTOMATIC or MANUAL methods. The query is submitted to the system, and a subset of the retrieved documents is used for manual feedback, i.e., a human makes judgments about the relevance of the documents in this subset. These judgments may be communicated to the system, which may automatically modify the query, or the human may simply choose to modify the query himself. At some point, feedback should end, and the query should be accepted as final. Systems that submit runs using this method must submit several different sets of results to allow tracking of the time/cost benefit of doing relevance feedback.

   routing queries -- FEEDBACK cannot be used for routing queries as routing systems have not supported feedback.

2.2 The Participants

There were 31 participating systems in TREC-2, using a wide range of retrieval techniques. The participants were able to choose from three levels of participation: Category A, full participation, Category B, full participation using a reduced dataset (1/4 of the full document set), and Category C for evaluation only (to allow commercial systems to protect proprietary algorithms). The program committee selected only 20 category A and B groups to present talks because of limited conference time, and requested that the rest of the groups present posters. All groups were asked to submit papers for the proceedings.

Each group was provided the data and asked to turn in either one or two sets of results for each topic. When two sets of results were sent, they could be made using different methods of creating queries (AUTOMATIC, MANUAL, or FEEDBACK), or by using different parameter settings for one query creation method. Groups could choose to do the routing task, the adhoc task, or both, and were requested to submit the top 1000 documents retrieved for each topic for evaluation.

3. The Test Collection

3.1 Introduction

The creation of the test collection (called the TIPSTER collection) was critical to the success of TREC. Like most traditional retrieval collections, there are three distinct parts to this collection -- the documents, the queries or topics, and the relevance judgments or "right answers." These test collection components are discussed briefly in the rest of this section. For a more complete description of the collection, see [Harman 1994].

3.2 The Documents

The documents needed to mirror the different types of documents used in the theoretical TREC application. Specifically they had to have a varied length, a varied writing style, a varied level of editing and a varied vocabulary. As a final requirement, the documents had to cover different timeframes to show the effects of document date on the routing task.

The documents were distributed as CD-ROMs with about 1 gigabyte of data each, compressed to fit. The following shows the actual contents of each disk.

Disk 1

- WSJ -- *Wall Street Journal* (1987, 1988, 1989)
- AP -- *AP Newswire* (1989)

- ZIFF -- Articles from *Computer Select* disks (Ziff-Davis Publishing)
- FR -- *Federal Register* (1989)
- DOE -- Short abstracts from DOE publications

Disk 2

- WSJ -- *Wall Street Journal* (1990, 1991, 1992)
- AP -- *AP Newswire* (1988)
- ZIFF -- Articles from *Computer Select* disks (Ziff-Davis Publishing)
- FR -- *Federal Register* (1988)

Disk 3

- SJMN -- *San Jose Mercury News* (1991)
- AP -- *AP Newswire* (1990)
- ZIFF -- Articles from *Computer Select* disks (Ziff-Davis Publishing)
- PAT -- U.S. Patents (1993)

The documents are uniformly formatted into an SGML-like structure, as can be seen in the following example.

<DOC>
<DOCNO> WSJ880406-0090 </DOCNO>
<HL> AT&T Unveils Services to Upgrade Phone Networks Under Global Plan </HL>
<AUTHOR> Janet Guyon (WSJ Staff) </AUTHOR>
<DATELINE> NEW YORK </DATELINE>
<TEXT>
American Telephone & Telegraph Co. introduced the first of a new generation of phone services with broad implications for computer and communications equipment markets.
AT&T said it is the first national long-distance carrier to announce prices for specific services under a world-wide standardization plan to upgrade phone networks. By announcing commercial services under the plan, which the industry calls the Integrated Services Digital Network, AT&T will influence evolving communications standards to its advantage, consultants said, just as International Business Machines Corp. has created de facto computer standards favoring its products.
.
.
.
</TEXT>
</DOC>

All documents have beginning and end markers, and a unique DOCNO id field. Additionally other fields taken from the initial data appear, but these vary widely across the different sources. The documents have differing amounts of errors, which were not checked or corrected. Not only would this have been an impossible task, but the errors in the data provide a better simulation of the TREC task. Errors in missing document separators or bad document numbers were screened out, although a few were missed and later reported as errors.

Table 1 shows some basic document collection statistics. Note that although the collection sizes are roughly equivalent in megabytes, there is a range of document lengths from very short documents (DOE) to very long (FR). Also the range of document lengths within a collection varies. For example, the documents from AP are similar in length (the median and the average length are very close), but the WSJ and ZIFF documents have a wider range of lengths. The documents from the Federal Register (FR) have a very wide range of lengths.

3.3 The Topics

In designing the TREC task, there was a conscious decision made to provide "user need" statements rather than more traditional queries. Two major issues were involved in this decision. First there was a desire to allow a wide range of query construction methods by keeping the topic (the need statement) distinct from the query (the actual text submitted to the system). The second issue was the ability to increase the amount of information available about each topic, in particular to include with each topic a clear statement of what criteria make a document relevant.

The topics were designed to mimic a real user's need, and were written by people who are actual users of a retrieval system. Although the subject domain of the topics was diverse, some consideration was given to the documents to be searched. The topics were constructed by doing trial retrievals against a sample of the document set, and then those topics that had roughly 25 to 100 hits in that sample were used. This created a range of broader and narrower topics.

The following is one of the topics used in TREC.

<top>
<head> Tipster Topic Description
<num> Number: 066
<dom> Domain: Science and Technology
<title> Topic: Natural Language Processing

<desc> Description:
Document will identify a type of natural language processing technology which is being developed or marketed in the U.S.

Table 1. Document Statistics

Subset of collection	WSJ (disks 1 and 2) SJMN (disk 3)	AP	ZIFF	FR (disks 1 and 2) PAT (disk 3)	DOE
Size of collection (megabytes)					
(disk 1)	295	266	251	258	190
(disk 2)	255	248	188	211	
(disk 3)	315	248	358	251	
Number of records					
(disk 1)	98,736	84,930	75,180	26,207	226,087
(disk 2)	74,520	79,923	56,920	20,108	
(disk 3)	90,257	78,325	161,021	6,711	
Median number of terms per record					
(disk 1)	182	353	181	313	82
(disk 2)	218	346	167	315	
(disk 3)	279	358	119	2896	
Average number of terms per record					
(disk 1)	329	375	412	1017	89
(disk 2)	377	370	394	1073	
(disk 3)	337	379	263	3543	

<smry> Summary:
Document will identify a type of natural language processing technology which is being developed or marketed in the U.S.

<narr> Narrative:
A relevant document will identify a company or institution developing or marketing a natural language processing technology, identify the technology, and identify one or more features of the company's product.

<con> Concept(s):
1. natural language processing
2. translation, language, dictionary, font
3. software applications

<fac> Factor(s):
<nat> Nationality: U.S.
</fac>
<def> Definition(s):
</top>

Each topic is formatted in the same standard method to allow easier automatic construction of queries. Besides a beginning and an end marker, each topic has a number, a short title, a one-sentence description, and a summary sentence or two that can be used as a surrogate for the full topic (often very similar to the one-sentence description). There is a narrative section which is aimed at providing a complete description of document relevance for the assessors. Each topic also has a concepts section with a list of assorted concepts related to the topic. This section is designed to provide a mini-knowledge base about a topic such as a real searcher might possess. Additionally each topic can have a definitions section and/or a factors section. The definition section has one or two of the definitions critical to a human understanding of the topic. The factors section is included to allow easier automatic query building by listing specific items from the narrative that constrain the documents that are relevant. Two particular factors were used in the TREC-2 topics: a time factor (current, before a given date, etc.) and a nationality factor (either involving only certain countries or excluding certain countries).

While the TREC topics did not present a problem in scaling, the challenge of either automatically constructing a query, or manually constructing a query with little foreknowledge of its searching capability, was a major challenge for TREC participants. In addition to filtering the relatively large amount of information provided in the topics into queries, the sometimes narrow definition of relevance as stated in the narrative was difficult for most systems to handle.

3.4 The Relevance Judgments

The relevance judgments are of critical importance to a test collection. For each topic it is necessary to compile a list of relevant documents; hopefully as comprehensive a list as possible. For the TREC task, three possible

methods for finding the relevant documents could have been used. In the first method, full relevance judgments could have been made on over one million documents, for each topic, resulting in over 100 million judgments. This was clearly impossible. As a second approach, a random sample of the documents could have been taken, with relevance judgments done on that sample only. The problem with this approach is that a random sample that is large enough to find on the order of 200 relevant documents per topic is a very large random sample, and is likely to result in insufficient relevance judgments. The third method, the one used in TREC, was to make relevance judgments on the sample of documents selected by the various participating systems. This method is known as the pooling method, and has been used successfully in creating other collections [Sparck Jones & van Rijsbergen 1975]. The sample was constructed by taking the top 100 documents retrieved by each system for a given topic and merging them into a pool for relevance assessment. This is a valid sampling method since all the systems used ranked retrieval methods, with those documents most likely to be relevant returned first.

Pooling proved to be an effective method. There was little overlap among the 31 systems in their retrieved documents, although considerably more overlap than in TREC-1.

Table 2. Overlap of Submitted Results

	TREC-2		TREC-1	
	Max	Actual	Max	Actual
Unique Documents Per Topic (Adhoc, 40 runs 23 groups)	4000	1106.0	3300	1278.86
Unique Documents Per Topic (Routing, 40 runs 24 groups)	4000	1465.6	2200	1066.86

Table 2 shows the overlap statistics. The first overlap statistics are for the adhoc topics (test topics against training documents disks 1 and 2), and the second statistics are for the routing topics (training topics against test documents disk 3 only). For example, out of a maximum of 4000 possible unique documents (40 runs times 100 documents), over one-fourth of the documents were actually unique. This means that the different systems were finding different documents as likely relevant documents for a topic. Whereas this might be expected (and indeed has been shown to occur, Katzer et. al. 1982) from widely differing systems, these overlaps were often between two runs for the same system. One reason for the lack of overlap is the very large number of documents that contain many of the same terms as the relevant documents, but the major reason is the very different sets of terms in the constructed queries. This lack of overlap should improve the coverage of the relevance set, and verifies the use of the pooling methodology to produce the sample.

The merged list of results was then shown to the human assessors. Each topic was judged by a single assessor to insure the best consistency of judgment. Varying numbers of documents were judged relevant to the topics. For the TREC-2 adhoc topics (topics 101-150), the median number of relevant documents per topic is 201, down from 277 for topics 51-100 (as used for adhoc topics in TREC-1). Only 11 topics have more than 300 relevant documents, with only 2 topics having more than 500 relevant documents. These topics were deliberately made narrower than topics 51-100 because of a concern that topics with more than 300 relevant documents are likely to have incomplete relevance assessments.

4. Evaluation

An important element of TREC was to provide a common evaluation forum. Standard recall/precision and recall/fallout figures were calculated for each TREC system and these are presented in Appendix A. A chart with additional data about each system is shown in Appendix B. This chart consolidates information provided by the systems that describe features and system timing, and allows some primitive comparison of the amount of effort needed to produce the results.

4.1 Definition of Recall/Precision and Recall/Fallout Curves

Figure 2 shows typical recall/precision curves. The x axis plots the recall values at fixed levels of recall, where

$$Recall = \frac{number\ of\ relevant\ items\ retrieved}{total\ number\ of\ relevant\ items\ in\ collection}$$

The y axis plots the average precision values at those given recall values, where precision is calculated by

$$Precision = \frac{number\ of\ relevant\ items\ retrieved}{total\ number\ of\ items\ retrieved}$$

These curves represent averages over the 50 topics. The averaging method was developed many years ago [Salton & McGill 1983] and is well accepted by the information retrieval community. The curves show system performance across the full range of retrieval, i.e., at the early stage of retrieval where the highly-ranked documents give

high accuracy or precision, and at the final stage of retrieval where there is usually a low accuracy, but more complete retrieval. Note that the use of these curves assumes a ranked output from a system. Systems that provide an unranked set of documents are known to be less effective and therefore were not tested in the TREC program.

The curves in figure 2 show that system A has a much higher precision at the low recall end of the graph and therefore is more accurate. System B however has higher precision at the high recall end of the curve and therefore will give a more complete set of relevant documents, assuming that the user is willing to look further in the ranked list.

A second set of curves was calculated using the recall/fallout measures, where recall is defined as before and fallout is defined as

$$fallout = \frac{number\ of\ nonrelevant\ items\ retrieved}{total\ number\ of\ nonrelevant\ items\ in\ collection}$$

Note that recall has the same definition as the probability of detection and that fallout has the same definition as the probability of false alarm, so that the recall/fallout curves are also the ROC (Relative Operating Characteristic) curves used in signal processing. A sample set of curves corresponding to the recall/precision curves is shown in figure 3. These curves show the same order of performance as do the recall/precision curves and are provided as an alternative method of viewing the results. The present version of the curves is experimental as the curve creation is particularly sensitive to scaling (what range is used for calculating fallout). The high precision section of the curves does not show well in figure 3; the high recall area dominates the curves.

Whereas the recall/precision curves show the retrieval system results as they might be seen by a user (since precision measures the accuracy of each retrieved document as it is retrieved), the recall/fallout curves emphasize the ability of these systems to screen out non-relevant material. In particular the fallout measure shows the discrimination powers of these systems on a large document collection. For example, system A has a fallout of 0.02 at a recall of about 0.48; this means that this system has found almost 50% of the relevant documents, while only retrieving 2% of the non-relevant documents.

4.2 Single-Value Evaluation Measures

In addition to recall/precision and recall/fallout curves, there were 2 single-value measures used in TREC-2.

The first measure, the non-interpolated average precision, corresponds to the area under an ideal (non-interpolated) recall/precision curve. To compute this average, a precision average for each topic is first calculated. This is done by computing the precision after every retrieved relevant document and then averaging these precisions over the total number of retrieved relevant documents for that topic. These topic averages are then combined (averaged) across all topics in the appropriate set to create the non-interpolated average precision for that set.

The second measure used is an average of the precision for each topic after 100 documents have been retrieved for that topic. This measure is useful because it reflects a clearly comprehended retrieval point. It took on added importance in the TREC environment because only the top 100 documents retrieved for each topic were actually assessed. For this reason it produces a guaranteed evaluation point for each system.

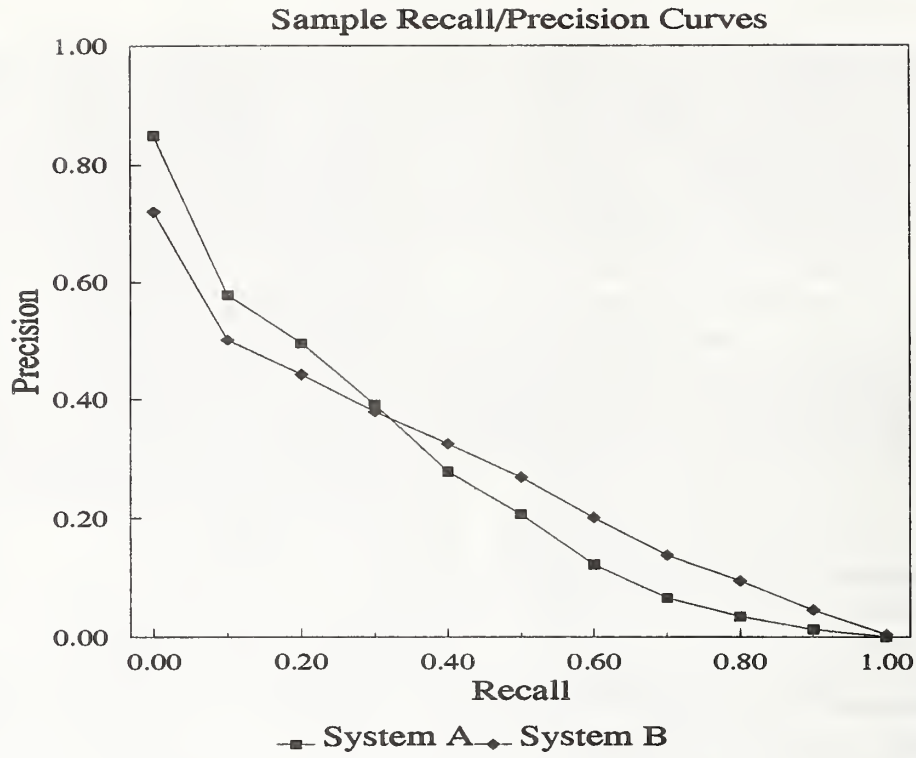
4.3 Problems with Evaluation

Since this was the first time that such a large collection of text has been used in open system evaluation, there were some problems with the existing methods of evaluation. The major problem concerned a thresholding effect caused by the inability to evaluate ALL documents retrieved by a given system.

For TREC-1 the groups were asked to send in only the top 200 documents retrieved by their systems. This artificial document cutoff is relatively low and systems did not retrieve all the relevant documents for most topics within the cutoff. All documents retrieved beyond the 200 mark were considered nonrelevant by default and therefore the recall/precision curves became inaccurate after about 40% recall on average. TREC-2 used the top 1000 documents for evaluation. Figure 4 shows the difference in the curves produced by various evaluation thresholds, including a curve for no threshold (similar to the way evaluation has been done on the smaller collections.). These curves show that the use of a 1000-document cutoff has solved most of the thresholding problem.

Two more issues in evaluation have become important. The first issue involves the need for more statistical evaluation. As will be seen in the results, the recall/precision curves are often close, and there is a need to check if there is truly any statistically significant differences between two systems' results or two sets of results from the same system. This problem is currently under investigation in collaboration with statistical groups experienced in the evaluation of information retrieval systems.

Another issue involves getting beyond the averages to better understand system performance. Because of the huge number of documents and the long topics, it is very



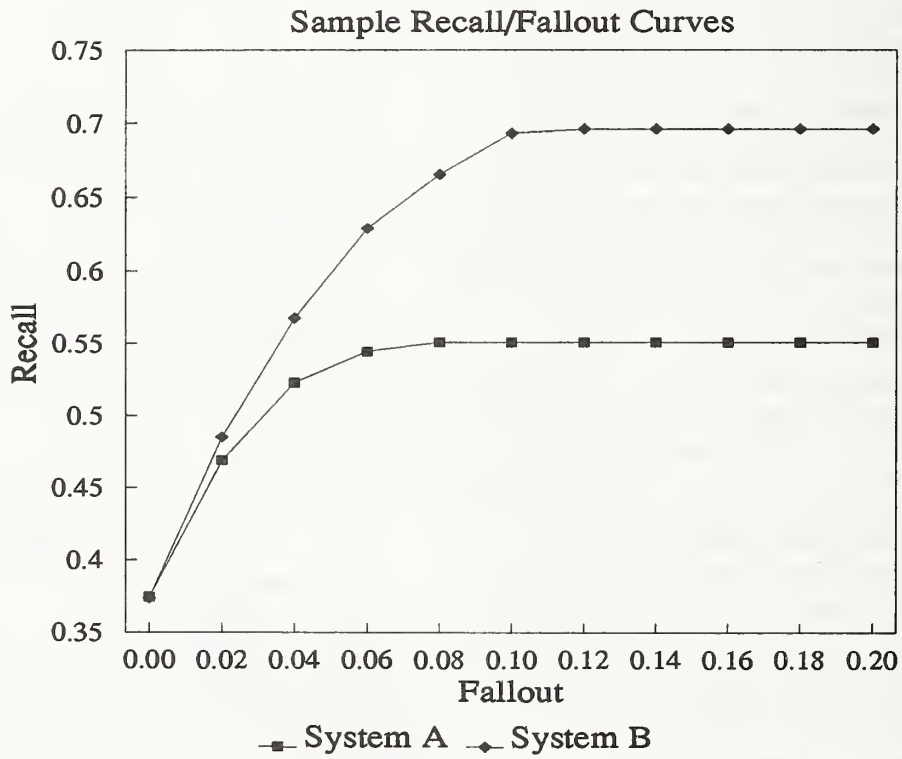


Figure 2. A Sample Recall/Precision Curve.
Figure 3. A Sample Recall/Fallout Curve.

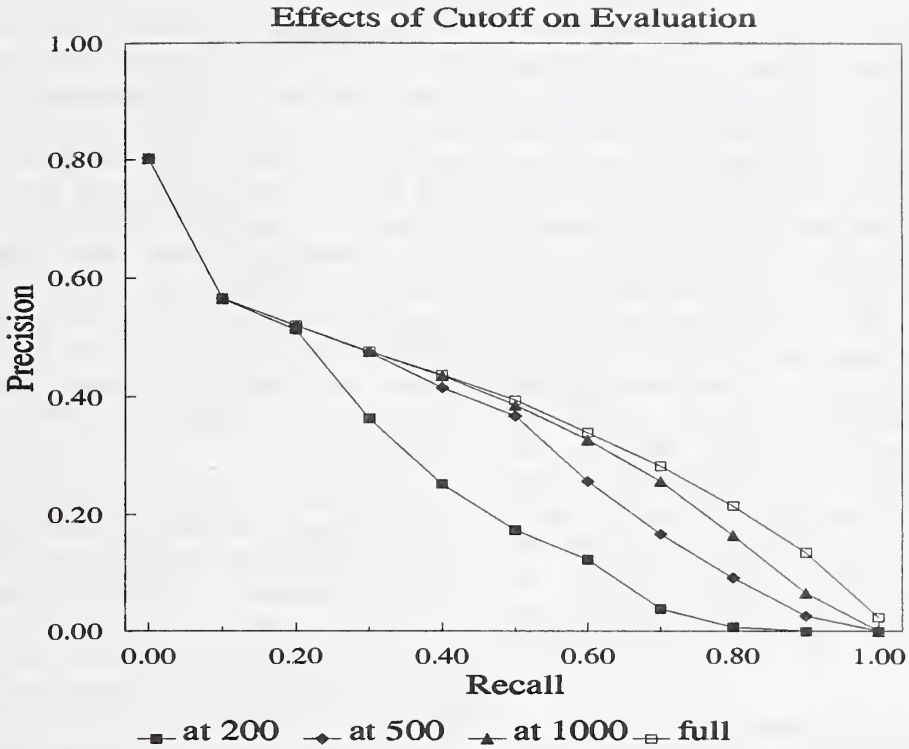


Figure 4. Effect of evaluation cutoffs on recall/precision curves.

difficult to perform failure analysis on the results to better understand the retrieval processes being tested. Without better understanding of underlying system performance, it will be hard to consolidate research progress. Some preliminary analysis of per topic performance is provided in section 6, and and more attention will be given to this problem in the future.

5. Results

5.1 Introduction

In general the TREC-2 results showed significant improvements over the TREC-1 results. Many of the original TREC-1 groups were able to "complete" their system rebuilding and tuning tasks. The results for TREC-2 therefore can be viewed as the "best first-pass" that most groups can accomplish on this large amount of data. The adhoc results in particular represent baseline results from the scaling-up of current algorithms to large test collections. The better systems produced similar results, results that are comparable to those seen using these algorithms on smaller test collections.

The routing results showed even more improvement over TREC-1 routing results. Some of this improvement was due to the availability of large numbers of accurate relevance judgments for training (unlike TREC-1), but most of the improvements came from new research by participating groups into better ways of using the training data.

For full descriptions of each system discussed in this section, see the individual papers in this proceedings.

5.2 Adhoc Results

The adhoc evaluation used new topics (101-150) against the two disks of training documents (disks 1 and 2). There were 44 sets of results for adhoc evaluation in TREC-2, with 32 of them based on runs for the full data set. Of these, 23 used automatic construction of queries, 9 used manual construction, and 2 used feedback.

Figure 5 shows the recall/precision curves for the six TREC-2 groups with the highest non-interpolated average precision using automatic construction of queries. The results marked "INQ001" are the INQUERY system from the University of Massachusetts (see Croft, Callan & Broglio paper). This system uses probabilistic term weighting and a probabilistic inference net to combine various topic and document features. The results marked "dortQ2", "Brkly3" and "crnlL2" are all based on the use of the Cornell SMART system, but with important variations. The "crnlL2" run is the basic SMART system from

Cornell University (see Buckley, Allan & Salton paper), but using less than optimal term weightings (by mistake). The "dortQ2" results from the University of Dortmund come from using polynomial regression on the training data to find weights for various pre-set term features (see Fuhr, Pfeifer, Bremkamp, Pollmann & Buckley paper). The "Brkly3" results from the University of California at Berkeley come from performing logistic regression analysis to learn optimal weighting for various term frequency measures (see Cooper, Chen & Gey paper). The "CLARTA" system from the CLARIT Corporation expands each topic with noun phrases found in a thesaurus that is automatically generated for each topic (see Evans & Lefferts paper). The "lsiasm" results are from Bellcore (see Dumais paper). This group uses latent semantic indexing to create much larger vectors than the more traditional vector-space models such as SMART. The run marked "lsiasm" represents only the base SMART pre-processing results, however. Due to processing errors the "improved" LSI run produced unexpectedly poor results.

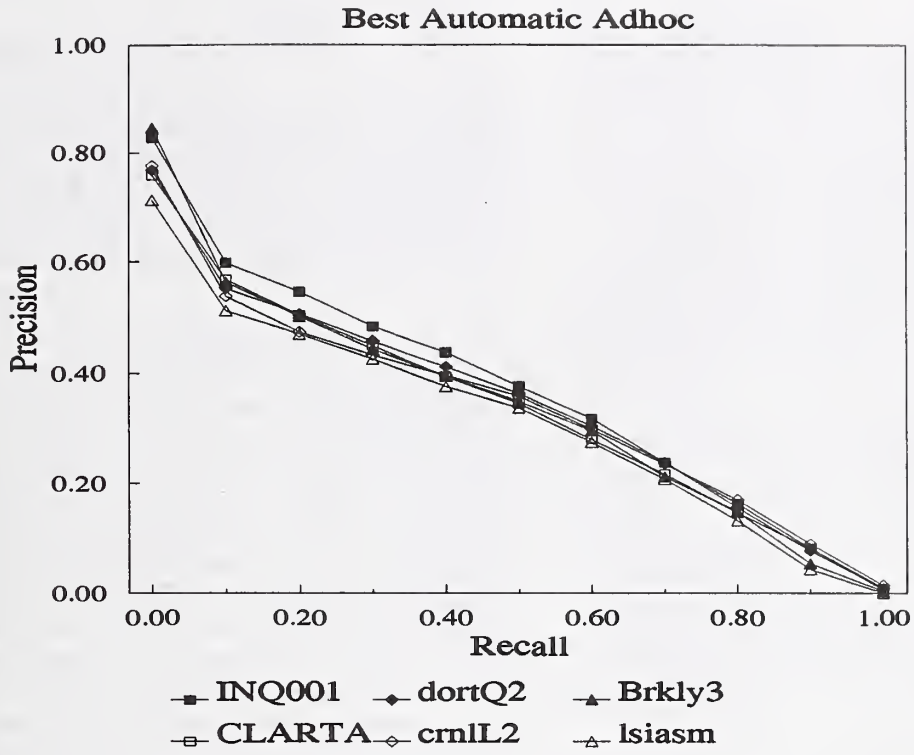
Figure 6 shows the recall/precision curve for the six TREC-2 groups with the highest non-interpolated average precision using manual construction of queries. It should be noted that varying amounts of manual intervention were used. The results marked "INQ002", "siems2", and "CLARTM" are automatically generated queries with manual modifications. The "INQ002" results reflect various manual modifications made to the "INQ001" queries, with those modifications guided by strict rules. The "siems2" results from Siemens Corporate Research, Inc. (see Voorhees paper) are based on the use of the Cornell SMART system, but with the topics manually modified (the "not" phases removed). These results were meant to be the base run for improvements using WordNet, but the improvements did not materialize. The "CLARTM" results represent manual weighting of the query terms, as opposed to the automatic weighting of the terms that was used in "CLARTA." The results marked "Vtcms2", "CnQst2", and "TOPIC2" are produced from queries constructed completely manually. The "Vtcms2" results are from Virginia Tech (see Fox & Shaw paper) and show the effects of combining the results from SMART vector-space queries with the results from manually-constructed soft Boolean P-Norm type queries. The "CnQst2" results, from ConQuest Software (see Nelson paper), use a very large general-purpose semantic net to aid in constructing better queries from the topics, along with sophisticated morphological analysis of the topics. The results marked "TOPIC2" are from the TOPIC system by Verity Corp. (see Lehman paper) and reflect the use of an expert system working off specially-constructed knowledge bases to improve performance.

Several comments can be made with respect to these adhoc results. First, the better results (most of the automatic results and the three top manual results) are very similar and it is unlikely that there is any statistical differences between them. There is clearly no "best" method, and the fact that these systems have very different approaches to retrieval, including different term weighting schemes, different query construction methods, and different similarity match methods implies that there is much more to be learned about effective retrieval techniques. As will be seen in section 6, whereas the averages for the systems may be similar, the systems do better on different topics and retrieve different subsets of the relevant documents.

A second point that should be made is that the automatic query construction methods continue to perform as well as the manual construction methods. Two groups (the INQUERY system and the CLARIT system) did explicit comparision of manually-modified queries vs those that were not modified and concluded that manual modification provided no benefits. The three sets of results based on completely manually-generated queries had even poorer performance than the manually-modified queries. Note that this result is specific to the very rich TREC topics; it is not clear that this will hold for the short topics normally seen in other retrieval environments.

As a final point, it should be noted that these adhoc results represent significant improvements over the results from TREC-1. Figure 7 shows a comparison of results for a typical system in TREC-1 and TREC-2. Some of this improvement is due to improved evaluation, but the difference between the curve marked "TREC-1" and the curve marked "TREC-2 looking at top 200 only" shows significant performance improvement. Whereas this improvement could represent a difference in topics (the TREC-1 curve is for topics 51-100 and the TREC-2 curves are for topics 101-150), the TREC-2 topics are generally felt to be more difficult and therefore this improvement is likely to be an understatement of the actual improvements.

Only two groups worked with less than the full document collection. Figure 9 shows the results for the one group with official TREC-2 category B results (the results from UCLA were received after the deadline). This figure shows the best results from New York University (see Strzalkowski & Carballo paper), compared with a category B version of the Cornell SMART results. The "nyuir3" results reflect a very intensive use of natural language processing (NLP) techniques, including a parse of the documents to help locate syntactic phrases, context-sensitive expansion of the queries, and other NLP improvements on statistical techniques.



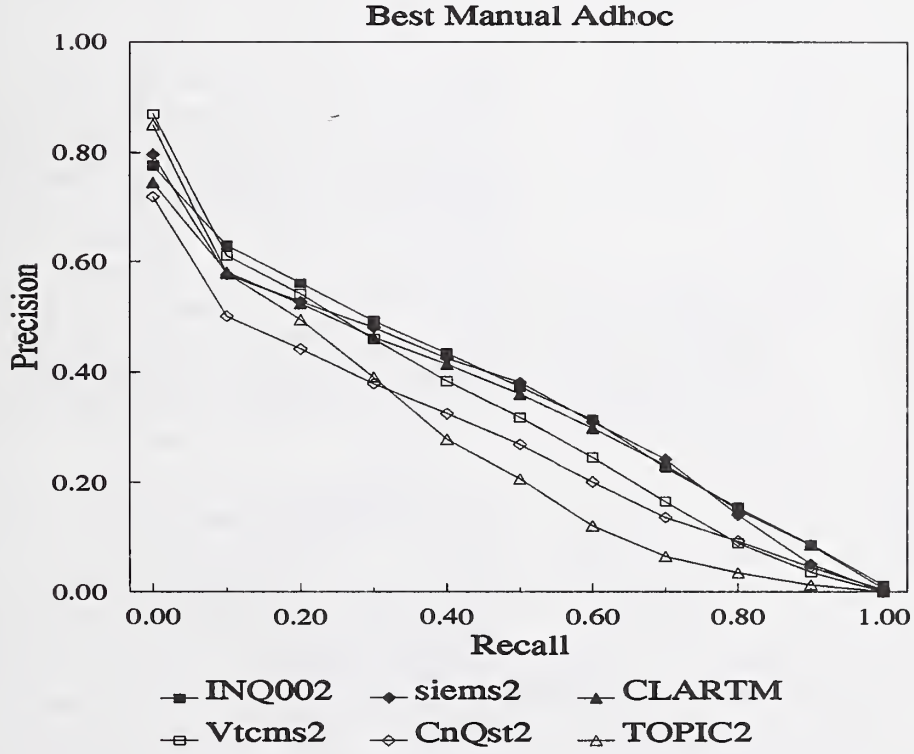
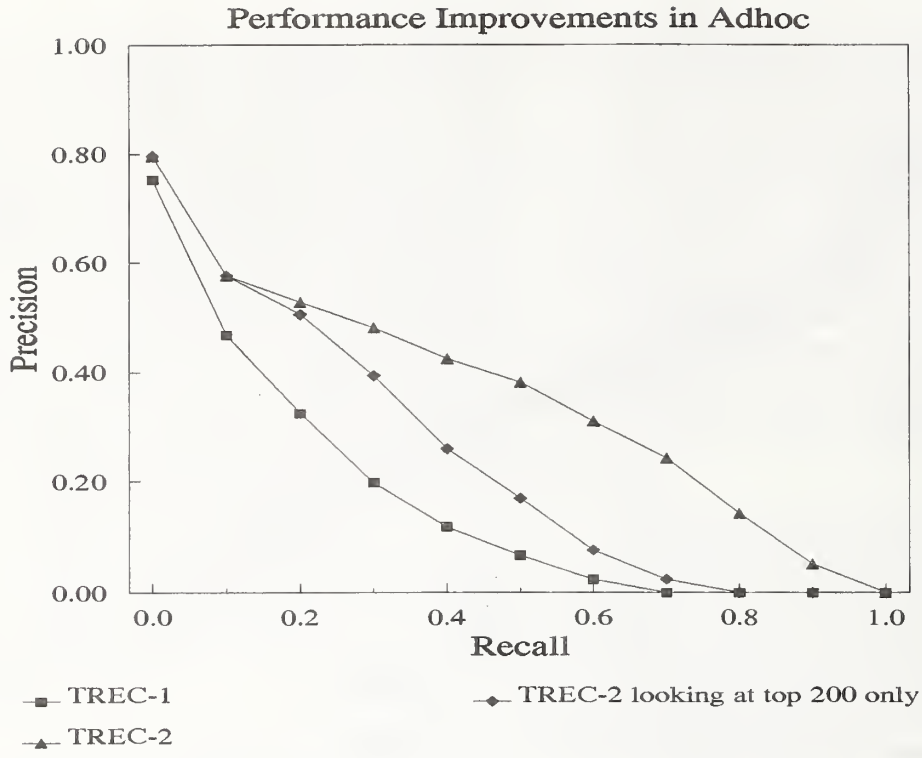


Figure 5. Best Automatic Adhoc Results.
Figure 6. Best Manual Adhoc Results.



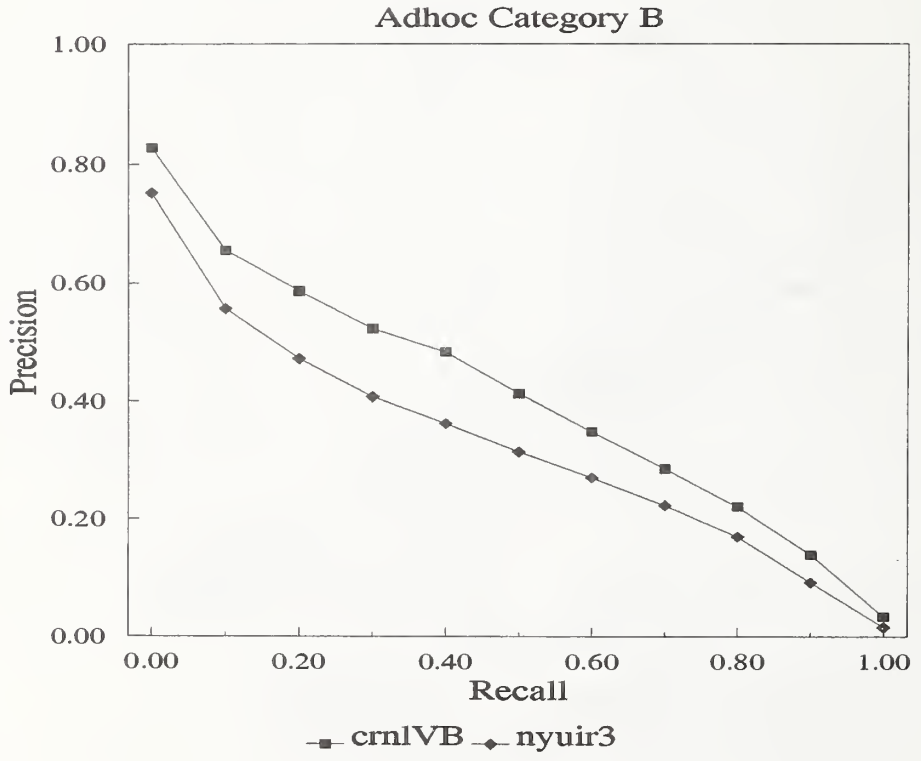


Figure 7. Typical Improvements in Adhoc Results.
Figure 8. Category B Adhoc Results.

5.3 Routing Results

The routing evaluation used a subset of the training topics (topics 51-100 were used) against the new disk of test documents (disk 3). There were 40 sets of results for routing evaluation, with 32 of them based on runs for the full data set. Of the 32 systems using the full data set, 23 used automatic construction of queries, and 9 used manual construction.

Figure 9 shows the recall/precision curves for the six TREC-2 groups with the highest non-interpolated average precision using automatic construction of the routing queries. Again three systems are based on the Cornell SMART system. The plot marked "crnlC1" is the actual SMART system, using the basic Rocchio relevance feedback algorithms, and adding many terms (up to 500) from the relevant training documents to the terms in the topic. The "dortP1" results come from using a probabilistically-based relevance feedback instead of the vector-space algorithm, and adding only 20 terms from the relevant documents to each query. These two systems have the best routing results. The "Brkly5" system uses logistic regression on both the general frequency variables used in their adhoc approach and on the query-specific relevance data available for training with the routing topics. The results marked "cityr2" are from City University, London (see Robertson, Walker, Jones, Hancock-Beaulieu & Gafford paper). This group automatically selected variable numbers of terms (10-25) from the training documents for each topic (the topics themselves were not used as term sources), and then used traditional probabilistic reweighting to weight these terms. The "INQ003" results also use probabilistic reweighting, but use the topic terms, expanded by 30 new terms per topic from the training documents. The results marked "lsir2" are more latent semantic indexing results from Bellcore. This run was made by creating a filter of the singular-value decomposition vector sum or centroid of all relevant documents for a topic (and ignoring the topic itself).

Figure 10 shows the recall/precision curves for the six TREC-2 groups with the highest non-interpolated average precision using manual construction of the routing queries. The results marked "INQ004" are from the INQRY system using an inferential combination of the "INQ003" queries and manually modified queries created from the topic. The "trw2" results represent an adaptation of the TRW Fast Data Finder pattern matching system to allow use of term weighting (see Mettler paper). The queries were manually constructed and the term weighting was learned from the training data. The "gecrd1" results from GE Research and Development Center (see Jacobs paper) also come from manually constructed queries, but using a general-purpose lexicon and the training data to suggest input to the Boolean pattern matcher.

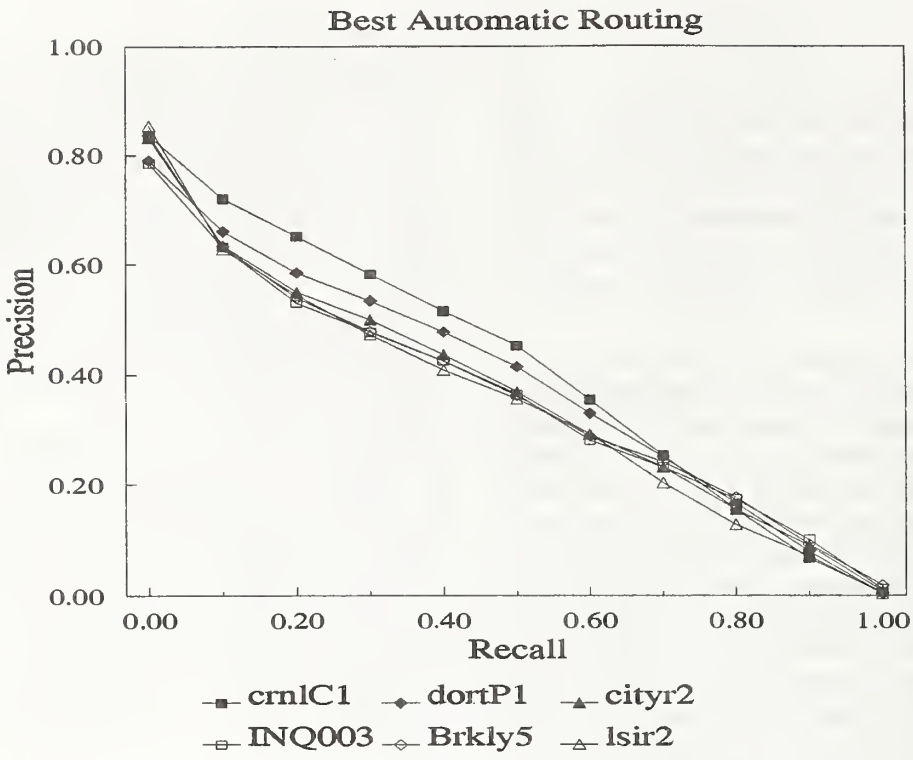
The results marked "CLARTM" are similar to the "CLARTM" adhoc results except that the training documents were used as the source for thesaurus building, as opposed to using the top set of retrieved documents. The "rutcombx" results from Rutgers University (see Belkin, Kantor, Cool & Quatrain paper) come from combining 5 sets of manually generated Boolean queries to optimize performance for each topic. The results marked "TOPIC2" are from the TOPIC system and reflect the use of an expert system working off specially-constructed knowledge bases to improve performance.

As was the case with the adhoc topics, the automatic query construction methods continue to perform as well as, or in this case, better than the manual construction methods. A comparision of the two INQRY runs illustrates this point and shows that all six results with manually generated queries perform worse than the six runs with automatically-generated queries. The availability of the training data allows an automatic tuning of the queries that would be difficult to duplicate manually without extensive analysis.

Unlike the adhoc results, there are two runs ("crnlC1" and "dortP1") that are clearly better than the others, with a significant difference between the "crnlC1" results and the "dortP1" results and also significant differences between these results and the rest of the automatically-generated query results. In particular the Cornell group's ability to effectively use many terms (up to 500) for query expansion was one of the most interesting findings in TREC-2 and represents a departure from past results (see Buckley, Allan, & Salton paper for more on this).

As a final point, it should be noted that the routing results also represent significant improvements over the results from TREC-1. Figure 11 shows a comparison of results for a typical system in TREC-1 and TREC-2. Some of this improvement is due to the improved evaluation techniques, but the difference between the curve marked "TREC-1" and the curve marked "TREC-2 looking at top 200 only" shows significant performance improvement. There is even more improvement for the routing results than for the adhoc results, due to better training data (mostly non-existent for TREC-1) and to major efforts by many groups in new routing algorithm experiments.

Only four groups worked with less than the full document collection. Figure 12 shows the results for two of the groups in category B compared with a category B version of the Cornell SMART results. These curves show the results of runs from New York University (that were done in a similar method as that used for the adhoc results) and results from Dalhousie University.



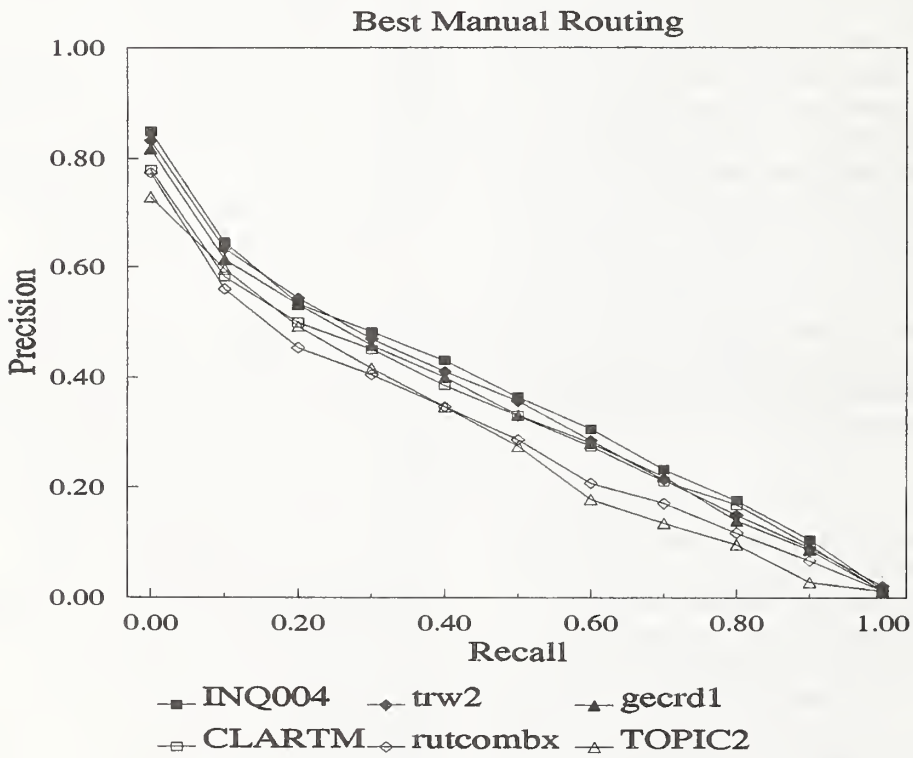
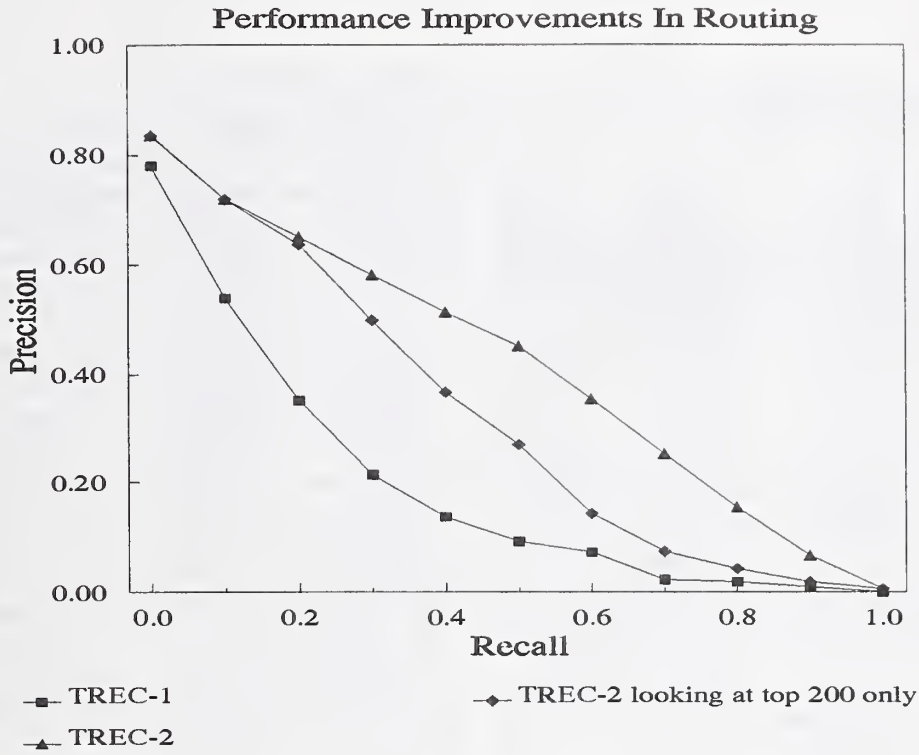


Figure 9. Best Automatic Routing Results.
Figure 10. Best Manual Routing Results.



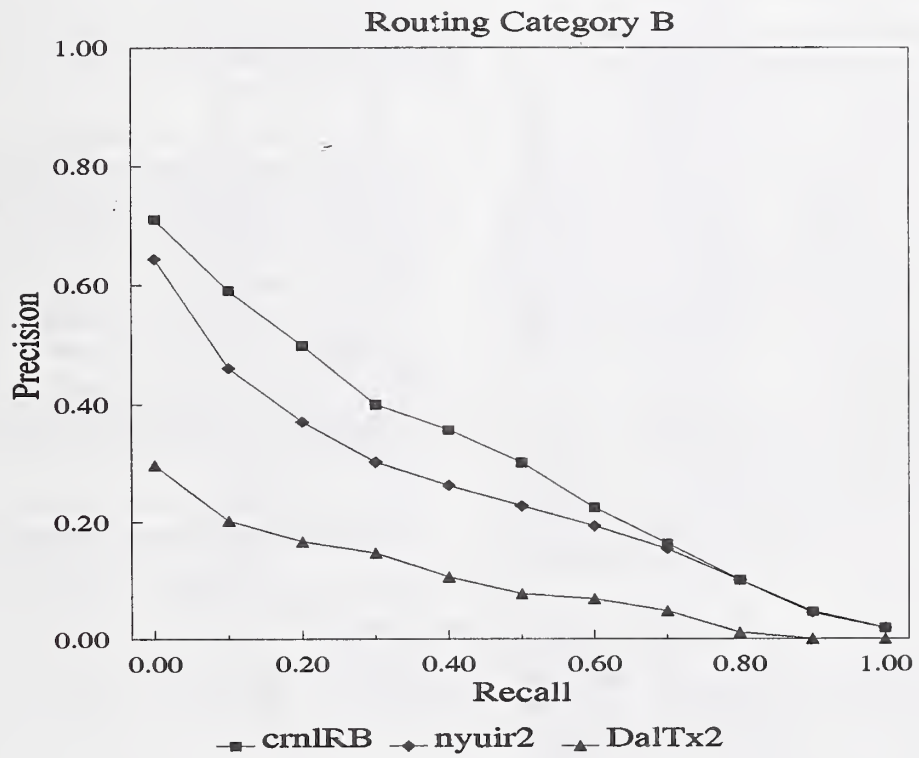


Figure 11. Typical Improvements in Routing Results.
Figure 12. Category B Routing Results.

6. Some Preliminary Analysis

6.1 Introduction

The recall/precision curves shown in section 5 represent the average performance of the various systems on the full sets of topics. It is important to look beyond these averages in order to learn more about how a given system is performing and to discover some generalizable principles of retrieval.

Individual systems are able to do this by performing failure analysis (see Dumais paper in this proceedings for a good example) and by running specific experiments to test hypotheses on retrieval behavior within a given system. However, additional information can be gained by doing some cross-system comparison: information about specific system behavior and information about generalized information retrieval principles. One way to do this is to examine system behavior with respect to test collection characteristics. A second method is to compare system behavior on a topic by topic basis.

6.2 The Effects of Test Collection Characteristics

One particular test collection characteristic is the length of documents, both the average length of documents in a collection, and the variation in document length across a collection. Document length has significant effect on system performance. A term that appears 10 times in a "short" document is likely to be more important to that document than if the same term appeared 10 times in a "long" document. Table 3 shows system performance across the different document subcollections for each of the adhoc topics, listing the total number of documents that were retrieved by the system as well as the number of relevant documents that were retrieved.

Two particular points can be seen from table 3. First, the better systems retrieve about 50% relevant documents from all the subcollections except the *Federal Register* (FR). For this subcollection the retrieval rates are in the 25% range because the varied length of these documents makes retrieval difficult.

The second point concerning table 3 is that the retrieval rate across the subcollections is highly varied among the systems. For example the "Brkly3" results show that many fewer *Federal Register* documents and more *AP* were retrieved than for the INQUERY system, whereas the "CLARTA" results show more *DOE* abstracts and fewer *Wall Street Journal* being retrieved. These "biases" towards particular subcollections reflect the methods used by systems such as the length normalization issues, domain concentrations of terminology, and methods used to "merge" results across subcollections (often implicit merges during indexing).

A second test collection characteristic worth examining is the varied broadness and varied difficulty of the topics. An analysis was done [Harman 1994] to find the topics for which the systems retrieved the lowest percentage of the relevant documents on average. These topics are 61, 67, 76, 77, 81, 85, 90, 91, 93, and 98 for the routing topics and 101, 114, 120, 121, 124, 131, 139, 140, 141, and 149 for the adhoc topics. Tables 4 and 5 show the top 8 system runs for the individual topics based in the average precision (noninterpolated). These tables mix automatic, manual, and feedback results for category A, and also category B results, so they should be interpreted carefully. However they do demonstrate that no consistent patterns appear for the "hard" topics. The two best routing runs ("crnlC1" and "dortP1") only do well on about half of these topics, and the adhoc results are even more varied. Often systems that do not perform well on average are the top performing system for a given topic. This verifies that, as usual, the variation across the topics is greater than the variation across systems.

6.3 Cross-System Analysis

Tables 4 and 5 not only show the wide variation in system performance, but also raise several questions about system performance in general.

1. Does better average performance for a system result from better performance on most topics or from comparable performance on most topics and significantly better performance on other topics?
2. If two systems perform similarly on a given topic, does that mean that they have retrieved a large proportion of the same relevant documents?
3. Do systems that use "similar" approaches have a high overlap in the particular relevant documents they retrieve?
4. And, if number 3 is not true, what are the issues that affect high overlap of relevant documents?

Work is ongoing at NIST on these questions and other related issues.

Table 3. Number of Documents Retrieved/Relevant by Document Subcollection

Run Tag	AP	DOE	FR	WSJ	ZIFF
Brkly3	2414/1155	293/155	97/22	1847/831	349/121
citri1	2152/970	474/179	129/16	1865/770	380/177
citri2	2206/1019	348/156	250/67	1814/756	382/179
cityau	1578/794	93/46	1359/147	1603/661	367/108
citymf	1939/1226	24/21	584/146	2026/812	427/173
CLARTA	2034/1048	403/170	412/95	1795/819	356/190
CLARTM	2131/1087	388/208	315/87	1769/820	397/221
CnQst1	2272/921	254/112	311/74	1763/689	400/181
CnQst2	2214/980	191/94	453/93	1787/738	355/184
crnlL2	1914/944	648/240	155/33	1774/773	509/213
crnlV2	2164/1083	687/236	79/22	1600/682	470/194
dortL2	2081/1053	305/169	473/78	1818/815	323/166
dortQ2	2205/1053	357/171	186/44	1924/874	328/171
erima1	1077/501	85/51	1364/110	2251/752	223/94
erima2	1267/614	122/68	1219/125	2124/773	268/133
gecrd2	2250/852	294/91	319/68	1952/743	185/77
HNCad1	2140/1042	409/145	164/53	1875/839	412/181
HNCad2	2163/1286	306/159	171/67	1974/1005	386/237
INQ001	2031/1071	206/107	297/115	2184/1023	282/151
INQ002	2087/1111	201/120	276/111	2141/1010	295/177
lsia1	2278/771	587/124	124/0	1448/376	563/61
lsiasm	2168/1052	711/211	70/17	1607/690	444/183
nyuir1	0/0	0/0	0/0	5000/1360	0/0
nyuir2	0/0	0/0	0/0	5000/1547	0/0
nyuir3	0/0	0/0	0/0	5000/1547	0/0
pircs3	2109/1021	358/152	246/86	1999/835	288/139
pircs4	2108/1014	342/148	254/85	2012/863	284/137
prceo1	1099/1024	315/83	1178/205	1377/980	1031/277
prceo1	1667/1024	695/83	381/205	1350/980	907/277
rutcomb1	1029/368	181/72	112/18	963/312	215/79
rutfmed	945/309	131/46	161/9	963/282	200/77
schau1	2038/901	534/189	173/18	1778/706	477/186
siems2	2225/1147	631/218	62/8	1655/770	427/202
siems3	2238/1173	654/208	53/7	1619/764	436/194
TMC8	2054/859	146/44	763/59	1472/526	565/183
TMC9	1923/802	77/29	975/63	1401/507	624/171
TOPIC2	2292/996	152/98	344/100	1762/889	384/229
UREKA2	385/215	0/0	4003/87	354/144	258/10
UREKA3	755/405	5/2	2654/67	1045/348	441/22
uicah	1612/628	234/104	797/137	1846/356	511/167
VTcms2	2110/1130	232/107	444/95	1859/894	355/169
totals	71354/4630	12073/669	21407/396	79396/3929	15504/1154

Table 4. System Rankings (using Average Precision) on Individual Topics

51	nyuir2	nyuir1	gecrd1	TOPIC2	ADS2	cityr2	INQ004	INQ003
52	INQ004	INQ003	Brkly4	pircs2	VTcms2	gecrd1	pircs1	trw1
53	gecrd1	nyuir2	trw2	nyuir1	CLARTM	CLARTA	dortP1	INQ003
54	siems1	crnlR1	schau1	Brkly4	INQ003	crnlC1	lsir1	CLARTM
55	dortP1	crnlR1	crnlC1	lsir2	dortV1	CLARTM	cityr1	CLARTA
56	trw2	dortP1	dortV1	INQ003	INQ004	HNCrt1	crnlR1	crnlC1
57	INQ003	lsir2	INQ004	trw2	crnlC1	TMC6	VTcms2	crnlR1
58	nyuir2	nyuir1	rutcombx	INQ003	lsir2	INQ004	gecrd1	Brkly5
59	trw2	Brkly5	gecrd1	lsir1	HNCrt1	VTcms2	HNCrt2	lsir2
60	dortP1	dortV1	rutcombx	crnlR1	INQ004	crnlC1	INQ003	TOPIC2
61	TOPIC2	rutcombx	Brkly4	idsra2	cityr2	lsir1	INQ004	Brkly5
62	crnlR1	crnlC1	dortP1	lsir1	CLARTA	Brkly4	CLARTM	Brkly5
63	dortV1	crnlC1	pircs2	crnlR1	pircs1	siems1	HNCrt1	dortP1
64	nyuir2	lsir2	INQ004	INQ003	Brkly5	crnlC1	crnlR1	cityr2
65	crnlC1	dortV1	dortP1	HNCrt1	crnlR1	trw2	HNCrt2	lsir1
66	pircs2	pircs1	dortP1	dortV1	crnlR1	crnlC1	siems1	INQ004
67	crnlR1	crnlC1	INQ004	nyuir2	dortP1	cityr2	lsir2	INQ003
68	Brkly5	crnlC1	cityr1	cityr2	trw2	INQ003	lsir2	CLARTA
69	erimr1	Brkly5	dortV1	cityr2	cityr1	erimr2	lsir1	Brkly4
70	TMC6	TMC7	rutcombx	VTcms2	HNCrt2	Brkly5	INQ004	cityr2
71	crnlR1	crnlC1	HNCrt2	siems1	CLARTM	HNCrt1	CLARTA	lsir2
72	crnlR1	crnlC1	dortP1	siems1	INQ003	Brkly5	INQ004	cityr1
73	INQ003	crnlR1	cityr2	INQ004	crnlC1	trw2	dortP1	dortV1
74	crnlR1	rutcombx	crnlC1	CLARTA	Brkly5	dortP1	siems1	dortV1
75	crnlC1	ADS2	crnlR1	trw1	lsir2	dortP1	cityr2	nyuir2
76	trw2	cityr2	TOPIC2	TMC6	TMC7	crnlC1	crnlR1	INQ003
77	crnlR1	crnlC1	INQ003	CLARTM	dortV1	dortP1	INQ004	CLARTA
78	rutcombx	TOPIC2	INQ004	CLARTM	INQ003	dortV1	pircs2	CLARTA
79	cityr2	crnlR1	crnlC1	INQ004	dortP1	gecrd1	lsir2	INQ003
80	trw1	crnlC1	crnlR1	cityr1	Brkly5	INQ003	INQ004	cityr2
81	gecrd1	TMC7	TMC6	cityr2	trw2	VTcms2	HNCrt2	cityr1
82	CLARTM	CLARTA	trw2	Brkly5	pircs1	pircs2	dortV1	dortP1
83	TOPIC2	gecrd1	trw1	crnlC1	HNCrt1	crnlR1	cityr2	cityr1
84	dortP1	crnlC1	lsir2	gecrd1	crnlR1	dortV1	trw1	VTcms2
85	crnlR1	crnlC1	dortP1	Brkly5	trw2	nyuir2	dortV1	siems1
86	gecrd1	VTcms2	lsir2	lsir1	cityr1	crnlR1	cityr2	crnlC1
87	lsir2	gecrd1	cityr1	cityr2	HNCrt1	Brkly5	crnlC1	HNCrt2
88	crnlC1	cityr2	crnlR1	Brkly4	dortP1	lsir2	dortV1	Brkly5
89	trw2	nyuir1	TOPIC2	TMC6	HNCrt1	uicr1	HNCrt2	gecrd1
90	gecrd1	trw1	crnlC1	crnlR1	schau1	VTcms2	Brkly5	dortP1
91	trw1	INQ004	schau1	Brkly5	trw2	TOPIC2	HNCrt2	HNCrt1
92	gecrd1	crnlR1	lsir2	crnlC1	CLARTM	CLARTA	nyuir1	INQ003
93	INQ004	rutcombx	INQ003	TMC6	trw1	Brkly5	TMC7	gecrd1
94	lsir2	crnlC1	cityr2	gecrd1	INQ004	CLARTM	trw2	cityr1
95	VTcms2	gecrd1	crnlC1	Brkly5	crnlR1	Brkly4	trw1	siems1
96	dortP1	TOPIC2	cityr1	dortV1	cityr2	lsir2	crnlC1	rutcombx
97	idsra2	HNCrt1	nyuir2	dortP1	HNCrt2	lsir2	crnlC1	TOPIC2
98	HNCrt1	HNCrt2	crnlC1	trw2	DalTx2	INQ004	crnlR1	dortP1
99	lsir2	crnlR1	dortP1	CLARTA	crnlC1	CLARTM	dortV1	cityr2
100	crnlC1	crnlR1	dortP1	lsir2	dortV1	CLARTA	CLARTM	lsir1

Table 5. System Rankings (using Average Precision) on Individual Topics

101	rutcomb1	VTcms2	crnlV2	INQ002	dortQ2	pircs3	Brkly3	CLARTM
102	crnlL2	crnlV2	VTcms2	siems3	dortL2	INQ002	siems2	CLARTM
103	siems3	siems2	schau1	citri1	crnlV2	lsiasm	HNCad2	HNCad1
104	dortQ2	CLARTM	CLARTA	pircs4	pircs3	dortL2	HNCad2	lsiasm
105	citri2	lsiasm	citri1	siems2	siems3	crnlV2	schau1	crnlL2
106	VTcms2	INQ002	INQ001	TOPIC2	pircs4	pircs3	CLARTM	dortL2
107	CnQst1	CnQst2	rutcomb1	TOPIC2	VTcms2	INQ002	rutfmed	CLARTM
108	citri1	dortQ2	siems3	VTcms2	siems2	HNCad2	schau1	dortL2
109	dortL2	crnlL2	dortQ2	CLARTA	CLARTM	pircs3	crnlV2	pircs4
110	INQ002	INQ001	Brkly3	dortQ2	nyuir3	nyuir2	cityau	siems2
111	CLARTA	CLARTM	INQ001	dortQ2	Brkly3	siems2	siems3	pircs4
112	INQ002	INQ001	VTcms2	nyuir2	nyuir3	HNCad1	HNCad2	CnQst2
113	VTcms2	crnlL2	dortL2	crnlV2	nyuir1	siems2	CLARTM	INQ002
114	INQ002	cityau	VTcms2	INQ001	siems3	siems2	lsia1	TOPIC2
115	nyuir2	nyuir3	nyuir1	siems2	dortL2	crnlV2	siems3	crnlL2
116	VTcms2	CLARTA	HNCad2	HNCad1	siems3	siems2	CLARTM	Brkly3
117	citri2	citri1	dortQ2	INQ001	TMC8	lsiasm	gecrd2	schau1
118	nyuir2	nyuir3	nyuir1	TOPIC2	citymf	dortQ2	CLARTA	INQ001
119	nyuir1	nyuir2	nyuir3	INQ002	INQ001	dortQ2	citymf	VTcms2
120	citymf	nyuir2	nyuir3	nyuir1	CnQst2	CnQst1	VTcms2	erima2
121	TOPIC2	CLARTM	VTcms2	Brkly3	nyuir1	prceo1	INQ002	rutfmed
122	siems2	siems3	INQ002	INQ001	dortQ2	Brkly3	CLARTM	crnlV2
123	nyuir1	nyuir2	nyuir3	CLARTA	INQ001	INQ002	CLARTM	pircs4
124	nyuir2	nyuir3	nyuir1	dortL2	dortQ2	INQ001	Brkly3	TMC9
125	crnlV2	Brkly3	crnlL2	CLARTM	siems3	CLARTA	pircs4	pircs3
126	siems3	crnlL2	siems2	Brkly3	crnlV2	INQ002	CLARTM	INQ001
127	cityau	Brkly3	CLARTA	HNCad2	INQ001	INQ002	siems2	siems3
128	VTcms2	CLARTA	siems3	siems2	CLARTM	TOPIC2	citri1	lsiasm
129	INQ001	INQ002	cityau	CLARTM	siems2	Brkly3	crnlL2	CLARTA
130	INQ002	INQ001	dortQ2	crnlL2	pircs4	CLARTM	dortL2	pircs3
131	TOPIC2	VTcms2	HNCad1	HNCad2	siems3	Brkly3	siems2	INQ002
132	dortL2	INQ001	INQ002	citri1	citri2	dortQ2	HNCad2	crnlL2
133	CnQst2	CnQst1	rutcomb1	pircs4	INQ002	pircs3	cityau	INQ001
134	crnlL2	dortL2	nyuir1	nyuir2	nyuir3	INQ002	INQ001	dortQ2
135	nyuir2	nyuir3	nyuir1	Brkly3	INQ001	INQ002	siems3	siems2
136	VTcms2	CnQst1	CnQst2	CLARTM	pircs4	CLARTA	dortQ2	TOPIC2
137	CLARTA	nyuir2	nyuir3	Brkly3	siems2	siems3	CLARTM	nyuir1
138	nyuir2	nyuir3	rutfmed	rutcomb1	nyuir1	schau1	gecrd2	citri1
139	nyuir2	nyuir3	nyuir1	VTcms2	dortL2	HNCad2	dortQ2	HNCad1
140	nyuir2	nyuir3	nyuir1	dortQ2	dortL2	INQ002	siems3	siems2
141	VTcms2	INQ002	CnQst2	INQ001	Brkly3	dortL2	dortQ2	CnQst1
142	dortQ2	siems2	crnlL2	VTcms2	siems3	CLARTM	crnlV2	Brkly3
143	INQ002	INQ001	siems2	siems3	crnlL2	crnlV2	nyuir2	nyuir3
144	VTcms2	Brkly3	citymf	crnlV2	siems3	lsiasm	siems2	HNCad2
145	crnlL2	crnlV2	dortL2	CLARTM	nyuir1	siems3	siems2	dortQ2
146	Brkly3	siems3	siems2	lsiasm	crnlV2	schau1	CLARTM	citri1
147	HNCad2	HNCad1	VTcms2	citri1	INQ002	INQ001	citymf	CLARTA
148	lsiasm	crnlL2	crnlV2	siems2	siems3	Brkly3	dortL2	dortQ2
149	nyuir1	CnQst2	TOPIC2	CnQst1	CLARTA	rutfmed	Brkly3	rutcomb1
150	crnlL2	dortQ2	CLARTM	siems3	INQ002	INQ001	crnlV2	siems2

6.4 Summary

The TREC-2 conference demonstrated a wide range of different approaches to the retrieval of text from large document collections. There was significant improvement in retrieval performance over that seen in TREC-1, especially in the routing task. The availability of large amounts of training data for routing allowed extensive experimentation in the best use of that data, and many different approaches were tried in TREC-2. The automatic construction of queries from the topics continued to do as well as, or better than, manual construction of queries, and this is encouraging for groups supporting the use of simple natural language interfaces for retrieval systems.

How well is the TREC initiative meeting its goals? There is certainly increased research using a much larger collection than had previously been tested. This leads not only to discovering interesting research problems, but also to developing algorithms that are ripe for transfer into commercial systems. The conference itself provided the opportunity for this; there was open exchange between the research groups in universities and the research groups in commercial organizations and this is a very critical part of technology transfer.

There will be a third TREC conference in 1994, and all the systems that participated in TREC-2 will be back, along with additional groups.

7. References

Harman, D. (1993) (Ed.).*The First Text REtrieval Conference (TREC-1).* National Institute of Standards and Technology Special Publication 500-207,

Harman, D. (1994). Data Preparation. In: Merchant R. (Ed.).*The Proceedings of the TIPSTER Text Program - Phase I.* San Mateo, California: Morgan Kaufmann Publishing Co., 1994.

Katzer, J., McGill, M.J., Tessier, J.A., Frakes, W., and DasGupta, P. (1982). A Study of the Overlap among Document Representations. *Information Technology: Research and Development*, 1(2), 261-274.

Salton, G. and McGill, M. (1983). *Introduction to Modern Information Retrieval.* New York, NY.: McGraw-Hill.

Sparck Jones, K. and Van Rijsbergen, C. (1975). *Report on the Need for and Provision of an "Ideal" Information Retrieval Test Collection*, British Library Research and Development Report 5266, Computer Laboratory, University of Cambridge. 1993.

Okapi at TREC–2

S E Robertson* S Walker* S Jones* M M Hancock-Beaulieu* M Gatford*

Advisers: E Michael Keen (University of Wales, Aberystwyth), Karen Sparck Jones (Cambridge University), Peter Willett (University of Sheffield)

1 Introduction

This paper reports on City University's work on the TREC–2 project from its commencement up to November 1993. It includes many results which were obtained after the August 1993 deadline for submission of official results.

For TREC–2, as for TREC–1, City University used versions of the Okapi text retrieval system much as described in [2] (see also [3, 4]). Okapi is a simple and robust set-oriented system based on a generalised probabilistic model with facilities for relevance feedback, but also supporting a full range of deterministic Boolean and quasi-Boolean operations.

For TREC–1 [1] the "standard" Robertson–Sparck Jones weighting function was used for all runs (equation 1, see also [5]). City's performance was not outstandingly good among comparable systems, and the intention for TREC–2 was to develop and investigate a number of alternative probabilistic term-weighting functions. Other possibilities included varieties of query expansion, database models enabling paragraph retrieval and the use of phrases obtained by query parsing.

Unfortunately, a prolonged disk failure prevented realistic test runs until almost the deadline for submission of results. A full inversion of the disks 1 and 2 database was only achieved a few hours before the final automatic runs. None of the new weighting functions (Section 1.1) was properly evaluated until after the results had been submitted to NIST; we have since discovered that several of these models perform much better than the weighting functions used for the official runs, and most of the results reported herein are from these later runs.

1.1 The system

The Okapi system comprises a search engine or basic search system (BSS), a low level interface used mainly for batch runs and a user interface for the manual search experiments (Section 5), together with data conversion and inversion utilities. The hardware consisted of Sun SPARC machines with up to 40 MB of memory, and, occasionally, about 8 GB of disk storage. Several databases were used from time to time: full disks 1 and 2, AP (disk 1) and WSJ (disk 1), full disk 3. All inverted indexes included complete within-document positional information, enabling term frequency and term proximity to be used. Typical index size overhead was around 80% of the textfile size. Elapsed time for inversion of disks 1 and 2 was about two days. Running a single topic with evaluation averaged from about one minute to ten minutes, depending strongly on the number of query terms. All preliminary evaluation used the "old" SMART evaluation program. Runs tabulated in this paper used an early version of the new evaluation program, for which we are grateful to Chris Buckley of Cornell University.

2 Some new probabilistic models

Statistical approaches to information retrieval have traditionally (to over-simplify grossly) taken two forms:

(a) approaches based on formal models, where the model specifies an exact formula;

(b) ad-hoc approaches, where formulae are tried because they seem to be plausible.

Both categories have had some notable successes. A more recent variant is the regression approach of Fuhr and Cooper (see, for example, [6]), which incorporates ad-hoc choice of independent variables and functions of them with a formal model for assessing their value in retrieval, selecting from among them and assigning weights to them.

One problem with the formal model approach is that it is often very difficult to take into account the wide variety of variables that are thought or known to influence retrieval. The difficulty arises either because there is no known basis for a model containing such variables, or because any such model may simply be too complex to give a usable exact formula.

One problem with the ad-hoc approach is that there is little guidance as to how to deal with specific variables—one has to guess at a formula and try it out. This

*Centre for Interactive Systems Research, Department of Information Science, City University, Northampton Square, London EC1V 0HB, UK

problem is also apparent in the regression approach—although "trying it out" has a somewhat different sense here (the formula is tried in a regression model, rather than in a retrieval test).

The discussions of Sections 2.1 and 2.3 exemplify an approach which may offer some reconciliation of these ideas. Essentially it is to take a formal model which provides an exact but intractable formula, and use it to suggest a much simpler formula. The simpler formula can then be tried in an ad-hoc fashion, or used in turn in a regression approach. Although we have not yet taken this latter step of using regression, we believe that the present suggestion lends itself to such methods.

2.1 The basic model

The basic probabilistic model is the traditional relevance weight model [5], under which each term is given a weight as defined below, and the score (matching value) for each document is the sum of the weights of the matching terms:

$$w = \log \frac{(r+0.5)/(R-r+0.5)}{(n-r+0.5)/(N-n-R+r+0.5)} \qquad (1)$$

where

N is the number of indexed documents;
n the number of documents containing the term;
R the number of known relevant documents;
r the number of relevant documents containing the term.

This approximates to inverse collection frequency (ICF) when there is no relevance information. It will be referred to below (with or without relevance information) as $w^{(1)}$.

2.2 The 2-Poisson model and term frequency

One example of these problems concerns within-document term frequency (tf). This variable figures in a number of ad-hoc formulae, and it seems clear that it can contribute to better retrieval performance. However, there is no obvious reason why any particular function of tf should be used in retrieval. There is not much in the way of formal models which include a tf component; one which does is the 2–Poisson model [7, 8].

The 2–Poisson model postulates that the distribution of within-document frequencies of a content-bearing term is a mixture of two Poisson distributions: one set of documents (the "elite" set for the particular term, which may be interpreted to mean those documents which can be said to be "about" the concept represented by the term) will exhibit a Poisson distribution of a certain mean, while the remainder may also contain the term but much less frequently (a smaller Poisson mean). Some earlier work in this area [8] attempted to use an exact formula derived from the model, but had limited success, probably partly because of the problem of estimating the required quantities. The approach here is to use the behaviour of the exact formula to suggest a very much simpler function of tf which behaves in a similar way.

The exact formula, for an additive weight in the style of $w^{(1)}$, of a term t which occurs tf times, is

$$w = \log \frac{(p'\lambda^{tf}e^{-\lambda} + (1-p')\mu^{tf}e^{-\mu})(q'e^{-\lambda} + (1-q')e^{-\mu})}{(q'\lambda^{tf}e^{-\lambda} + (1-q')\mu^{tf}e^{-\mu})(p'e^{-\lambda} + (1-p')e^{-\mu})} \qquad (2)$$

where

λ is the Poisson mean for tf in the elite set for t;
μ is the Poisson mean for tf in the non-elite set;
p' is the probability of a document being elite for t given that it is relevant;
q' is the probability of a document being elite given that it is non-relevant.

As a function of tf, this can be shown to behave as follows: it is zero for $tf = 0$; it increases monotonically with tf, but at an ever-decreasing rate; it approaches an asymptotic maximum as tf gets large. The maximum is approximately the binary independence weight that would be assigned to an infallible indicator of eliteness.

A very simple formula which exhibits similar behaviour is $tf/(tf + constant)$. This has an asymptotic limit of unity, so must be multiplied by an appropriate binary independence weight. The regular binary independence weight for the presence/absence of the term may be used for this purpose. Thus the weight becomes

$$w = \frac{tf}{(k_1 + tf)} w^{(1)} \qquad (3)$$

where k_1 is an unknown constant.

Several points may be made concerning this argument. It is not by any stretch of the imagination a strong quantitative argument; one may have many reservations about the 2–Poisson model itself, and the transformations sketched above are hardly justifiable in any formal way. However, it results in a modification of the binary independence weight which is at least plausible, and has just slightly more justification than plausibility alone.

The constant k_1 in the formula is not in any way determined by the argument. The effect of choice of constant is to determine the strength of the relationship between weight and tf: a large constant will make for a relation close to proportionality (where tf is relatively

small); a small k_1 will mean that tf has relatively little effect on the weight (at least when $tf > 0$, i.e. when the term is present).

Our approach has been to try out various values of k_1 (around 1 may be about right for the full disks 1 and 2 database). However, in the longer term we hope to use regression methods to determine the constant. It is not, unfortunately, in a form directly susceptible to the methods of Fuhr or Cooper, but we hope to develop suitable methods.

2.3 Document length

The 2–Poisson model in effect assumes that documents (i.e. records) are all of equal length. Document length is a variable which figures in a number of weighting formulae.

We may postulate at least two reasons why documents might vary in length. Some documents may simply cover more material than others; an extreme version of this hypothesis would have a long document consisting of a number of unrelated short documents concatenated together (the "scope hypothesis"). An opposite view would have long documents like short documents, but longer: in other words, a long document covers a similar scope to a short document, but simply uses more words (the "verbosity hypothesis").

It seems likely that real document collections contain a mixture of these effects; individual long documents may be at either extreme or of some hybrid type. All the discussion below assumes the verbosity hypothesis; no progress has yet been made with models based on the scope hypothesis.

The simplest way to deal with this model is to take the formula above, but normalise tf for document length (dl). If we assume that the value of k_1 is appropriate to documents of average length ($avdl$), then this model can be expressed as

$$w = \frac{tf}{(\frac{k_1 \times dl}{avdl} + tf)} w^{(1)} \qquad (4)$$

A more detailed analysis of the effect on the Poisson model of the verbosity hypothesis is given in Appendix 7.4. This shows that the appropriate matching value for a document contains two components. The first component is a conventional sum of term weights, each term weight dependent on both tf and dl; the second is a correction factor dependent on the document length and the number of terms in the query (nq), though *not* on which terms match. A similar argument to the above for tf suggests the following simple formulation:

$$correction\ factor = k_2 \times nq \frac{(avdl - dl)}{(avdl + dl)} \qquad (5)$$

where k_2 is another unknown constant.

Again, k_2 is not specified by the model, and must (at present, at least) be discovered by trial and error. Values in the range 0.0–0.3 appear about right for the TREC databases (if natural logarithms are used in the term-weighting functions[1]), with the lower values being better for equation 4 termweights and the higher values for equation 3.

2.4 Query term frequency and query length

A similar approach may be taken to within-query term frequency. In this case we postulate an "elite" set of queries for a given term: the occurrence of a term in the query is taken as evidence for the eliteness of the query for that term. This would suggest a similar multiplier for the weight:

$$w = \frac{qtf}{(k_3 + qtf)} w^{(1)} \qquad (6)$$

In this case, experiments suggest a large value of k_3 to be effective—indeed the limiting case, which is equivalent to

$$w = qtf \times w^{(1)} \qquad (7)$$

appears to be the most effective.

We may combine a formula such as 6 or 7 with a document term frequency formula such as 3. In practice this seems to be a useful device, although the theory requires more work to validate it.

2.5 Adjacency

The recent success of weighting schemes involving a term-proximity component [9] has prompted consideration of including some such component in the Okapi weighting. Although this does not yet extend to a full Keen–type weighting, a method allowing for adjacency of some terms has been developed.

Weighting formulae such as $w^{(1)}$ can in principle be applied to any identifiable and searchable entity (such as, for example, a Boolean search expression). An obvious candidate for such a weight is any identifiable phrase. However, the problem lies in identifying suitable phrases. Generally such schemes have been applied only to predetermined phrases (e.g. those given in a dictionary and identified in the documents in the course of indexing). Keen's methods would suggest constructing phrases from all possible pairs (or perhaps larger sets) of query terms at search time; however, for queries of the sort of size found in TREC, that would probably generate far too many phrases.

The approach here has been to take pairs of terms which are adjacent in the query as candidate phrases.

[1] To obtain weights within a range suitable for storage as 16–bit integers, the Okapi system uses logarithms to base $2^{0.1}$

The present Okapi allows adjacency searches, so a phrase that is not specifically indexed can be searched, and assigned a weight in the usual Okapi fashion as if it had been indexed.

One problem with that approach is that the single words that make up the phrase will probably also be included in the query, and that suggests that a document which contains the phrase will be overweighted, as it will be given the weight assigned to the phrase in addition to the individual term weights. So in the present experiments the weight assigned to the phrase has been adjusted downwards, by deducting the weights of the constituent terms, to allow for the fact that the individual term weights have necessarily been added. Where this correction would give a negative weight to the phrase, it has been adjusted again to an arbitrary small positive number.

2.6 Weighting functions used

More than 20 combinations of the weighting functions discussed above were implemented at one time or another. Those mentioned in this paper are listed here. For brevity, most of the functions are referred to as BMnn (Best Match).

BM0: Flat, or quorum, weighting. Each term is given the same weight.

BM1: $w^{(1)}$ termweights.

BM15: 2–Poisson termweights as equation 3 with document length correction as equation 5.

BM11: 2–Poisson termweights with document length normalisation as equation 4[2].

3 Document processing

For TREC–1 City used an elaborate 25–field structure which was intended to make all the disparate datasets on the CDs fit a unified model. It would, for example, have been possible to restrict searches to "title", "headline" etc. In the event only the TEXT was used. For TREC–2, fields which looked useful for searching were simply concatenated into one long field. For most datasets fields other than DOCNO and TEXT were ignored, but the SJM LEAD PARAGRAPH, the Ziff SUMMARY and a few additional fields from the Patents records were included. This was done using a simple *perl* script (in contrast to the TREC–1 conversion program which used *lex*, *yacc* and *C*). Most of the known data errors were handled satisfactorily, although for some reason there still remained a few duplicate DOCNOs from disk 1 and/or 2.

[2] In theory there was also an equation 5 document length *correction*, but the best value of k_2 was found to be zero.

4 Automatic query processing

4.1 Ad-hoc

A large number of evaluation runs have been done to investigate

- the effect of query term source
- the use of a query term frequency (*qtf*) component in term weighting, and
- the use of algorithmically derived term pairs.

4.1.1 Derivation of queries from the topics

Topic processing was very simple. An program (written in *awk*) was used to isolate the required topic fields, which were then parsed and the resulting terms stemmed in accordance with the indexing procedures of the database to be searched. A small additional stop list was applied to the NARRATIVE and DESCRIPTION fields only. If required, the procedure also output pairs of adjacent terms which occur in the same subfield of the topic and with no intervening punctuation. For example the command

```
get_qterms 70 trec12_93 tcd pairs=1
```

applied to

```
<title> Topic: Surrogate Motherhood
<desc> Description:
Document will report judicial proceedings and
opinions on contracts for surrogate mother-
hood.
<con> Concept(s):
1. surrogate, mothers, motherhood
2. judge, lawyer, court, lawsuit, custody, hear-
ing, opinion, finding
(topic 70)
```

gave

```
70:19:desc:1:contract:1
70:19:con:1:court:1
70:19:con:1:custodi:1
70:19:con:1:find:1
70:19:con:1:hear:1
70:19:con:1:judg:1
70:19:desc:1:judici:1
70:19:con:1:lawsuit:1
70:19:con:1:lawyer:1
70:19:con:1:mother:1
70:19:tit:1:motherhood:3
70:19:con:1:opinion:2
70:19:desc:1:proceed:1
70:19:tit:1:surrog:3
70:19:desc:2:contract:surrog:1
```

70:19:desc:2:judici:proceed:1
70:19:desc:2:opinion:contract:1
70:19:desc:2:proceed:opinion:1
70:19:tit:2:surrog:motherhood:2

where the fields are *topic number*, *topic length* (number of terms counting repeats but not pairs), *source field* (in precedence order TITLE > CONCEPTS > NARRATIVE > DESCRIPTION > DEFINITIONS), *number of terms*, *term ...*, *frequency* of this term or pair in the topic.

4.1.2 Document and query term weighting

Table 1 shows the effect of varying query term source fields when no account is taken of within-query term frequency.

Some tentative conclusions can be drawn: adding TITLE to CONCEPTS improves most measures slightly; TITLE alone works well in a surprising proportion of the topics; the DESCRIPTION field is fairly harmless used in conjunction with CONCEPTS, but NARRATIVE and DEFINITIONS are detrimental. (TIME and NATIONALITY fields, which are occasionally present, were never used.) This really only confirms what may be evident to a human searcher: that CONCEPTS consists of search terms, but most of the other fields apart from TITLE are instructions and guidance to relevance assessors. A sentence such as "To be relevant, a document must identify the case, state the issues which are or were being decided and report at least one ethical or legal question which arises from the case." (from the NARRATIVE field of topic 70) can only contribute noise.

However, when a within-query term frequency (*qtf*) component is used in the term weighting, the information about the relative importance of terms gained from the use of all or most of the topic fields seems to outweigh the detrimental effect of noisy terms such as "identify", "state", "issues", "question". Some results are summarised in Table 2. A number of values of k_3 were tried in equation 6, and a large value proved best overall, giving the limiting case (equation 7), in which the term weight is simply multiplied by *qtf*.

Many combinations of the weighting functions discussed in Section 1.1, as well as others not described here, were first tested on the AP and/or WSJ databases. Some of them were eliminated immediately. The function defined as BM15 gave almost uniformly better results than $w^{(1)}$, after suitable values for the constants had been found. BM11 appeared slightly less good than BM15 on the small databases, but later runs on the large databases showed that, with suitable choice of constants, it was substantially, though not uniformly, better. This may be a consequence of the greater variation in document lengths found in the large databases. Table 3 compares the more elaborate term weighting functions with the standard $w^{(1)}$ weighting and with a baseline coordination level run.

Some work was done on the addition of adjacent pairs of topic terms to the queries (see Section 2.5). A number of runs were done, using several different ways of adjusting the "natural" weights of adjacent pairs. There was little difference between them, and the results are at best only slightly better than those from single terms alone (Table 3). There was also little difference between using all adjacent pairs and using only those pairs which derive from the same sentence of the topic, with no intervening punctuation.

4.2 Routing

Potential query terms were obtained by "indexing" all the known relevant documents from disks 1 and 2; the topics themselves were not used (nor were known non-relevant documents). These terms were then given $w^{(1)}$ weights and *selection values* [11] given by $\frac{r}{R} \times w^{(1)}$ where r and R are as in equation 1.

A large number of retrospective test runs were performed on the complete disks 1 and 2 database, in which the number of terms selected and the weighting function were the independent variables. Overall, there was little difference in the average precision over the range 10–25 terms. This is consistent with the results reported by Harman in [10]. With regard to weighting functions, BM1 was slightly better than BM15. However, looking at individual queries, the optimal number of terms varied between three (several topics) and 31 (topic 89) with a median of 11; and BM15 was better than BM1 for 27 of the topics.

Two sets of official queries and results were produced. For the **cityr1** run, the top 20 terms were selected for each topic and the weighting function was BM1. For **cityr2** the test runs were sorted for each topic by precision at 30 documents within recall within average precision, and the "best" combination of number of terms and weighting function was chosen. When evaluated retrospectively against the full disks 1 and 2 database the **cityr2** queries were about 17% better on average precision and 10% better on recall than the **cityr1**. The official results (first and second rows of Table 4) show a similar difference. Later, both sets of queries were repeated using BM11 instead of the previous weighting functions (third and fourth rows of the table). These final runs both show substantially better results than either of the official runs.

Table 1: Effect of varying query term sources (no query term frequency component)

Query source	ave lgth	Ave Prec	Prec at 5	Prec at 30	Prec at 100	R-Prec	Recall	% of tops where AveP $\geq$ median
TC	30.3	0.300	0.624	0.536	0.440	0.349	0.683	66
C	26.7	0.296	0.636	0.524	0.436	0.346	0.686	58
TCD	39.7	0.297	0.592	0.519	0.429	0.340	0.667	62
TCND	81.0	0.263	0.612	0.485	0.394	0.306	0.605	48
TCN	71.6	0.262	0.624	0.481	0.397	0.309	0.604	50
TCNDDef	86.3	0.257	0.580	0.468	0.387	0.303	0.604	46
TN	44.9	0.181	0.500	0.418	0.320	0.245	0.491	26
TND	54.4	0.179	0.492	0.403	0.317	0.243	0.491	24
TD	13.1	0.170	0.428	0.381	0.297	0.244	0.492	28
T	3.6	0.165	0.380	0.343	0.271	0.233	0.471	32
Terms: single. Document termweights: BM11. Database: disks 1 and 2. Topics 101–150								
Query average length is the average number of terms taking account of repeats								

Table 2: Effect of varying query term sources (with query term frequency component)

Query source	Weight function	AveP	P5	P30	P100	RP	Rcl	% of tops where AveP $\geq$ median
TCND	BM11	0.360	0.652	0.569	0.479	0.401	0.754	92
TCN	BM11	0.356	0.644	0.565	0.482	0.399	0.749	92
TCNDDef	BM11	0.354	0.648	0.559	0.474	0.395	0.751	92
TCD	BM11	0.353	0.644	0.565	0.481	0.394	0.750	90
TC	BM11	0.335	0.636	0.560	0.468	0.375	0.723	86
TC	BM15	0.284	0.560	0.485	0.416	0.336	0.685	56
TND	BM11	0.283	0.556	0.503	0.414	0.338	0.652	60
TN	BM11	0.274	0.556	0.497	0.399	0.331	0.643	56
TC	BM1	0.232	0.504	0.435	0.361	0.289	0.601	28
Document term weights were multiplied by qtf, equivalent to large k_3 in eqn 6								
Terms: single. Database: disks 1 and 2. Topics 101–150								

Table 3: Effect of different document term weighting functions: single terms and adjacent pairs

Weight function	Terms	AveP	P5	P30	P100	RP	Rcl	% of tops where AveP $\geq$ median
BM11	singles + "natural" pairs	0.307	0.628	0.541	0.448	0.358	0.696	62
BM11	singles + all adj pairs	0.304	0.612	0.544	0.447	0.357	0.694	62
BM11	singles	0.300	0.624	0.536	0.440	0.349	0.683	66
BM15	singles	0.227	0.500	0.434	0.351	0.285	0.595	38
BM1	singles	0.199	0.468	0.416	0.326	0.261	0.542	22
BM0	singles	0.142	0.412	0.336	0.270	0.209	0.411	12
"Natural" means adjacent in the same sentence of the topic with no intervening punctuation								
Query term source: TC. qtf component: none. Database: disks 1 and 2. Topics: 101–150								

Table 4: Some routing results

Weight function	Number of terms	AveP	P5	P30	P100	RP	Rcl	% of tops where AveP $\geq$ median
BM1/BM15	variable	0.356	0.692	0.561	0.449	0.388	0.680	78
BM1	top 20	0.315	0.628	0.533	0.432	0.361	0.648	70
BM11	variable	0.394	0.700	0.599	0.481	0.429	0.713	92
BM11	top 20	0.362	0.684	0.605	0.459	0.397	0.707	80
Best predictive run for comparison (BM11, *qtf* with large k_3, source TCD)								
		0.300	0.612	0.524	0.394	0.345	0.632	68
Database: disk 3. Topics: 51–100								

5 Manual queries with feedback

5.1 The user interface

The interface allowed the entry of any number of *find* commands operating on "natural language" search terms. By default, the system would combine the resulting sets using the BM15 function described in Section 2.6, but any operation specified by the searcher would override this. All user-entered terms were added to a pool of terms for potential use in query expansion. Every set produced had any documents previously seen by the user removed from it.

The *show* (document display) command displayed the full text of a single document (or as much as the user wished to see) with the retrieval terms highlighted (sometimes inaccurately). Unless specified by the user this would be the highest-weighted remaining document from the most recent set. At the end of a document display the relevance question

"Is this relevant (y/n/?)"

appeared; the system counted documents eliciting the "?" response as relevant[3]. The DOCNO was then output to a results file, together with the iteration number.

Once some documents had been judged relevant the *extract* command would produce a list of terms drawn from the pool consisting of user-entered terms and terms extracted from all relevant documents. Terms in the pool were given $w^{(1)}$ weights. User-entered terms were weighted as if they had occurred in four out of five fictitious relevant documents (in addition to any real relevant documents they might have been present in). Thus for user-entered terms the numerator in equation 1 becomes $(r + 4 + 0.5)/(R + 5 - r - 4 + 0.5)$ [2].

Query expansion terms were selected from the term pool in descending order of the selection value [11] $termweight \times (r + 4)/(R + 5)$ for user-entered terms, otherwise $termweight \times r/R$, subject to not all documents containing the term having been displayed, and the term not being a semi-stopword[4] (unless it was entered by the user). A maximum of 20 terms was used. These selected terms were then used automatically in an expansion search, again with the BM15 weighting function.

Each invocation of *extract* used all the available relevance information, and there was no "new search" command. This was intended to encourage compliance with the TREC guidelines; it was not possible for a dissatisfied user to restart a search. When the searcher decided to finish, after some sequence of *find*, *show* and *extract* commands, the *results* command invoked a final iteration of *extract* (provided there had been at least three positive relevance judgments). Finally, the top 1000 DOCNOs from the current set were output to the results file. Apart from the aforementioned commands, users could do *info sets* and *history*.

5.2 Searchers and search procedure

The searches were done by a panel of five staff and research students from City University's Department of Information Science. Search procedure was not rigidly prescribed, although some guidelines were given. There was a short briefing session and searchers were encouraged to experiment with the system before starting. Procedures seemed to be considerably influenced by individual preferences and styles. Some searches were done collaboratively.

Searchers tried to find relevant documents by any means they liked within a single session. The number of iterations of query expansion varied between zero and four, with a mean of two. The IDs of all documents looked at were output to the results file, together with the iteration number. At the end of the session, if at least three relevant documents had been found the system did a final iteration of query expansion and output

[3] It was possible for searchers to change their minds about the relevance of a document. Subsequent feedback iterations handled this correctly, but the DOCNO would be duplicated in the search output. This appears to have led to some minor errors in the frozen ranks evaluation in a few topics.

[4] Semi-stopwords are words which, while they may be useful search terms if entered by a user, are likely to be detrimental if used in query expansion: numerals, month-names, common adverbs etc.

the top 1000 IDs; if less than three the top 1000 from the set which was finally "current" were output.

There seemed to be an impression that the new topics (topics3) are more difficult than the old. Results may also have been affected by the huge stoplist which was being used at that time because of a breakdown of the only disk large enough to hold the very large scratch files generated during inversion. Lack of the number "6" affected one topic, days of the week another ("Black Monday"). The searcher was urged to leave "Black Monday" to the end in case we were able to reindex before the deadline, but she decided to try it and thought it worked quite well.

An edited transcript of one searcher's notes is given below as Appendix B.

5.3 Results

The official results of the manual run (Table 5) are disappointing, with average precision 0.232 (60% of topics below median), precision at 100 docs 0.4 and recall 0.59. The final iteration was later re-run with BM11 instead of BM15, and the results combined with the feedback documents from the original searches for a frozen ranks evaluation[5]. This did somewhat better on a majority of the topics, but overall the manual results were very poor compared to some of the automatic runs.

6 Other experiments

6.1 Query modification without relevance information

Some iterative automatic ad hoc runs were done in which the top 10–50 documents obtained by the best existing method were used (a) as a source of additional terms and (b) as a source of "relevance" information for the $w^{(1)}$ weight calculation.

Expansion terms were selected as described in Section 4.2, in descending order of $\frac{r}{R} \times w^{(1)}$. The maximum number of additional terms was set at half the number of query terms. For many of the topics most of the top terms extracted from the feedback documents were in any case topic terms, so the number of additional terms was small.

Example (topic 112)

Topic 112: Funding biotechnology
30 feedback documents used
In the table which follows, term sources are given either as *doc*, in the case of expansion terms, or as a topic field, where *tit* > *con* > *nar* > *desc*. In this example, final weights involve a *qtf* component, and were obtained using equation 6 with $k_3 = 8$ (the resulting weight was multiplied by k_3 to obtain adequate granularity in an integer representation). For expansion terms, *qtf* was taken as 1 and the same correction applied.

					Weights	
Term	Src	*qtf*	# docs	Orig	$w^{(1)}$	Final
biotechnologi	tit	9	30	765	145	614
invest	con	4	29	148	80	213
fund	tit	2	23	78	55	88
capit	nar	2	21	78	51	81
pharmaceut	doc	(0)	15	-	73	64
ventur	nar	1	21	55	67	59
financi...	nar	2	17	64	36	57
startup...	nar	1	11	70	62	55
research	nar	1	26	35	61	54
financ	doc	(0)	15	-	54	48
partner	doc	(0)	17	-	55	48
drug	doc	(0)	18	-	53	47
investor	doc	(0)	19	-	52	46
provid	nar	3	14	66	21	45
firm	nar	1	22	36	50	44
technologi	doc	(0)	23	-	50	44
company...	doc	(0)	28	-	48	42
academ	nar	1	4	73	48	42
corpor	nar	2	9	76	26	41
monei	desc	1	18	37	43	38
stock	nar	1	20	33	43	38
industri...	doc	(0)	23	-	42	37
develop	doc	(0)	25	-	42	37
laboratori	nar	1	9	51	39	34
quantifi	nar	1	1	82	39	34
profit	nar	1	14	40	38	33
enterpr	nar	1	4	59	33	29
establish	nar	1	10	38	29	25
arena*	nar	2	0	148	15	24
data	nar	4	6	108	8	21
sale	nar	1	12	30	24	21
loss	nar	1	7	39	22	19
government...	nar	1	13	24	20	17
assist	nar	1	6	39	20	17
much	desc	1	11	28	20	17
answer	desc	1	2	52	16	14
follow	nar	1	7	26	9	8
rel*	desc	1	1	52	9	8
eg*	nar	1	0	67	8	7
question	desc	1	3	37	8	7
worldwid*	nar	2	0	126	4	6
division*	nar	1	2	41	6	5
figur*	nar	1	2	41	5	4

Here, nine of the 43 terms[6] are not from the topic. The starred terms were not used in the final search because their selection value $w^{(1)} \times \frac{r}{R}$ is zero (to the nearest integer). For this topic, the additional terms were beneficial and reweighting alone rather neutral.

[5] There were two topics where the searcher found no relevant documents, so for these topics the original results were inserted.

[6] The terms followed by ellipses represent synonym classes.

Terms	Wts	AveP	P5	P30	P100	RP	Rcl
All	Final	0.407	1.000	0.867	0.640	0.457	0.739
Topic	Final	0.362	1.000	0.733	0.620	0.440	0.698
Topic	Orig	0.373	0.600	0.800	0.700	0.433	0.680

Discussion

The main motive for experimenting with this type of query expansion is that it is one way of finding terms which are in some sense closely associated with the query as a whole. It does not fit particularly well with the Robertson/Sparck Jones type of probabilistic theory [5], the validity of which depends on pairwise independence of terms in both relevant and nonrelevant documents. However, it is clear, if only from the results in this paper, that mutual dependence does not necessarily lead to poor results.

There are many variables involved. In our rather limited experiments most of the initial feedback searches were done under the conditions of the first row of Table 2, that is with terms from title, concepts, narrative and description (there were a few runs using title and concepts only, but the results for most topics were not good); and weighting function BM11 with termweights given by equation 6 with large k_3 (1000). This gave nearly the best precision at 5 and 30 documents of any of our results. The number of feedback documents was constant across topics and was varied between 10 and 50. For the final search, terms were always weighted with BM11, but several values of k_3 were tried (including zero). Some runs used topic terms only and some used expansion terms as well. There was one run omitting narrative and description terms from the final search, but it was not among the very best and is not reported in the table. The number of terms in the final search was varied from 10 upwards, terms being selected as usual in descending order of $termweight \times \frac{r}{R}$. Some evaluations were done using frozen ranks, in case the initial searches tended to give better low precision, but this turned out not to be the case.

A few of the results are summarised in Table 6. They include results which appear better than the best otherwise obtained, but the difference is small, and these runs have not yet been repeated on the other topic sets. A *qtf* weight component is still needed (compare rows 2 and 14 of the table). The number of feedback documents is not critical. Speeding searching by using only the top 10 or 20 terms is detrimental.

It is interesting that results do not seem to be very greatly affected by the precision of the feedback set. Looking at the individual topics in the run represented by the top row of Table 6, 25 did better than in the feedback run, 18 did worse and the remainder about the same. Restricting to the 20 topics where the precision at 30 in the feedback set was below 0.5, the corresponding figures are 7, 10 and 3.

6.2 Stemming

A comparison was made on the AP database between the normal Okapi stemming which removes many suffixes and a "weak" stemming procedure which only conflates singular and plural forms and removes "ing" endings. For some weighting functions weak stemming increased precision by about 2% and decreased recall by about 1%, but the observed difference is unlikely to be significant.

6.3 Stoplists

Some runs were done on the AP database to investigate the effect of stoplist size. A small stoplist consisted of the 17 words

> **a, the, an, at, by, into, on, for, from, to, with, of, and, or, in, not, et**

and a large one contained 209 articles, conjunctions, prepositions, pronouns and verbs.

There was no significant difference in the results of the runs, but the index size was about 25% greater with the small stoplist.

7 Conclusions and prospects

7.1 The new probabilistic models

The most significant result is perhaps the great improvement in the automatic results brought about by the new term weighting models. In the ad-hoc runs, with no *qtf* component, BM15 is 14% better than BM1 on average precision and about 9% better on high precision and recall. The corresponding figures for BM11 are 51% and 34% (Table 3). For the routing runs, where a considerable amount of relevance information had contributed to the term weights, the improvement is less, but still very significant (Table 4). For the manual feedback searches (Table 5) there was a small improvement when they were re-run with BM11 replacing BM15 in the final iteration.

The drawback of these two models is that the theory says nothing about the estimation of the constants, or rather parameters, k_1 and k_2. It may be assumed that these depend on the database, and probably also on the nature of the queries and on the amount of relevance information available. We do not know how sensitive they are to any of these factors. Estimation cannot be done without sets of queries and relevance judgments, and even then, since the models are not linear, they do not lend themselves to estimation by logistic regression. The values we used were arrived at by long sequences of trials mainly using topics 51–100 on the disks 1 and 2 database, with the TREC–1 relevance sets.

Taking advantage of the very full topic statements to derive query term frequency weights gives another substantial improvement in the automatic ad-hoc results. Comparing the top row of Table 2 with the top row of Table 1, there is a 20% increase in average precision. The "noise" effect of the narrative and description fields is far more than outweighed by the information they give about the relative importance of terms (compare the "TCND" row of Table 1 with the top row of Table 2).

It remains to be discovered how well these new models perform in searching other types of database. Term frequency and document length components may not be very useful in searching brief records with controlled indexing, but one would expect these models to do well on abstracts. It is also rare to have query statements which are as full as the TIPSTER ones, so there are many situations in which a *qtf* component would have little or no effect.

7.2 Routing

Our results here (Table 4) were relatively good, and further improved when re-run with BM11. However, the TREC routing scenario is perhaps not particularly realistic, given the large amount of relevance information, which we made full use of as the sole source of query terms. In addition, the best of our runs depended on a long series of retrospective trials in which the number of query terms was varied. In a real-world situation one would have to cope with the early stages when there would be few documents and little relevance information (initially none at all). It would be necessary to develop a term selection and weighting procedure which was capable of progressing smoothly from a minimum of prior information up to a TREC-type situation. It may be possible to come up with a decision procedure for term selection using something similar to the selection value $w^{(1)} \times \frac{r}{R}$. Perhaps a future TREC could include some more restrictive routing emulations.

7.3 Interactive ad-hoc searching

The result of this trial was disappointing except on precision at 100 documents (Table 5), scarcely better than the official automatic ad-hoc run. On three topics it gave the best result of any of our runs, and two more were good, but the remaining 45 ranged from poor to abysmal. Little analysis has yet been done. For some topics it is clear that the search never got off the ground because the searcher was unable to find enough relevant documents to provide reliable feedback information, but the mean number found per topic was ten, which should have been enough to give reasonable results (cf Table 6, where ten feedback documents performs quite well). Currently, there are discussions towards a more realistic set of rules for interactive searching for TREC–3, and we hope to develop a better procedure and interface.

7.4 Prospects

Paragraphs

When searching full text collections one often does not want to search, or even necessarily to retrieve, complete documents. Our new probabilistic models do not apply to documents where the verbosity hypothesis does not apply (Section 2.3). Some of the TREC–2 participants searched "paragraphs" rather than documents, and this is clearly right, provided a sensible division procedure can be achieved. We made some progress towards developing a "paragraph" database model for the Okapi system, but there has not been time to implement it. Further work then needs to be done on methods of deriving the retrieval value of a document from the retrieval value of its constituent paragraphs.

Parameter estimation

Work is in progress on methods of using logistic regression or similar techniques to estimate the parameters for the new models.

Derivation and use of phrases and term proximity

A few results are reported in Table 3. They are not particularly encouraging. There is probably scope for further experiments in this area, not only on tuples of adjacent words but also on Keen-type [9] weighting of query term clusters in retrieved documents.

References

[1] D.K. Harman (Ed.), *The First Text REtrieval Conference (TREC–1).* Gaithersburg, MD: NIST, 1993.

[2] Robertson S.E. *et al.* Okapi at TREC. In: [1] (pp.21–30).

[3] Walker, S. and Hancock-Beaulieu, M. *Okapi at City: an evaluation facility for interactive IR.* London: British Library, 1991. (British Library Research Report 6056.)

[4] Hancock-Beaulieu, M.M. and Walker, S. An evaluation of automatic query expansion in an online library catalogue. *Journal of Documentation, 48,* Dec. 1992, 406–421.

[5] Robertson, S.E. and Sparck Jones, K. Relevance weighting of search terms. *Journal of the American Society for Information Science, 27,* 1976, 129–146.

[6] Cooper, W. *et al.* Probabilistic retrieval in the TIPSTER collection: an application of staged logistic regression. In: [1] (pp.73-88).

[7] Harter, S.P. A probabilistic approach to automatic keyword indexing. *Journal of the American Society for Information Science, 26*, 197–206 and 280–289.

[8] Robertson, S.E, Van Rijsbergen, C.J. & Porter, M.F. Probabilistic models of indexing and searching. In Oddy, R.N. *et al.* (Eds.), *Information Retrieval Research* (pp.35–56). London: Butterworths, 1981.

[9] Keen, E.M. The use of term position devices in ranked output experiments. *Journal of Documentation, 47*, 1991, 1–22.

[10] Harman, D. Relevance feedback revisited. In: *SIGIR 92. Proceedings of the 15th International Conference on Research and Development in Information Retrieval* (pp.280–289). ACM Press, 1992.

[11] Robertson, S.E. On Term Selection for Query Expansion. *Journal of Documentation, 46*, 1990, 359–364.

A 2-Poisson model with document length component

Basic ideas

The basic weighting function used is that developed in [8], and may be expressed as follows:

$$w(\underline{x}) = \log \frac{P(\underline{x}|R)P(\underline{0}|\overline{R})}{P(\underline{x}|\overline{R})P(\underline{0}|R)} \qquad (8)$$

where

$\underline{x}$ is a vector of information about the document;
$\underline{0}$ is a reference vector representing a zero-weighted document;
R and $\overline{R}$ are relevance and non-relevance respectively.

For example, each component of $\underline{x}$ may represent the presence/absence of a query term in the document (or, as in the case of formula 2 in the main text, its document frequency); $\underline{0}$ would then be the "natural" zero vector representing all query terms absent. In this formulation, independence assumptions lead to the decomposition of w into additive components such as individual term weights.

A document length may be added as a component of $\underline{x}$; however, document length does not so obviously have a "natural" zero (an actual document of zero length is a pathological case). Instead, we may use the average length of a document for reference; thus we would expect to get a formula in which the document length component disappears for a document of average length, but not for other lengths.

Suppose, then, that the average length of a document is Δ. The weighting formula becomes:

$$w(\underline{x}, d) = \log \frac{P((\underline{x}, d)|R)P((\underline{0}, \Delta)|\overline{R})}{P((\underline{x}, d)|\overline{R})P((\underline{0}, \Delta)|R)}$$

where d is document length, and $\underline{x}$ represents all other information about the document. This may be decomposed as follows:

$$w(\underline{x}, d) = w(\underline{x}, d)_1 \; + \; w(\underline{x}, d)_2 \qquad (9)$$

where

$$w(\underline{x}, d)_1 = \log \frac{P(\underline{x}|(R,d))P(\underline{0}|(\overline{R},d))}{P(\underline{x}|(\overline{R},d))P(\underline{0}|(R,d))}$$

and

$$w(\underline{x}, d)_2 = \log \frac{P((\underline{0},d)|R)P((\underline{0},\Delta)|\overline{R})}{P((\underline{0},d)|\overline{R})P((\underline{0},\Delta)|R)}$$

These two components are discussed further below.

Hypotheses

As indicated in the main text, one may imagine different reasons why documents should vary in length. The two hypotheses given there ("scope" and "verbosity" hypotheses) may be regarded as opposite poles of explanation. The arguments below are based on the Verbosity hypothesis only.

The Verbosity hypothesis would imply that document properties such as relevance and eliteness can be regarded as independent of document length; given eliteness for a term, however, the number of occurrences of that term would depend on document length. In particular, if we assume that the two Poisson parameters for a given term, λ and μ, are appropriate for documents of average length, then the number of occurrences of the term in documents of length d will be 2-Poisson with means $\lambda d/\Delta$ and $\mu d/\Delta$.

Second component

The second component of equation 9 is

$$w(\underline{x}, d)_2 = \log \frac{P(\underline{0}|(R,d))P(\underline{0}|(\overline{R},\Delta))}{P(\underline{0}|(\overline{R},d))P(\underline{0}|(R,\Delta))} \; + \; \log \frac{P(d|R)P(\Delta|\overline{R})}{P(d|\overline{R})P(\Delta|R)}.$$

Under the Verbosity hypothesis, the second part of this formula is zero. Making the usual term-independence assumptions, the first part may be decomposed into a sum of components for each query term, thus:

$$w(t,d)_2 = \log \frac{(p'e^{-\lambda d/\Delta} + (1-p')e^{-\mu d/\Delta})(q'e^{-\lambda} + (1-q')e^{-\mu})}{(q'e^{-\lambda d/\Delta} + (1-q')e^{-\mu d/\Delta})(p'e^{-\lambda} + (1-p')e^{-\mu})} \qquad (10)$$

where t is a query term and p', q', λ and μ are as in formula 2. Note that there is a component for each query term, whether or not the term is in the document.

For almost all normal query terms (i.e. for any terms that are not actually detrimental to the query), we can assume that $p' > q'$ and $\lambda > \mu$. In this case, formula 10 can be shown to be monotonic decreasing with d, from a maximum as $d \to 0$, through zero when $d = \Delta$, and to a minimum as $d \to \infty$. As indicated, there is one such factor for each of the nq query terms.

Once again, we can devise a very much simpler function which approximates to this behaviour; this is the justification for formula 5 in the main text.

First component

Expanding the first component of 9 on the basis of term independence assumptions, and also making the assumption that eliteness is independent of document length (on the basis of the Verbosity hypothesis), we can obtain a formula for the weight of a term t which occurs tf times. This formula is similar to equation 2 in the main text, except that λ and μ are replaced by $\lambda d/\Delta$ and $\mu d/\Delta$. The factors d/Δ in components such as λ^{tf} cancel out, leaving only the factors of the form $e^{-\lambda d/\Delta}$.

Analysis of the behaviour of this function with varying tf and d is a little complex. The simple function used for the experiments (formula 4) exhibits some of the correct properties, but not all. In particular, the maximum value obtained as $d \rightarrow 0$ should be strongly dependent on tf; formula 4 does not have this property.

B Extracts from a searcher's notes

Choice of search terms

Suitable words and phrases occurring in title, description, narrative, concept and definition fields were underlined—often this provided more than enough material to begin with. Sometimes they were supplemented by extra words, e.g. for a query on international terrorism I added "negotiate", "hostage", "hijack", "sabotage", "violence", "propaganda", as well as the names of known terrorist groups likely to fit the US bias of the exercise.

I did not look at reference books or other on-line databases, and tended to avoid very specific terms like proper names from the query descriptions, as I found they could lead the search astray. For instance, the 1986 Immigration Law was also known as the Simpson-Mazzoli Act, but the name Mazzoli also turned up in accounts of other pieces of legislation, so it was better to use a combination of "real" words about this topic.

In some queries, it was necessary to translate an abstract concept, e.g. "actual or alleged private sector economic consequences of international terrorism" into words which might actually occur in documents, e.g. "damage", "insurance claims", "bankruptcy", etc. For this purpose the use of a general (rather than domain-specific) thesaurus might be a useful adjunct to the system.

Like the other participants I was surprised at the contents of the stop-word list, e.g. "talks", "recent", "people", "new", but not "these"! However it was usually possible to find synonyms for stop-words and their absence was not seriously detrimental to any query.

Grouping of terms, use of operators

Given the complexity of the queries, it was obviously necessary to build them up from smaller units. My original intention was to identify individual facets and create sets of single words representing each, then put them together to form the whole query. [...] For example, for a query about the prevention of nuclear proliferation I had a set of "nuclear" words (reprocessing, plutonium, etc.), a set of "control" words (control, monitor, safeguards, etc.) and sets of words for countries (argentina, brazil, iraq, etc.) suspected of violating international regulations on this point. This proved a bad strategy—the large sets (whether ORed or BMed[7] together) had low weightings because of their collectively high frequencies, and the final query was very diffuse.

A more successful approach was to build several small, high-weighted sets using phrases with OP=ADJ or OP=SAMES[entence] (e.g. economic trends, gross national product, standard of living, growth rate, productivity gains), and then to BM them together, perhaps with a few extra singletons (e.g. decline, slump, recession). Because of the TREC guidelines, I didn't look at any documents for the small sets as I went along, although under normal circumstances I would have done so.

Our initial instructions were to use default best-matching if at all possible, rather than explicit operators. As already suggested, ADJ and SAMES were an absolute necessity given the length of documents to be searched, but AND and OR were generally avoided—on the occasions when I tried AND (out of desperation) it was not particularly useful. For one query where I thought it might be necessary (to restrict a search to documents about the US economy) it luckily proved superfluous because of the biased nature of the database, indeed it would have made the results worse as the US context of these documents was implied rather than stated.

Viewing results, relevance feedback

Normally I looked at about the top 5–10 records from the first full query. If 40% or more seemed relevant, the query was considered to be fairly satisfactory and I went on down the list trying to accumulate a dozen or so records for the extraction phase. As ... noted by other participants, there was a conflict between judging a record relevant because it fitted the query, and because it was likely to yield useful new terms for the next phase. On the one hand were the "newsbyte" type of documents containing one clearly relevant paragraph amidst a great deal of potential noise, and on the other the documents which were in the right area, contained all the right words, but failed the more abstract exclusion conditions of the query. I tried to judge on query relevance, but erred on the side of permissiveness for documents containing the right sort of terms.

The competition conditions discouraged a really thorough exploration of possibilities when a query was not initially successful. In one very bad case, having seen more than 20 irrelevant records and knowing that they would appear at the head of my output list, I felt that the query would show up badly in the [results] anyway and that it was not worth exploring further, as I might had there been a real question to answer.

[7] BM = "best match"; the default weighted set combination operation was BM15 (see Section 2.6)

Extracting new terms

I tried to get at least six relevant documents for the extraction phase, and usually managed a few more. As already noted, sets generated by term extraction contain only single words, so before looking at the new records I sometimes added in a few phrases to this set, either important ones from the original query or others which had occurred in relevant documents. The extracted sets of terms tended to be larger than the original query and certainly included items which a human searcher (at least one unfamiliar with this genre of literature) would not have thought of. It was amusing, for instance, to see "topdrawer" and "topnotch" (epithets for companies) extracted from documents about investment in biotechnology, and "leftist" (an invariable collocate for Sandanista) pulled out of documents about Nicaraguan peace talks. Some material for socio-linguistic analysis here!

My impression ... is that where the original document set from which terms were extracted was fairly coherent, the derived set [from query expansion] also had a high proportion of relevant documents. Not surprisingly, where I had scraped the barrel and tried several different routes to a few relevant documents, extraction produced equally miscellaneous and disappointing results.

Normally I went through two or three cycles of selection/extraction, but looking at fewer records each time. The set of extracted terms did not seem to change materially from one cycle to the next, and I would have expected the final result file reflected the query quite well even though the phrases had been lost.

Conclusion

In spite of the frustrations of this exercise, I found it a more interesting retrieval task than normal bibliographic searching, mainly because it was possible to see the full documents to gauge the success of the query, and use a broader range of natural-language skills to dream up potentially useful search terms.

Acknowledgments

We are most grateful to the British Library Research & Development Department and to DARPA/NIST for their financial support of this work. Our advisers have been unstintingly helpful. We blame the system, not our panel of searchers, for the poor results of the interactive trial. Above all, we wish to thank Donna Harman of NIST for her outstandingly efficient and courteous organisation and management of the TREC projects.

Table 5: Manual searches with feedback

Run	AveP	P5	P30	P100	RP	Rcl	% of tops where AveP $\geq$ median
Official (BM15)	0.232	0.492	0.468	0.400	0.297	0.591	40
Re-run (BM11)	0.247	0.480	0.477	0.411	0.315	0.607	48
Database: disks 1 and 2. Topics: 101–150							

Table 6: Some results from query modification

# fbk docs	Term source	k_3	Terms	AveP	P5	P30	P100	RP	Rcl	% of tops where AveP $\geq$ median
50	TCNDdoc	8	all	0.369	0.660	0.591	0.487	0.408	0.754	92
30	TCNDdoc	8	all	0.368	0.668	0.585	0.486	0.407	0.749	88
10	TCNDdoc	8	all	0.363	0.668	0.584	0.485	0.400	0.748	88
50	TCNDdoc	9	all	0.360	0.624	0.573	0.482	0.399	0.748	88
50	TCNDdoc	9	top 40	0.354	0.620	0.573	0.478	0.394	0.741	86
50	TCNDdoc	9	top 30	0.353	0.632	0.567	0.480	0.395	0.742	88
30	TCNDdoc	8	top 30	0.360	0.676	0.577	0.479	0.402	0.741	88
30	TCNDdoc	8	top 20	0.348	0.636	0.571	0.474	0.392	0.734	82
30	TCNDdoc	8	top 10	0.318	0.604	0.537	0.449	0.366	0.702	78
50	TCND	8	all	0.364	0.636	0.573	0.487	0.406	0.755	92
30	TCND	8	all	0.362	0.644	0.573	0.484	0.408	0.749	90
10	TCND	8	all	0.363	0.640	0.574	0.481	0.406	0.754	88
30	TCND	0	all	0.334	0.652	0.559	0.458	0.374	0.711	80
30	TCNDdoc	0	all	0.310	0.645	0.546	0.448	0.359	0.675	66
Initial feedback run for comparison (top row of Table 2)										
None	TCND	large	all	0.360	0.652	0.569	0.479	0.401	0.754	92
Retrospective run using all known relevant documents to reweight the topic terms										
Variable	TCND	0	all	0.371	0.708	0.600	0.497	0.408	0.758	92
Database: disks 1 and 2. Topics: 101–150										

Combining Evidence for Information Retrieval

N.J. Belkin, P. Kantor, C. Cool, R. Quatrain
School of Communication, Information & Library Studies
Rutgers University
New Brunswick, NJ 08903 USA
[belkin/kantorp/ccool/quatrain]@cs.rutgers.edu

Abstract

This study investigated the effect on retrieval performance of two methods of combination of multiple representations of TREC topics. Five separate Boolean queries for each of the 50 TREC routing topics and 25 of the TREC ad hoc topics were generated by 75 experienced online searchers. Using the INQUERY retrieval system, these queries were both combined into single queries, and used to produce five separate retrieval results, for each topic. In the former case, results indicate that progressive combination of queries leads to progressively improving retrieval performance, significantly better than that of single queries, and at least as good as the best individual single query formulations. In the latter case, data fusion of the ranked lists also led to performance better than that of any single list.

1. Introduction

The general goal of our project in the TREC-2 program was to investigate the effect of making use of several different formulations of a single information problem, on information retrieval (IR) system performance. The basis for this work lies in both theory and empirical evidence. From the empirical point of view, it has been noted for some time, that different representations of the same information problem retrieve sets (or ranked lists) of documents which contain different relevant, as well as non-relevant documents (see, e.g. McGill, Koll & Norreault, 1979; Saracevic & Kantor, 1988). There is some implication from this evidence (made explicit by Saracevic and Kantor, 1988), that taking account of the different results of the different formulations, could lead to retrieval performance that is better than that of any of the individual query formulations. From the theoretical point of view, IR can be considered as a problem of inference (see, e.g. van Rijsbergen, 1986). That is, IR is concerned with estimating, given available evidence about such things as information problems and documents (or in general, retrievable information objects), the likelihood (or probability, or degree) of relevance of a document to the information problem. From this point of view, different query formulations constitute different sources of evidence which could be used to infer the probable relevance of a document to an information problem, and it is thus reasonable to consider ways in which to use (i.e. combine) these sources of evidence in the inference process.

These ideas are general to any source of evidence which might be used for IR, such as the evidence of different retrieval techniques, or different document representation techniques, or, in general, different IR systems. One aspect of our project uses the example of different query formulations as a simulation of the general problem of combination of evidence from different systems.

An additional argument is available for the special case of different query representations. That is, if we consider an information problem to be a complex, and in general difficult-to-specify entity (see, e.g. Taylor, 1968; Belkin, Oddy & Brooks, 1982), then we might conclude that each different representation, derived from some statement by the user, is a different *interpretation* of the user's underlying information problem, highly unlikely to be like anyone else's (or any other system's) interpretation. Given the empirical evidence, whether any one such interpretation is 'better' than another seems moot. However, we might say that each captures some different, yet pertinent aspect of the user's underlying problem; or, that those aspects of the different interpretations which are common to them all (or more than one) reflect some 'core' aspect of the problem. Although techniques for making use of the different interpretations might vary according to which of these two views one takes, the general position suggests that it will always be a good idea to take advantage of as many such interpretations as possible. For this case, we therefore consider the issue of combination of different query representations within the 'same' IR system.

Our project, thus, considers the problem of inference in IR at two levels of analysis. The first level, as introduced by Turtle & Croft (1991), asks about the effect of evidence obtained when two or more formal query statements are produced for the same information problem. The second level, which is simulated in this study, asks about combination of evidence provided by two or more distinct systems, ranking the same set of documents in response to the same problem. To distinguish those two levels, and in keeping with earlier discussions of the issues involved, we henceforth refer to the combination of query statements as "query combination", and we refer to the combination of evidence from differing systems as "data fusion". Others have also addressed various aspects of this general question. Apart from those already cited, we mention in particular the work of Fox and his colleagues (Fox et al., 1993; Fox and Shaw, this volume), and that of Belkin, et al. (1993). These studies in fact address precisely the question of query combination, the Belkin et al. work being a direct precursor to this, and the Fox et al. studies using different query formulation, combination and retrieval techniques, but with very similar results.

Why ought either of these two methods work in the IR situation? The central idea is that either the specific internal score, assigned to a document for a query, or the rank of

a document in the list produced for a query, represents information about the relevance of the document to the query. For Boolean retrieval, we may address this question with concepts of signal detection. In this framework, there are two conditional probabilities. The probability that a relevant document is retrieved by system S is d_S. The probability that a not relevant document is retrieved is f_S. If two systems (or formulations) are independent, the posterior relevance odds are increased by the product d_1d_2/f_1f_2. In actual application (Saracevic and Kantor, 1988), improvements are not this large, suggesting either the existence of an effective base of not-relevant documents, or some effect of interdependence. It can be shown that if several query formulations are drawn from a normal distribution centered at the optimal query formulation, then some fraction of the time, the simple average of these formulations will be closer to the optimum than even the best of them. An even larger fraction of the time, there will be an optimum linear combination which is more nearly optimal than any of the cases from which it is formed (Kantor, 1993).

The existence of such models explains why we might expect combination of evidence, or data fusion, to work for the case of several query formulations, as, for instance, in the INQUERY retrieval system (Turtle & Croft, 1991). But these models do not predict that these techniques must work. The investigation of whether they do work, is the subject of this paper.

Specifically, we investigate whether data fusion methods will produce better performance than any single method; and, whether combination of query formulations does better than the best individual query formulations, and whether progressive combination of query formulations leads to progressively better IR performance. For each of these questions, we also address the issue of what methods to use in the combination of evidence.

In this paper, we do not discuss the "official" results which we submitted to TREC-2, except in passing. The reason for this is that we are not so much interested in the absolute performance of the techniques which we use, as in their performance relative to one another. What we are most concerned with is what happens to retrieval performance as we combine evidence; if we find that combining evidence in specific ways leads to improvements over our starting point of non-combination, then we can begin to investigate how to optimize starting points, as well as rules for combination.

The general plan of our study was as follows. We collected, from experienced online searchers, five different query formulations for each of the 50 routing topics and for 25 of the ad hoc topics. These query formulations were then put to the INQUERY retrieval system (made available to us by the University of Massachusetts), both as single queries, and as combinations of queries for each topic. The combinations were studied at various levels, with the five-fold combination for each set being reported as "official" TREC-2 results for query combination. The five retrieved lists for the ad hoc topics were merged, and reported as "official" TREC-2 results for data fusion.

2. Methods

2.1 Query Formulation Procedures

The query formulations used in this study were generated by volunteer online searchers, all of whom were experienced users of large bibliographic retrieval systems. In order to obtain the multiple query representations, we asked five different searchers to generate Boolean search statements for each of the TREC topics in our analysis. We asked each of our volunteer searchers to generate a query formulation for five different topics, resulting in five independently generated query formulations for each topic. After formulating each query, searchers were asked to answer four questions about the process: how long it took to formulate the query; how related the topic was to their normal searches; how easy it was for them to formulate the query; and, the extent to which they had enough information to construct the query. A total of 75 searchers participated in our study; 50 for the routing topics, and 25 for the ad hoc topics. In addition to the questionnaire items mentioned above, the ad hoc searchers were also asked how many years of online searching experience they had. Searchers for the routing queries were not asked this question. See the Appendix for a sample response sheet.

Our study is based on analysis of the entire set of 50 routing topics, and a selected sample of 25 ad hoc topics. The sample was stratified according to the domain of the topic, in an effort to represent the distribution of domains in the entire set of ad hoc topics.

In our experiments, we used the INQUERY retrieval engine (version 1.5), developed at the University of Massachusetts (Turtle & Croft, 1991). INQUERY is a probabilistic inference network-based system, which is based upon the idea of combining multiple sources of evidence in order to plausibly infer the relevance of a document to a query. The underlying formalism is that of a Bayesian probabilistic inference network (Pearl, 1988), which provides strict rules for how to combine sources of evidence. Turtle and Croft (1991) give a detailed description of the model and its implementation; a more general description is available in Belkin and Croft (1992). Here, we note a few characteristics of the system which are germane to the project at hand.

First, INQUERY provides a natural means for combination of multiple query formulations, as a function of its design. Second, it incorporates a large set of operators which allow, in addition to sophisticated natural language query formulations, complex Boolean formulations. The Boolean operators in INQUERY are not strict, however, which allows ranking of output, and also leads to significantly better performance than strict Boolean retrieval (Turtle and Croft, 1991). See the paper by Croft in this volume for more detail on INQUERY.

2.2 Query Combination Experiments

Each of the Boolean query formulations produced by our searchers was translated into INQUERY syntax. Two methods of query combination were then used in our study, each specific to the TREC-2 tasks of responding to ad hoc

and routing topics. The first, which we label "comb1" was applied to the ad hoc topics. In this procedure, we simply combine the five query formulations for each topic directly, into one query, using the INQUERY "unweighted sum" operator. This query is then used as the search statement in our experiments. In the ad hoc search environment, we cannot expect to have relevance judgments, and so we can do no more than simple combination.

The second combinatorial procedure, called "combx", was used for the routing topics. Here, we did a separate search for each separate query formulation for all 50 topics, on the training set supplied from the TREC-1 data. From these results, we used the average 11-point precision (in the "official" results reported at TREC-2; precision at 100 documents for the "unofficial results" reported in this paper) of each query formulation as a weight for that formulation in the combination of all five formulations for each topic. For this, we used INQUERY's "weighted sum" operator. This procedure corresponds to constructing a simple combined query, learning something about how that query's components perform on the current database, and taking account of that evidence to modify the query formulation for searching the next database.

These methods of combining queries give us a very straightforward way to test our hypotheses about the effectiveness of multiple sources of evidence. For our experiments (as opposed to the results which were submitted to TREC-2, which were just the comb1 and combx results as described above), we divided the query formulations for both ad hoc and routing topics, into five different groups. In each group, each topic was represented by one query, and no searcher was represented more than once in any one group. This distribution was meant to control for possible searcher effects. We then did runs for each single group, and for each combination of groups, for both ad hoc and routing topics. With these data, we were able to compare retrieval performance of different levels of query combination, and to compare retrieval performance of combined queries with uncombined.

2.3 Data Fusion Experiments

Data fusion was accomplished by a list-merging method which is the natural extension of a 3-out-of-5 data fusion logic in the binary case. The basic data used was the five lists of documents retrieved by the five different query formulations for each topic. Every document has some rank in each of the five lists being joined together. An effective rank is calculated by taking the third highest of the five ranks which the document has. This has the same effect as moving a threshold along the list of effective ranks, and including a document in the output when it has appeared on three of the lists. Since there are five scores all together, this can also be thought of as a median rule.

In practice, to maintain consistency with other parts of our work, we did not calculate the rank of every document, but worked with the lists of the top 1000 documents produced in response to each query formulation. This meant that some documents would appear on all five of the lists, others on just four, or three, or even fewer. Of course, the whole logic of data fusion suggests that those which appear on more lists are more likely to be relevant. We implemented this, in fact, by forming a combined sort key consisting of (10-degeneracy, 3-rd rank). The degeneracy is the number of lists on which a specific document appears in the top 1000. We used a lexicographic sort, so that all items with degeneracy 5 appeared before any items with degeneracy 4, and so on. Within a given degeneracy, items with lower values for the 3rd rank were ranked first.

3. Results

3.1 Caveats

The results presented in this paper differ in several ways from those submitted as "official" results to TREC-2, which are published at the end of this volume. According to our experimental design, there are five independently produced query formulations for each of the TREC topics. However, due to uneven return rate among our searchers, we were missing one searcher's set of queries for the ad hoc topics, and three searchers' sets of queries for the routing topics, when we did the "official" runs. Consequently, in the official results, five ad hoc topics and fifteen routing topics are represented by four searches, rather than five. However, we were subsequently able to obtain substitute searchers, and so for the "unofficial" results presented and discussed in this paper, we have the full complement of 75 searchers and five query formulations per topic.

We were unable to report the data fusion results for routing topics for the official results, because of time constraints. We have subsequently been able to do those runs, and report them here as unofficial results.

We also caution that one query for one of our ad hoc topics is known to have a syntactic error which resulted in very poor performance for that single query, and for all unweighted combinations of queries in which it was present. Therefore, some of our comparative results in the ad hoc case may be slightly incorrect.

3.2 General Results

Because our analyses of ad hoc topics are based on a subset of the total sample, we here consider questions of the sample representativeness. As explained above, the sample was originally chosen to represent topic domains. To see if this had introduced some other bias, we compared the distribution of our 25 topics along the three dimensions of topics proposed by Harman (this volume). These are: broadness, operationally defined as the total number of relevant documents found for that topic; hardness, operationally defined as inverse to the median average precision for that topic; and, restriction, defined according to linguistic characteristics of the topic. The distribution of the 25 topics in our sample did not differ significantly from the total ad hoc topic distribution on any of these dimensions, so we feel reasonably confident that we did not select a markedly biased subset of topics.

Tables 1 and 2 present a descriptive profile of the queries and topics in our study, based upon the query formu-

lations, and the searchers' responses to our questionnaire. Table 1 shows the distribution of numbers of words and operators per query, and also of time required to construct a query. Table 2 shows the distribution of searchers' attitudes to the topics, each indicated on a scale of one to five, from least to most.

	Mean	Std Dev.	Min.	Max.	N
Operators	9.94	5.72	1.00	44.00	375
Words	19.40	14.63	1.0	145.00	375
Time (minutes)	11.31	7.48	1.00	40.00	367

Table 1. Characteristics of queries for ad hoc and routing topics.

	Mean	Std Dev.	Min.	Max.	N
Familiarity	1.81	1.15	1.00	5.00	388
Ease of construction	2.82	1.11	1.00	5.00	372
Enough information	3.20	1.11	1.00	5.00	322

Table 2. Characterization of topics by searchers, for routing and ad hoc topics.

Our ad hoc questionnaire also included a question on how many years of experience each searcher had in online searching. The mean response was 6.8 years. Unfortunately, we do not have these data for the routing searchers.

We wished to consider whether there were any relationships between the various characteristics of queries and topics and the performance of the queries themselves. For this purpose, we constructed a table in which each separate query formulation (75x5=375) is associated with performance measures, the characteristics enumerated in tables 1 and 2, and the three topic categories of broadness, hardness and restriction defined by Harman (this volume). For performance, we considered using one or more of three measures: average of 11-point precision; precision at 100 documents; and R-precision (defined by Harman, this volume). Factor analysis of these three measures showed that a single factor accounts for more than 90% of the variance among them, so that they represent, in effect, a single aspect or factor of performance. The average precision was chosen as representative of this factor, and we have used it both in evaluation of our retrieval results, and in attempting to determine the effect of the other variables we have considered, on retrieval performance. Since this variate does not exhibit a normal distribution, logarithmic and logistic transforms were explored. The logistic leads to a most nearly normal distribution of the transformed score, but we can still not say that the transformed variable follows a normal distribution.

The results of applying ANOVA to seek a predictor of p are shown in Table 3. No significant relations appear. Because of the range of values assumed by the variables Operators, Words and Time, the relation was sought using regression analysis. Once again, no significant relations were found, and the scatter plots (not included here) make it clear that there is no trend to be found.. Both hardness and broadness are significantly related to performance. The former is expected, since the hardness is determined by median average precision; the latter is less obvious.

Analysis of variance for log(p/(1-p))	
Independent variable	Significance
Familiarity	0.149
Easiness	0.169
Information	0.907

Table 3. Significance levels of F-tests using ANOVA to seek dependence of the logistically transformed average precision on the searcher's assessments of their query formulation.

The search for relations between average precision and characteristics of the query formulation, whether provided by the search, or determined from the query text itself, was motivated by the results, discussed below, which show that it is desirable to weight formulations in proportion to their average precision. Thus, if we could find a surrogate for average precision which can be known without evaluating the retrieved documents, it would be possible to approximate the effective combination on the first pass of a retrieval operation. This hope is frustrated at this time.

3.3 Query Combination and Data Fusion Results: Ad hoc Topics

The official results reported to TREC-2 were for the overall performance of each of two treatments for the ad hoc topics, and of one treatment for the routing topics. For those results, we refer the reader to the relevant section of this volume. Here we report on our further investigations on the effect of combination of queries, and of data fusion, on performance.

Our first investigation in query combination was to see if combining query formulations has a regular, beneficial effect, as hypothesized. To do this, we generated the five different search groups for the ad hoc topics, as described in section 2.2, and did experimental runs on all single query groups, all 2-way combinations of queries, all 3-way combinations of queries, all 4-way combinations of queries, and the combination of all 5 query formulations. The results are presented in Table 4, where it is evident that the average performance increases monotonically as more evidence is added. The increase is strict and significant, as shown in Table 4a, where we display the number of times that each combination level performed better than each other level. We note that the data fusion results are not significantly better than any but 1-way combination (that is, average performance for single queries), but also that its performance is not significantly different from unweighted 5-way combination.

1-way	2-way	3-way	4-way	5-way	fusion
0.1441	0.2016	0.2235	0.2361	0.2349	0.2042
0.1571	0.1823	0.2304	0.2225		
0.1121	0.2051	0.2102	0.2292		
0.1589	0.1951	0.2043	0.2200		
0.1378	0.1763	0.2079	0.2166		
	0.2113	0.2171			
	0.1727	0.2172			
	0.1683	0.1873			
	0.1633	0.2116			
	0.1885	0.1934			
0.1420	0.1864	0.2103	0.2249	0.2349	0.2042

For example, column two represents the ten possible ways of choosing two groups of query formulations from the collection of five groups. Each entry is an average over 25 topics.

Table 4. For ad hoc topics, average 11-point precision, by group, for each combination of queries, and mean average precision for all groups at each level of combination.

	1-way	2-way	3-way	4-way	5-way	fusion
1-way		1**	3**	3**	3**	5**
2-way	24**		5**	6**	5**	9
3-way	22**	20**		6**	3.5**	11
4-way	22**	19**	19**		3**	12
5-way	22**	20**	21.5**	22**		15
fusion	20**	16	14	13	10	

**= significant difference at $p \leq .01$, sign test
*= significant difference at $p \leq .05$, sign test
Read row with respect to column, e.g. 2-way performed better than 1-way 24 out of 25 times, or 1-way performed better than 2-way 1 out of 25 times

Table 4a. Number of times, for average performance of combinations for ad hoc topics, that one treatment performed better than another.

The results presented in Tables 4 and 4a are based on the average performance for the query formulations in any one set. In Tables 5 and 5a, we present data on performance, for ad hoc topics, when only the best query formulation, or best combination of query formulations, for each topic is used. These results are compared with the single 5-way combination (which is the only combination possible at this level with our data), and with the fusion results. It is of some interest to note that the ranking of level of combination is now very much different than that for average performance, with 2-way and 3-way combination being significantly better than 1-way, 4-way, 5-way and fusion (see Table 5a).

1-way	2-way	3-way	4-way	5-way	fusion
0.2712	0.3002	0.2959	0.2702	0.2350	0.2042

Table 5. For ad hoc topics, mean 11-point precision for best-performing combination of queries for each topic.

	1-way	2-way	3-way	4-way	5-way	fusion
1-way		6**	7*	13	17	21**
2-way	19**		16.5	20**	22**	24**
3-way	18*	8.5		20.5**	22**	22**
4-way	12	5**	4.5**		22.5**	20**
5-way	8	3**	3**	2.5**		15
fusion	4**	1**	3**	5**	10	

**= significant difference at $p \leq .01$, sign test
* = significant difference at $p \leq .05$, sign test
Read row with respect to column, e.g. 2-way performed better than 1-way 19 out of 25 times, or 1-way performed better than 2-way 6 out of 25 times

Table 5a. Number of times, for performance of best combinations for ad hoc topics, that one treatment performed better than another.

3.4 Adaptive Combination: Ad hoc Topics

Finally, to get an overall idea of how query combination in the ad hoc case worked, and to estimate whether taking account of the evidence of search performance could improve subsequent performance, we compared performance of simple combination of all five query formulations (comb1) with performance when only the best single query formulation for each topic was used (best), with combination of all five query formulations weighted according to the precision at 100 documents retrieved, of each formulation (comby). The results, reported in Tables 6 and 6a, show that there is no significant difference between comb1 and best, but that comby is significantly better than comb1. While formation of comby would not be possible under the conditions of the ad hoc TREC task, these results are of interest because they simulate the kind of operations that could be implemented in a fully interactive interface to an IR system.

comb1	best	comby	fusion
0.2350	0.2712	0.2819	0.2042

comb1 = unweighted combination of all queries for each topic
best = best performing query for each topic
comby = weighted (by prec.@100 docs) combination of all queries for each topic

Table 6. For ad hoc topics, mean 11-point precision for four treatments.

In reading Tables 6 and 6a, note that the choice referred to as "best" corresponds exactly to the choice called "1-way" in Table 5. However, it does not correspond to any of the entries in the first column of Table 4. The entries in Table

4 refer to combinations based upon the fixed groups. But, the combination of groups which performs best for, say, Topic 57, need not be the one which performs best for Topic 72. In Tables 6 and 6a, the best possible combination is chosen for each topic individually. Note also that, in the INQUERY system, the "unweighted sum" corresponds to a symmetrical assignment of each weight to all formulations.

	comb1	best	comby	fusion
comb1		8	4**	15
best	17		9	21**
comby	21**	16		20**
fusion	10	4**	5**	

** = significant difference at $p \leq .01$, sign test
* = significant difference at $p \leq .05$, sign test
Read row with respect to column, e.g. comby performed better than comb1 21 times , or comb1 performed better than comby 4 times.

Table 6a. Number of times that one treatment for ad hoc topics performed better than another.

3.5 Query Combination and Data Fusion Results: Routing Topics

We ran further experiments on the routing queries, analogous to those we used for the ad hoc queries. Our first set of results shows the progressive effect of unweighted combination of query formulations, by level of combination, when average performance at each level is considered (tables 7 and 7a). Again, as for the ad hoc queries (tables 4 and 4a), there is a progressive, significant effect of level of query combination. For the routing queries, data fusion appears to have a somewhat stronger effect than for ad hoc, being significantly better than 1-, 2- and 3-way combination. It is of some interest to note that the overall level of performance for routing topics is much higher than for the ad hoc topics.

1-way	2-way	3-way	4-way	5-way	fusion
0.1763	0.2311	0.2599	0.2619	0.2807	
0.1890	0.2202	0.2503	0.2748		
0.1684	0.2258	0.2603	0.2735		
0.2025	0.2229	0.2314	0.2512		
0.1793	0.2436	0.2415	0.2745		
	0.2364	0.2471			
	0.2388	0.2509			
	0.2160	0.2654			
	0.2149	0.2642			
	0.2338	0.2417			
0.1831	0.2283	0.2513	0.2672	0.2807	0.2661

Each entry is an average over 50 topics.

Table 7. For routing topics, average 11-point precision, by group, for each combination of queries, and mean average precision for all groups at each level of combination.

	1-way	2-way	3-way	4-way	5-way	fusion
1-way		3**	2**	1**	2**	8**
2-way	47**		5.5**	6**	5.5**	13**
3-way	48**	44.5**		9**	7**	18*
4-way	49**	44**	41**		8**	22.5
5-way	48**	44.5**	43**	42**		28
fusion	42**	37**	32*	27.5	22	

** = significant difference at $p \leq .01$, sign test
* = significant difference at $p \leq .05$, sign test
Read row with respect to column, e.g. 2-way performed better than 1-way 47 out of 50 times, or 1-way performed better than 2-way 3 out of 50 times

Table 7a. Number of times, for average performance of combinations for routing topics, that one treatment performed better than another.

As for the ad hoc topics, we then compared the results of the best query formulation combinations for each level of combination, with the unweighted 5-way combination, and fusion results. As for the ad hoc queries, this gave us quite a different ranking of levels of combination, with 3-way and 2-way combinations being significantly better than all others, and 4-way being significantly better than 5-way and fusion (tables 8 and 8a).

1-way	2-way	3-way	4-way	5-way	fusion
0.2931	0.3173	0.3199	0.3069	0.2807	0.2661

Table 8. For routing topics, mean 11-point precision for best-performing combination of queries for each topic.

	1-way	2-way	3-way	4-way	5-way	fusion
1-way		8.5**	13.5**	22	29	36**
2-way	41.5**		20.5	34*	38**	39**
3-way	36.5**	29.5		37**	42**	45**
4-way	28	16*	13**		44**	40**
5-way	21	12**	8**	6**		28
fusion	14**	11**	5**	10**	22	

** = significant difference at $p \leq .01$, sign test
* = significant difference at $p \leq .05$, sign test
Read row with respect to column, e.g. 2-way performed better than 1-way 41.5 times, or 1-way performed better than 2-way 8.5 times

Table 8a. Number of times, for performance of best combinations for routing topics, that one treatment performed better than another.

3.6 Adaptive Combination: Routing Topics

Finally, we wished to investigate the effectiveness of progressively taking account of retrieval performance in

modification of the query formulation. To do this, we compared performance of unweighted 5-way query combination (comb1) with performance using the best- performing query formulations in the training database (best1), the best performing query formulations in the test database (best2), the weighted 5-way query combination using weights from the training database (combx), the weighted 5-way query combination using weights from the test database (comby), and 5-way query combination weighted by the mean of the weights for test and training databases. The weights that we used were the precision at 100 retrieved documents for each query formulation. In the official results, we used average 11-point precision. The reason for the change, is that precision at some cutoff level is a realistic measure for the routing task in general, and especially in an operational environment, whereas the average precision is a measure that we cannot realistically expect to have in an operational environment. When we compared the performance of both weights in the combx formulation, there was no significant difference. The results are presented in tables 9 and 9a, and show that taking account of subsequent evidence has a positive and significant effect on performance. When reading Tables 9 and 9a, note that the entries for comb1 and fusion have already appeared in Table 7, as "5-way" and "fusion", respectively. Also, "best2" has already appeared in Table 8, as the best "1-way" combination.

comb1	best1	best2	combx	comby	comxy	fusion
.2807	.2721	.2931	.3012	.3090	.3068	.2661

comb1 = unweighted combination of all queries for each topic
best1 = best performing query (on training set) for each topic
best2 = best performing query (on test set) for each topic
combx = weighted (by prec.@100 docs in training set) combination of all queries for each topic
comby = weighted (by prec.@100 docs in test set) combination of all queries for each topic
combxy = weighted (by mean of the sum of prec.@100 docs in training and test sets) combination of all queries for each topic

Table 9. For routing topics, mean 11-point precision for seven treatments.

Table 9a encapsulates all of the key concepts of the several approaches to combination that we have explored. We have two approaches which are *a priori* and symmetric in their treatment of the query formulations (fus and comb1). As expected, the fusion system, using the least information, performs worse. comb1, the symmetric formulations does better, although the difference is not statistically significant. Both of these methods often perform better than the best of the individual formulations, and their relations to other combination schemes are (except for the relation to best2) quite similar. The query that performs best on the training set (best1) does not perform significantly better than any of the combination schemes. But that formulation which performs best on the test set (best2, also called 1-way in Table 8) is significantly better than best1 and the fusion scheme.

	com1	best1	best2	comx	comy	cxy	fus
com1		29	21	13.5 **	16*	14.5 **	28
best1	21		13**	14**	14**	12.5 **	22.5
best2	29	37**		23	20	23	36**
comx	36.5 **	36**	27		21.5	18**	40**
comy	34*	36**	30	28.5		25.5	36.5 **
cxy	35.5 **	37.5 **	27	32*	24.5		37**
fus	22	27.5	14**	10**	13.5 **	13**	

** = significant difference at $p \leq .01$, sign test
* = significant difference at $p \leq .05$, sign test
Read row with respect to column, e.g. combx performed better than comb1 36.5 times, or comb1 better than combx 13.5 times.

Table 9a. Number of times that one treatment for routing topics performed better than another.

Of greater interest are the methods representing adaptive weighting schemes: combx, comby and combxy. Most significantly, combx, the adaptive weighting formulation, is better than the symmetrically weighted combination (comb1), the fusion rule, and the best single formulation in a substantial fraction (over 70%) of all cases. The weighting based on the test set (comby) stands also in essentially the same relation to those three other schemes. Finally, the weighting scheme combxy simulates a situation which might arise in updating or tuning a combination rule after two batches of documents have been retrieved. This is accomplished by averaging the weights assigned to each formulation in the training run, with those assigned based on the test run. This scheme shows essentially the same profile as combx and comby when compared with the comb1, fusion, best1 and best2 schemes. It performs significantly better than combx, but not significantly better than comby.

4. Discussion

4.1 General Results

As is customary, we begin this section with a general disclaimer. In this case, we need to point out that all of our results were obtained with a very specific kind of query formulation technique and very special kinds of queries, and, that all of our results were obtained within a very special retrieval context, the INQUERY system. It is certainly possible that these circumstances strongly affected our results, so that we cannot make widely general claims for them. On the other hand, the results reported by Fox and Shaw (this volume), using queries generated in quite different ways, and using a quite different IR system and retrieval technique, are quite similar in general form and trend to

ours, although their specific figures are different. So we are willing to believe that the influence of our experimental situation is probably not enough to invalidate our results at some level of generality.

There are several aspects of our general results which are of some interest, apart from the issues of query combination and data fusion. One has to do with the lack of any significant relationship between number of words in a query, and the performance of a query. It has been at least informally suggested in the IR community, that the retrieval performance of queries increases with the number of words in the query. There is no support in our data for this hypothesis. Indeed, in our data, there is at least one one-word query, which performed better than all of the other, multi-word queries for that topic.

It is also of interest that none of the query/searcher characteristics was related to performance. This may be a characteristic of our particular data set, but it also suggests that it will be rather difficult to identify characteristics of people or topics at this level, which will be predictive of performance of the query.

Although the level of familiarity by the searchers on the topics was in general rather low, our searchers nevertheless found it not too difficult to formulate queries (mean of 2.82 on a scale of 1 to 5), and felt that they had sufficient information to construct a reasonable query, on the basis of the topic (mean of 3.2 on a scale 1 of 5). This makes us think that the queries are likely to be reasonable formulations of the search topics, at least as far as the searchers are concerned. But the range and variability of the numbers of words and numbers of operators per topic seems to indicate that the query formulations themselves are rather different (we have not yet compared them for overlap in specific words, but work on this issue is in progress). These two results seem to us to confirm our initial idea that each query formulation is indeed a "different" interpretation of the information problem, and thus to substantiate our general approach.

4.2 Query Combination Results

Our results, for both ad hoc and routing topics, seem clearly to show that, in general, the more evidence one has, and uses, in the form of different query formulations, the better the IR performance is going to be. In particular, tables 4, 6, 7 and 9 support this conclusion, in various respects. From the results of tables 6 and 9, we can see that, taking advantage of what one learns about query performance from one iteration doesn't help a lot, after the first iteration, but on the other hand, it doesn't hurt, either. This suggests to us that continual modification and reweighting of the multiple query formulations in a combined query, is likely to be useful in the general routing environment. But even doing it once, given the initial evidence, seems to help. This also suggests that continuing to add new query formulations to a combined query will likely help performance on subsequent runs.

Having said all this, it is worth considering the results of tables 5 and 8, which showed that picking the best 2-way or 3-way combination of query formulations was significantly better than using 4-way or 5-way combinations. On the face of it, this runs counter to the general result of "the more, the better". However, it is possible that this result is an artifact of our data. For both 2-way and 3-way combinations, it was possible to choose the best from ten different combinations. Because we had only five different query formulations for each topic, we had smaller pools from which to choose, for both single query formulations, and for the 4-way and 5-way combinations. This issue needs further investigation.

4.3 Data Fusion Results

There are several points to be made with regard to the median-fusion scheme as implemented here. First, as may be expected from general arguments, it sometimes performs better than the best of the lists which are joined by the fusion process. Second, it does not perform as well as even the symmetric (unweighted) combinations made using the internal scores generated by the INQUERY system. This is expected, since those scores contain more information than the rankings alone. One can imagine special cases in which the distribution of scores assigned to a document by several queries is such that the internal combination rule of unweighted sum does not perform as well, but this has apparently not occurred in the cases studied here.

Third, in the application to the routing problem we have, in fact, operated in a batch mode. For a true routing situation, it would be necessary to estimate cutoff scores for the several query formulations, corresponding to the cutoff rank on the fused list. For large data sets this can be done easily. Without this step, it is not possible to make an immediate decision about a newly presented document.

Fourth, the stability induced by using this system was manifested in the case of the one query for which we discovered, too late to make the change, that one of the query formulations was in error. For this case, all of the "average combination of evidence formulations" performed more poorly than the fusion rule. This is because one, or even two disastrously bad query formulations will have little effect on the results of the 3-of-5 fusion rule. Of course, expect for the case of combining all five query formulations, the best of one, two, three or four query formulations can do well because the one bad formulation will be missing from the combinations that are best.

Finally, the application of data fusion here, at the so-called decision level (that is to say, after the documents have been ranked according to several rules) is a simulation for the case to which it should be applied. Since the specific system that we used permits internal manipulation of scores, there is no need to delay combination until after the output lists have been formed. But in realistic settings, several distinct systems will have internal operations which are not compatible, so that, even if it were possible to extract the internal scores, it would not be apparent how to combine them.

5. Conclusions

In general, we conclude that our initial research questions with respect to query combination have been positively answered. That is, if one has available several different representations of a single information problem, then it makes sense to use all of them, in combination, in order to improve retrieval performance, rather than to try to identify and use only the best one. In addition, it is reasonably clear that progressive and continuous combination of query formulations leads to continuing and progressive improvement of performance. This may extend to progressive modification of query formulations in the routing situation, for instance, on the basis of each iteration of retrieval. Nevertheless, some of our results appear anomalous, and in particular we need to address more carefully the issue of how best to combine query formulations.

As far as our data fusion questions are concerned, we have clearly demonstrated that doing data fusion is better than using only one query formulation. Although performance improvement in these experiments was rather low, for operational settings in which there are multiple systems with incompatible scores, a data fusion method that works with the ranked outputs, rather than the scores is the precise method that is needed. In the present study we have shown how that method can be extended from the case of binary (set) retrieval to the case of ranked lists. We have shown that the results are, on the average, better than the results of the individual formulations. In some cases, they are better than the best of the component formulations. This lends support to a program of seeking optimal tunings for fusion of any number of given systems, to achieve results better than any of them alone could provide.

Overall, we find strong support for adaptive weighting in query combination. This is applicable to both routing, as shown directly here, and to relevance feedback, which we have simulated in our application to the ad hoc topics. We also find strong support for enlarging the set of query representations. This success raises many interesting possibilities. For example, one might systematically explore the k-way combinations to see how they compare to the adaptive weighting scheme. Or, one might apply the notion of adaptive weighting to the best of the k-way combinations. The possibilities for combining these two concepts explodes (of course) combinatorially. We feel that the present experiments point a way into the forest of possibilities.

6. Acknowledgments

We wish to thank the 75 searchers who so generously donated their time and effort to this project. Without them this research could not have been done. We also wish to thank Audrey Gorman and Kathy Mrowka, who provided invaluable assistance on the project in planning, data gathering and input, and Dong Li, who helped with the data analysis. We owe special thanks to Bruce Croft and Jamie Callan, not only for permission to use the INQUERY system for this investigation, but also for the unstinting support they gave us in using it. This research was performed with partial funding from a TREC support grant from the ARPA.

7. References

BELKIN, N.J., COOL, C., CROFT, W.B. & CALLAN, J.P. (1993). The effect of multiple query representations on information retrieval performance. In: *Proceedings of the 16th International Conference on Research and Development in Information Retrieval (SIGIR '93)*, Pittsburgh, 1993. New York, ACM: 339-346.

BELKIN, N.J. & CROFT, W.B. (1992) Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35,12: 29-38.

BELKIN, N.J, ODDY, R.N. & BROOKS, H.M. (1982) ASK for information retrieval. *Journal of Documentation*, 38, 2&3: 61-71, 145-164.

FOX, E.A. et al. (1993) Combining evidence from multiple searches. In: D.K Harman, ed., *The First Text REtrieval Conference (TREC-1)*. GPO, Washington, D.C.: 319-328.

KANTOR, P. (1993) Vector space models of data combination in information retrieval. Technical Report APlab/TR-93-3. New Brunswick, NJ., Rutgers University, School of Communication, Information & Library Studies.

MCGILL, M., KOLL, M. & NORREAULT, T. (1979) An evaluation of factors affecting document ranking by information retrieval systems. Syracuse, Syracuse University School of Information Studies.

SARACEVIC, T. & KANTOR, P. (1988) A study of information seeking and retrieving. III. Searchers, searches, overlap. *Journal of the ASIS*, 39,3: 197-216.

TAYLOR, R.S. (1968) Question negotiation and information seeking in libraries. *College and Research Libraries*, 29: 178-194.

TURTLE, H. & CROFT, W.B. (1991) Evaluation of an inference network-based retrieval model. *A C M Transactions on Information Systems*, 9,3: 187-222.

VAN RIJSBERGEN, C.J. (1986) A new theoretical framework of information retrieval. In: *Proceedings of the 1986 International Conference on Research and Development in Information Retrieval (SIGIR '86)*, Pisa, 1986. New York, ACM: 194-200.

APPENDIX: SEARCHER RESPONSE SHEET

QUERY FORMULATION

TOPIC NUMBER:_____ PACKET NUMBER:_____

Please formulate a search query for one of the five topics you have been sent, in the space below. Don't forget to indicate the topic number in the space above. Please read the entire topic description before you begin your query forumlation. (Use the back of this sheet, if there isn't enough room below)

Please answer the following questions, as they relate to this specific query formulation.

1. About how many minutes did it take (including reading the topic description)? _____ minutes

2. Is this topic related to things you normally search on (please circle one number)?

1----------2----------3----------4----------5
Not at all Somewhat Very much

3. How easy was it to formulate this query?

1----------2----------3----------4----------5
Not easy Somewhat Very easy

4. Do you feel you had enough information to construct an effective query?

1----------2----------3----------4----------5
Too little Adequate Plenty

5. About how many years have you been doing online searching? _____ years.

Automatic Routing and Ad-hoc Retrieval Using SMART : TREC 2

Chris Buckley*, James Allan, and Gerard Salton

Abstract

The Smart information retrieval project emphasizes completely automatic approaches to the understanding and retrieval of large quantities of text. We continue our work in the TREC 2 environment, performing both routing and ad-hoc experiments. The ad-hoc work extends our investigations into combining global similarities, giving an overall indication of how a document matches a query, with local similarities identifying a smaller part of the document which matches the query. The performance of the ad-hoc runs is good, but it is clear we are not yet taking full advantage of the available local information.

Our routing experiments use conventional relevance feedback approaches to routing, but with a much greater degree of query expansion than was done in TREC 1. The length of a query vector is increased by a factor of 5 to 10 by adding terms found in previously seen relevant documents. This approach improves effectiveness by 30–40% over the original query.

Introduction

For over 30 years, the Smart project at Cornell University has been interested in the analysis, search, and retrieval of heterogeneous text databases, where the vocabulary is allowed to vary widely, and the subject matter is unrestricted. Such databases may include newspaper articles, newswire dispatches, textbooks, dictionaries, encyclopedias, manuals, magazine articles, and so on. The usual text analysis and text indexing approaches that are based on the use of thesauruses and other vocabulary control devices are difficult to apply in unrestricted text environments, because the word meanings are not stable in such circumstances and the interpretation varies depending on context. The applicability of more complex text analysis systems that are based on the construction of knowledge bases covering the detailed structure of particular subject areas, together with inference rules designed to derive relationships between the relevant concepts, is even more questionable in such cases. Complete theories of knowledge representation do not exist, and it is unclear what concepts, concept relationships, and inference rules may be needed to understand particular texts.[11]

Accordingly, a text analysis and retrieval component must necessarily be based primarily on a study of the available texts themselves. Fortunately very large text databases are now available in machine-readable form, and a substantial amount of information is automatically derivable about the occurrence properties of words and expressions in natural-language texts, and about the contexts in which the words are used. This information can help in determining whether a query and a text are semantically homogeneous, that is, whether they cover similar subject areas. When that is the case, the text can be retrieved in response to the query.

Automatic Indexing

In the Smart system, the vector-processing model of retrieval is used to transform both the available information requests as well as the stored documents into vectors of the form:

$$D_i = (w_{i1}, w_{i2}, \ldots, w_{it})$$

where D_i represents a document (or query) text and w_{ik} is the weight of term T_k in document D_i. A weight of zero is used for terms that are absent from a particular document, and positive weights characterize terms actu-

*Department of Computer Science, Cornell University, Ithaca, NY 14853-7501. This study was supported in part by the National Science Foundation under grant IRI 89-15847.

ally assigned. The assumption is that t terms in all are available for the representation of the information.

In choosing a term weighting system, low weights should be assigned to high-frequency terms that occur in many documents of a collection, and high weights to terms that are important in particular documents but unimportant in the remainder of the collection. The weight of terms that occur rarely in a collection is relatively unimportant, because such terms contribute little to the needed similarity computation between different texts.

A well-known term weighting system following that prescription assigns weight w_{ik} to term T_k in query Q_i in proportion to the frequency of occurrence of the term in Q_i, and in inverse proportion to 'the number of documents to which the term is assigned.[12, 10] Such a weighting system is known as a tf×idf (term frequency times inverse document frequency) weighting system. In practice the query lengths, and hence the number of nonzero term weights assigned to a query, varies widely. To allow a meaningful final retrieval similarity, it is convenient to use a length normalization factor as part of the term weighting formula. A high-quality term weighting formula for w_{ik}, the weight of term T_k in query Q_i is

$$w_{ik} = \frac{(\log(f_{ik}) + 1.0) * \log(N/n_k)}{\sqrt{\sum_{k=1}^{t}[(\log(f_{ik}) + 1.0) * \log(N/n_k)]^2}} \quad (1)$$

where f_{ik} is the occurrence frequency of T_k in Q_i, N is the collection size, and n_k the number of documents with term T_k assigned. The factor $\log(N/n_k)$ is an inverse collection frequency ("idf") factor which decreases as terms are used widely in a collection, and the denominator in expression (1) is used for weight normalization. This particular form will be called "ltc" weighting within this paper.

The weights assigned to terms in *documents* are much the same. In practice, for both effectiveness and efficiency reasons the *idf* factor in the documents is dropped.[1]

The terms T_k included in a given vector can in principle represent any entities assigned to a document for content identification. In the Smart context, such terms are derived by a text transformation of the following kind:[10]

1. recognize individual text words
2. use a stop list to eliminate unwanted function words
3. perform suffix removal to generate word stems
4. optionally use term grouping methods based on statistical word co-occurrence or word adjacency computations to form term phrases (alternatively syntactic analysis computations can be used)
5. assign term weights to all remaining word stems and/or phrase stems to form the term vector for all information items.

Once term vectors are available for all information items, all subsequent processing is based on term vector manipulations.

The fact that the indexing of both documents and queries is completely automatic means that the results obtained are reasonably collection independent and should be valid across a wide range of collections. No human expertise in the subject matter is required for either the initial collection creation, or the actual query formulation.

Phrases

The same phrase strategy (and phrases) used in TREC 1 ([1]) is used for TREC 2. Any pair of adjacent non-stopwords are regarded as potential phrases. The final list of phrases is composed of those pairs of words occurring in 25 or more documents of the initial TREC 1 document set (D1, TREC 1 initial collection). Phrase weighting is again a hybrid scheme where phrases are weighted with the same scheme as single terms, except that normalization of the entire vector is done by dividing by the length of the single term subvector only. In this way, the similarity contribution of the single terms is independent of the quantity or quality of the phrases.

Text Similarity Computation

When the text of document D_i is represented by a vectors of the form $(d_{i1}, d_{i2}, \ldots, d_{it})$ and query Q_j by the vector $(q_{j1}, q_{j2}, \ldots, q_{jt})$, a similarity (S) computation between the two items can conveniently be obtained as the inner product between corresponding weighted

term vector as follows:

$$S(D_i, Q_j) = \sum_{k=1}^{t} (d_{ik} * q_{jk}) \qquad (2)$$

Thus, the similarity between two texts (whether query or document) depends on the weights of coinciding terms in the two vectors.

Information retrieval and text linking systems based on the use of global text similarity measures such as that of expression (2) will be successful when the common terms in the two vectors are in fact used in semantically similar ways. In many cases it may happen that highly-weighted terms that contribute substantially to the text similarity are semantically distinct. For example, a sound may be an audible phenomenon, or a body of water.

TREC 1 ([1]) demonstrated that local contexts could be used to disambiguate word senses, for example rejecting documents about "industrial salts" when given a query about the "SALT peace treaty". Overall, however, the improvement in effectiveness due to local matching was minimal in TREC 1. One reason for this is the richness of the TREC queries. Global text matching is almost invariably sufficient for disambiguation. Another reason is the homogeneity of the queries. They deal primarily with two subjects: finance, and science and technology. Within a single subject area vocabulary is more standardized and ambiguity is therefore minimized.

One other potential reason for the unexpectedly slight improvement is that most of the information from local matches is simply being thrown away. Local matches are used as a filter to reject documents that do not satisfy a local criteria: the overall global similarity used for ranking is changed only by the addition of a constant indicating the local match criteria was satisfied. The positive information that a long document might have a single paragraph which very closely matched the query is ignored.

For TREC 2, we look at combining global and local similarities into a single final similarity to be used for ranking purposes.

The other focus of our TREC 2 work is taking advantage of the vast quantity of relevance judgements available for the routing experiments. In TREC 1, the relevance information was fragmentary and even occasionally incorrect. It was hard to use this information in a reasonable fashion. Happily, the results of the TREC 1 experiments furnished a large number of very good relevance judgements to be used for TREC 2. Conventional vector-space feedback methods of query expansion and reweighting are tuned for the TREC environment in the routing portion of TREC 2.

System Description

The Cornell TREC experiments use the SMART Information Retrieval System, Version 11, and are run on a dedicated Sun Sparc 2 with 64 Mbytes of memory and 5 Gbytes of local disk.

SMART Version 11 is the latest in a long line of experimental information retrieval systems, dating back over 30 years, developed under the guidance of G. Salton. Version 11 is a reasonably complete re-write of earlier versions, and was designed and implemented primarily by C. Buckley. The new version is approximately 44,000 lines of C code and documentation.

SMART Version 11 offers a basic framework for investigations into the vector space and related models of information retrieval. Documents are fully automatically indexed, with each document representation being a weighted vector of concepts, the weight indicating the importance of a concept to that particular document (as described above). The document representatives are stored on disk as an inverted file. Natural language queries undergo the same indexing process. The query representative vector is then compared with the indexed document representatives to arrive at a similarity (equation (2)), and the documents are then fully ranked by similarity.

Ad-hoc Results

Cornell submitted two runs in the ad-hoc category. The first, `crnlV2`, is a very simple vector comparison. The second, `crnlL2`, makes use of simplified least squares analysis and a training set to combine global similarity and partwise similarities in a meaningful ratio. Both systems performed at or above the median in almost all queries, as can be seen in in Table 1. The `crnlV2-b` run is the same as the official `crnlV2` run, but with an error in the experimental procedure corrected (discussed below).

Run	Best	$\geq$ median	$<$ median
crnlV2	1	38	11
crnlL2	4	40	6
crnlV2-b	9	36	5

Table 1: Comparative Ad-hoc results

Global Similarity

The crnlV2 run demonstrates the quality of results obtainable with simple methods. The weighting for terms is chosen based upon results from TREC 1. Query terms are weighted by the formula in equation (1) ("ltc" in Smart's vocabulary). Document terms are weighted using a normalized logarithmic term frequency ("lnc"):

$$d_{ik} = \frac{\log f_{ik} + 1.0}{\sqrt{\sum_{j=1}^{t} (\log f_{ij} + 1.0)^2}} \qquad (3)$$

where d_{ik} is the weight of term T_k in document D_i, f_{ik} is the occurrence frequency of term T_k in document D_i, and t is the total number of terms in the collection. The denominator provides normalization of vector length. Note the absence of the "idf" factor $\log(N/n_k)$.

Table 2 shows the results of that weighting scheme in crnlV2-b. Regrettably, because of an oversight during the official run (a misnamed inverted file), the official submitted run crnlV2 did not use the weighting approach described above (and recommended in our TREC 1 report). Instead crnlV2 accidentally used the "idf" factor in both the query *and* the document. That mistake caused 10% loss in retrieval effectiveness (from a recall-precision average of 0.3512 to 0.3163).

Global and Local

Cornell's TREC 1 ad-hoc submission increased the similarity measure of a query and document if some sentence in the query matched some sentence in the document sufficiently well.[1] The result was that any query/document pair which contained a sentence match was retrieved before all that did not have such a match. For TREC 2, we hoped to find a less restrictive balance between the global and local similarities. At the same time, we wished to investigate local similarities using parts other than sentences, and to investigate combining *multiple* local similarities.

Our approach is similar to that used in [4]. We built a training collection using the 50 queries from Q2 and the 74,520 documents from the Wall Street Journal included in D2. For each of the 3.7 million query/document pairs, we calculate the global similarity and some set of local similarity values. The least squares polynomials (LSP) approach developed for [4] are used to find the "ideal" coefficients for the global and local values in the equation:

$$\text{sim} = \alpha \cdot \text{global} + \beta_1 \cdot \text{local}_1 + \beta_2 \cdot \text{local}_2 + \ldots$$

(The LSP functions actually yield a constant factor which we ignore since it does not affect ranking.)

We consider local values from the following broad classes:

- Comparing sentences of the query against sentences of the document. In general, we use a simple "tf×idf" weight without normalization, though we experimented with other weights.
- Comparing paragraphs of the query against paragraphs of the document. (For the most part, each section of the query topic is a separate paragraph.) In this case, we use the weighting of equation 1 above for the query paragraphs, and try a variety of weights for the document paragraphs.
- Comparing the query against paragraphs of the document. We use the same weighting schemes as above.

We also tried combinations of the above categories: e.g., the best matching paragraph pair *and* the best matching sentence pair. See Table 3 for a complete list of local values that were considered.

In all, we tried 72 combinations of local and global values, using from one local value to 19 different local values.[1] The LSP-determined α and β_i's of those values are then applied to a retrieval run on that same set of queries and documents. The top performing result includes only a single local value: the best match

[1] There were roughly 1.2 million possible combinations; we chose 72 that seemed, based on earlier experiments, likely to succeed.

Run	R-prec	Total Rel	recall-prec
`crnlV2`	3640	8018	3163
`crnlV2-b`,	4053	8256	3512
`crnlV2-b`, (no *not's*	4061	8254	3560
`crnlL2`	3641	8224	3258
`crnlL2-b`	3922	8379	3538
sentence restricted	3960	8252	3477

Table 2: Ad-hoc results

of the query against the paragraphs of the candidate document (III.a.1 from Table 3), with the query terms weighted 1 iff present, and the document terms weighted using formula 1 above (that used by the query in the global similarity).

We then use the global/local values in a series of retrieval runs using the same queries but against the entire TREC 1 document set (D12). We tried a range of α and β values and use the best values for the official run, `crnlL2`. The formula used for `crnlL2` is:

$$\text{sim} = 100 \cdot \text{global} + 16 \cdot \text{local}$$

where "global" is the query/document similarity described above ("ltc-lnc"), and "local" is the top query/paragraph similarity.

It takes roughly 5 hours clock time to determine the suggested weighting coefficients, though multiple combinations of values could be weighted simultaneously—in one case, we calculated each of the 48 possible local variables simultaneously. Each of the retrospective runs takes from 60 to 90 minutes to run, depending on its complexity. These runs take an unusually large amount of time (compared to `crnlV2`) since they require re-indexing from scratch a large number of documents. The basic procedure is to discover the top 1750 documents for each query using the global similarity. Then each of those documents is re-indexed, breaking it down into its component parts (e.g., paragraphs). Then each component part is compared against the query to obtain local similarities.

Other Experiments

The Smart indexing procedures that are used in our experiments do not analyze the documents or queries for negative terms such as *not*. A query which explicitly requests documents "*not* about the United Kingdom or Canada" will actually match any document with those terms. Removing the negative keywords results in insignificant improvement: 16 queries are helped, 16 are hurt, all in only a minor fashion. These results suggest that other terms in the query were more important for locating the relevant documents.

Earlier experiments with an on-line encyclopedia ([14, 16]) demonstrated that precision can be improved by discarding documents which fail a local context check (cf. [1] where such documents were merely given lower similarity measures). That approach on the TREC 2 queries and collection yields almost exactly the same performance as `crnlV2-b` (see "sentence restricted" in Table 2). [1] discusses probable reasons for the limited success of this method.

Analysis of Ad-hoc Results

The results of Table 2 suggest that there is little advantage to using local values in combination with global matches. From run `crnlV2` to run `crnlL2` there is negligible improvement, `crnlL2` does retrieve an additional 120-200 relevant documents.

The retrospective runs using the Wall Street Journal sub-collection suggested there would be greater improvement between `crnlV2` and `crnlL2` than actually occurred. The most obvious problem is that the definition of a paragraph is sub-collection dependent. Our results were tailored to the WSJ sub-collection and probably did not apply well to the other sub-collections where "paragraphs" might be extremely large.

Name	Num	Description
Pairs	I	Query sentences vs. doc sentences
	II	Query paragraphs vs. doc paragraphs
	III	Entire query vs. doc paragraphs
Which	a	Top matching pair
	b	Non-zero matching pairs
	c	Pairs where similarity exceeds threshold
	d	*All* pairs
Value	1	Similarity (avg)
	2	Number of common terms (avg)
	3	Top matching term (avg)
	4	Count of pairs

Table 3: Local values considered for LSP weighting
(all combinations, choosing one from each category)

Future work

We are currently investigating the use of regression analysis to find correlation between relevance and local similarity values. Using such analysis will allow the local values to be selected for cause rather than solely because of experience and intuition. If successful, it will also provide a collection-independent method of selecting which local values are useful. Note that this approach does require a training set of queries and relevance judgements.

We are interested in applying these techniques to the TREC collections with a more useful definition of "paragraph." [17, 2] suggest the possibility of narrowing the search window to fixed-size pieces, ignoring paragraph boundaries. Hearst's "TextTiling" approach ([7]) is intriguing for the topic-coherent units of text it produces.

Routing

In this work, routing queries are formed in two distinct phases. In the first phase, concepts which occur often in relevant documents are added to the original query to expand the vocabulary used. In the second phase, the original concepts *plus* the added concepts are weighted based upon their occurrences in relevant and non-relevant documents.

In TREC 1, query expansion was a major obstacle. It was clear that only very limited expansion was useful, and indeed the best automatic routing run ([5]) used no expansion at all. Thus the original plans for TREC 2 routing included extensive investigation into very selectively adding concepts to queries.

However, as work on TREC 2 progressed it became obvious that the TREC 1 results were somewhat anomalous. For the routing approaches used in this work, selectivity of added terms is not an issue. Rather, the more terms that are added, the better the result—up to a point of diminishing returns. This result agrees with our experiences on the (small) feedback test collections that we have worked with in the past. The original TREC 1 training data for routing was extremely sketchy and the resulting unusual query expansion results were probably due to the lack of information about what a representative relevant document looked like.

The basic routing approach chosen is the feedback approach of Rocchio ([9, 13]). Expressed in vector space terms, the final query vector is the initial query vector moved toward the centroid of the relevant documents, and away from the centroid of the non-relevant documents.

$$\begin{aligned} Q_{\text{new}} &= A * Q_{\text{old}} \\ &+ B * \text{average\_wt\_in\_rel\_docs} \\ &- C * \text{average\_wt\_nonrel\_docs} \end{aligned}$$

Terms that end up with negative weights are dropped (less than 3% of terms were dropped in the most massive query expansion below).

The parameters of Rocchio's method are the relative importance of the original query, the

relevant documents, and the non-relevant documents (A,B,C above); and then exactly what terms are to be considered part of the final vector.

In TREC 1, a similar approach originally proposed by Ide was used.[8, 13] That seemed to work well for the fragmentary relevance information available for TREC 1. TREC 2 has considerably more information available—and of much higher quality—so Rocchio's approach is more appropriate.

The best parameter values to use for Rocchio's algorithm were investigated by splitting the TREC 1 collection into two parts along the natural D1 (TREC 1 initial collection) and D2 (TREC 1 routing collection) lines. D1 formed the learning set and D2 the evaluation set for a large number of experimental runs determining these parameters. The original TREC 1 routing queries (Q2) are expanded and weighted using Rocchio's algorithm with the relevance information from D1. They are then evaluated by running them against D2 and using the known Q2–D2 relevance information.

Queries are expanded by adding the "best" X single terms and the "best" Y phrases to the original query. We used a simple notion of "best" for TREC 2: terms that occurred in the most relevant documents (ties were broken by considering the highest average weight in the relevant documents).

There is a core set of 158 runs using different parameter values for both expansion and weighting. Table 4 gives the six parameter possibilities The trends noticeable in this investigatory set of runs are:

1. Overall effectiveness increases strongly as the number of terms added increases, up until 200 terms at which point it starts to level off.

2. Phrases are reasonably important (6% difference) at low single term expansion numbers, but become less important at higher values (1% difference)

3. As expected, weights in relevant documents are far more important than weights in non-relevant documents.

The parameters of our official run, `crnlR1` are: adding $X = 300$ single terms, adding $Y = 50$ phrases, importance of original query of $A = 8$, importance of weight in relevant documents of $B = 16$, importance of weight in non-relevant documents of $C = 4$, and relative importance of phrases at retrieval time of $P = 0.5$.

Query-by-Query Parameter Estimation

We examined the results for the 158 test routing runs in more detail, query by query. For each of the 50 queries, we found the best test run. The results (see Table 5) show some interesting patterns not brought out by the overall averages. Not surprisingly, the parameters used for `crnlR1` are not best for any single query; they are just a reasonable compromise. There seem to be two main groups of queries: one in which very limited expansion is useful (even 6 queries where *no* expansion is preferred); and one in which the more terms are added, the better (23 queries with expansion of 500 single terms). If massive expansion is useful, in general the original query is less important than the expanded terms: A is much less than B. There is another separate distinction between those queries where phrases are useful and those where phrases appear useless: 1 query worked best adding 100 phrases, 6 with 50 added, 2 with 10, 16 using the original phrases only, and 25 using no phrases at all.

If we retrospectively choose the best parameters for each query (something that cannot be done in practice), then we achieve roughly a 10% improvement. This is substantial enough to actually try a predictive run, so our second official run (`crnlC1`) uses query-by-query choice of parameter values in a predictive (as opposed to retrospective) fashion. The values given in Table 5 were used.

Routing Results

Both `crnlR1` and `crnlC1` do extremely well in comparison with other TREC 2 routing runs:

Run	Best	$\geq$ median	$<$ median
`crnlR1`	7	40	3
`crnlC1`	5	45	0

Evaluation measures in Table for both the official and some non-official runs show the importance of query expansion. Run 1 is the base case original query only (ltc weights).

X	number of single terms to add (possible values 0 to 500)
Y	number of phrases to add (0 to 100)
A	relative importance of original query (fixed at 8)
B	relative importance of average weight in relevant documents (4 to 48)
C	relative importance of average weight in non-relevant documents (0 to 16)
P	relative importance of phrases in final retrieval as compared to single terms (0, 0.5, or 1.0)

Table 4: Parameters of routing

Just re-weighting the query terms according to Rocchio's algorithm gives a 7% improvement. Adding a few terms (20 single terms + 10 phrases) gives 17% improvement over the base case, and expanded by 350 (300+50) terms results in a 38% improvement.

The official run `crnlC1` is actually a bit disappointing. It only results in a 3% improvement over the `crnlR1` run, which is not very significant considering the effort required. Few people are going to keep track of 158 test runs on a per query basis. It may be practical to keep track of 4 or so main query variants, but then the improvement would probably be less than 3%. We are conducting experiments in this area currently.

An open question is the effectiveness of varying the feedback approach itself between queries. Preliminary experiments using Fuhr's RPI ([3]) weighting schemes in addition to the Rocchio variants show larger improvements. In general, RPI (and the other probabilistic models) perform noticeably better than Rocchio if there is very little query expansion, though quite a bit worse under massive expansion. We expect that the combination of RPI for those queries with little expansion and Rocchio for other queries will work well.

One benefit of the `crnlC1` run not entirely represented by the evaluation figures is that retrieval performance is more even. Potential mismatches between feedback method and query are far less likely. `crnlC1` does reasonably on all the queries (above the median system for every query when compared against the other systems).

Routing Implementation and Timing

The original routing queries are automatically indexed from the query text, and weighted using the "ltc" weighting scheme (equation (1)). Collection frequency information used for the idf factors is gathered from D12 documents only. Relevance information about potential query terms is gathered and stored on a per query basis. For each query, statistics (including relevant and non-relevant frequency and total "ltc" weights) are kept about the 1000 most frequently occurring terms in the D12 relevant documents. For TREC 2, this is done by a batch run taking about 90 CPU minutes. In practice, this would be done incrementally as each document was compared to the query and judged. The statistics amounted to about 40,000 bytes per query.

Using these statistics, and the decided upon parameters for the feedback process (A, B, etc.), actual construction of the final query takes about 0.5 seconds per query.

Retrieval times vary tremendously with length of query. We ran in batch mode, constructing an inverted file for the entire D3 document set ("lnc" document weights) and then comparing a query against that inverted file. Not only is this not what would be done in practice, but it is much less efficient than would be done in practice given our massive expansion of queries: for each query in `crnlR1`, well over half the entire inverted file was read! CPU time per query ranged from about 5 seconds (no expansion) to 65 seconds (expansion by 500 terms).

Conclusion

No firm conclusions can be reached regarding the usefulness of combining local and global similarities in the TREC environment. In some limited circumstances minor improvements can be obtained, but in general we have not (yet!) been able to take advantage of the local information we know should be useful.

Query	Phrase Import	Single Expand	Phrase Expand	Orig Query	Weight Relevant	Weight Non-relevant
51	0.0	500	·	8	24	4
52	0.0	200	·	8	16	0
53	1.0	500	50	8	48	4
54	0.0	200	·	8	36	4
55	0.0	200	·	8	24	8
56	1.0	0	0	8	8	4
57	0.0	500	·	8	24	4
58	1.0	500	0	8	48	4
59	0.5	100	10	8	8	4
60	1.0	0	0	8	8	4
61	0.0	0	·	8	8	4
62	0.0	500	·	8	24	4
63	1.0	0	0	8	8	4
64	0.0	500	·	8	16	4
65	1.0	200	0	8	16	4
66	1.0	200	0	8	16	4
67	0.0	500	·	8	16	4
68	1.0	300	0	8	48	4
69	1.0	500	0	8	24	4
70	1.0	500	50	8	48	4
71	0.0	500	·	8	8	4
72	1.0	300	0	8	48	4
73	0.0	500	·	8	16	4
74	1.0	100	100	8	8	4
75	0.0	100	·	8	24	6
76	1.0	300	0	8	24	4
77	0.0	500	·	8	8	4
78	0.0	0	·	8	24	4
79	0.0	500	·	8	24	4
80	1.0	500	0	8	24	4
81	1.0	500	0	8	32	4
82	1.0	500	0	8	32	4
83	0.0	500	·	8	24	4
84	0.0	100	·	8	36	4
85	1.0	100	10	8	24	4
86	0.0	30	·	8	24	4
87	1.0	500	0	8	48	4
88	1.0	500	50	8	48	4
89	1.0	500	0	8	48	4
90	0.0	500	·	8	16	4
91	1.0	0	0	8	24	4
92	0.0	100	·	8	36	4
93	1.0	100	50	8	8	4
94	0.0	200	·	8	36	4
95	0.0	200	·	8	16	8
96	1.0	300	50	8	48	4
97	0.0	200	·	8	36	4
98	0.0	500	·	8	24	4
99	0.0	50	·	8	24	4
100	1.0	500	50	8	48	4

Table 5: Optimum routing parameters, query-by-query

		X.Y	*A.B.C*	R-prec	Total Rel	recall-prec
1.	no fdbk	0.0	8.0.0	3382	6509	2869
2.	no expand	0.0	8.8.4	3531	6849	3087
3.	little expand	20.10	8.8.4	3756	7192	3345
4.	`crnlR1`	300.50	8.16.4	4273	7764	3952
5.	`crnlC1`	varies	varies	4367	7808	4091

Table 6: Routing evaluation

For TREC 2, this failure is ameliorated by the base level performance of the global run. If the correct weights are used, the effectiveness of automatic indexing is extremely good.

Automatic massive query expansion proves to be very effective for routing. Conventional relevance feedback techniques are used to weight the expanded queries. Parameters for the relevance feedback algorithms are estimated both over all the queries and for each query individually. The individual query estimation perform better (3–4%) but by an insufficient amount to be convincing.

References

[1] Chris Buckley, Gerard Salton, and James Allan. Automatic retrieval with locality information using SMART. In D. K. Harman, editor, *Proceedings of the First Text REtrieval Conference (TREC-1)*, pages 59–72. NIST Special Publication 500-207, March 1993.

[2] James P. Callan and W. Bruce Croft. An evaluation of query processing strategies using the tipster collection. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 347–355, June 1993.

[3] Norbert Fuhr. Models for retrieval with probabilistic indexing. *Information Processing and Management*, 25(1):55–72, 1989.

[4] Norbert Fuhr and Chris Buckley. Automatic structuring of text files. *ACM Transactions on Information Systems*, 9(3):223–248, 1991.

[5] Norbert Fuhr and Chris Buckley. Optimizing document indexing and search term weighting based on probabilistic models. In D. K. Harman, editor, *Proceedings of the First Text REtrieval Conference (TREC-1)*, pages 89–99. NIST Special Publication 500-207, March 1993.

[6] D. Harman. Towards interactive query expansion. In *Proceedings of the Eleventh International Conference on Research and Development in Information Retrieval*, pages 321–331. Association for Computing Machinery, 1988.

[7] Marti A. Hearst and Christian Plaunt. Subtopic structuring for full-length document access. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 59–68, June 1993.

[8] E. Ide. New experiments in relevance feedback. In Gerard Salton, editor, *The SMART Retrieval System—Experiments in Automatic Document Processing*, chapter 16. Prentice Hall, Englewood Cliffs, NJ, 1971.

[9] J.J. Rocchio. Relevance feedback in information retrieval. In Gerard Salton, editor, *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice Hall, Englewood Cliffs, NJ, 1971.

[10] Gerard Salton. *Automatic Text Processing — the Transformation, Analysis and Retrieval of Information by Computer*. Addison-Wesley Publishing Co., Reading, MA, 1989.

[11] Gerard Salton. Developments in automatic text retrieval. *Science*, 253:974–980, August 1991.

[12] Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[13] Gerard Salton and Chris Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288–297, 1990.

[14] Gerard Salton and Chris Buckley. Automatic text structuring and retrieval: Experiments in automatic encyclopedia searching. In *Proceedings of the Fourteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–30, 1991.

[15] Gerard Salton and Chris Buckley. Global text matching for information retrieval. *Science*, 253:1012–1015, August 1991.

[16] Gerard Salton, Chris Buckley, and James Allan. Automatic structuring of text files. *Electronic Publishing*, 5(1):1–17, March 1992.

[17] Craig Stanfill and David L. Waltz. Statistical methods, artificial intelligence, and information retrieval. In Paul S. Jacobs, editor, *Text-based Intelligent Systems: Current Research and Practice in Information Extraction and Retrieval.* Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, 1971.

Full Text Retrieval based on Probabilistic Equations with Coefficients fitted by Logistic Regression

Wm. S. Cooper
Aitao Chen
Fredric C. Gey

S.L.I.S., University of California
Berkeley, CA 94720

ABSTRACT

The experiments described here are part of a research program whose objective is to develop a full-text retrieval methodology that is statistically sound and powerful, yet reasonably simple. The methodology is based on the use of a probabilistic model whose parameters are fitted empirically to a learning set of relevance judgements by logistic regression. The method was applied to the TIPSTER data with optimally relativized frequencies of occurrence of match stems as the regression variables. In a routing retrieval experiment, these were supplemented by other variables corresponding to sums of logodds associated with particular match stems.

Introduction

The full-text retrieval design problem is largely a problem in the combination of statistical evidence. With this as its premise, the Berkeley group has concentrated on the challenge of finding a statistical methodology for combining retrieval clues in as powerful a way as possible, consistent with reasonable analytic and computational simplicity. Thus our research focus has been on the general logic of how to combine clues, with no attempt made at this stage to exploit as many clues as possible. We feel that if a straightforward statistical methodology can be found that extracts a maximum of retrieval power from a few good clues, and the methodology is clearly hospitable to the introduction of further clues in future, progress will have been made.

We join Fuhr and Buckley (1991, 1992) in thinking that an especially promising path to such a methodology is to combine a probabilistic retrieval model with the techniques of statistical regression. Under this approach a probabilistic model is used to deduce the general form that the document-ranking equation should take, after which regression analysis is applied to obtain empirically-based values for the constants that appear in the equation. In this way the probabilistic theory is made to constrain the universe of logically possible retrieval rules that could be chosen, and the regression techniques complete the choice by optimizing the model's fit to the learning data.

The probabilistic model adopted by the Berkeley group is derived from a statistical assumption of 'linked dependence'. This assumption is weaker than the historic independence assumptions usually discussed. In its simplest form the Berkeley model also differs from most traditional models in that it is of 'Type 0' -- meaning that the analysis is carried out w.r.t. sets of query-document pairs rather than w.r.t. particular queries or particular documents. (For a fuller explanation of this typology see Robertson, Maron & Cooper 1982.) But when relevance judgement data specialized to the currently submitted search query is available, say in the form of relevance feedback or routing history data, the model is flexible enough to accommodate it (resulting in 'Type 2' retrieval.)

Logistic regression (see e.g. Hosmer & Lemeshow (1989)) is the most appropriate type of regression for this kind of IR prediction. Although standard multiple regression analysis has been used successfully by others in comparable circumstances (Fuhr & Buckley op. cit.), we believe logistic regression to be logically more appropriate for reasons set forth elsewhere (Cooper, Dabney & Gey 1992). Logistic regression, which accepts binary training data and yields probability estimates in the form of logodds values, goes hand-in-glove with a probabilistic IR model that is to be fitted to binary relevance judgement data and whose predictor variables are themselves logodds.

The Fundamental Equation

As the starting point of our probabilistic retrieval model we adopt (with reservations to be explained presently) a statistical simplifying assumption called the 'Assumption of Linked Dependence' (Cooper 1991). Intuitively, it states that in the universe of all query-document pairs, the degree of statistical dependency that exists between properties of pairs in the subset of all relevance-related pairs is linked in a certain way with the degree of dependency that exists between the same pair-properties in the subset of all nonrelevance-related pairs. Under at least one crude measure of degree of dependence (specifically, the ratio of the joint probability to the product of individual probabilities), the linkage in question is one of identity. That is, the claim made by the assumption is that the degree of the dependency is the same in both sets.

For N pair-properties $A_1, \ldots, A_N$, the mathematical statement of the assumption is as follows.

ASSUMPTION OF LINKED DEPENDENCE:

$$\frac{P(A_1, \ldots, A_N \mid R)}{P(A_1, \ldots, A_N \mid \bar{R})} = \frac{P(A_1 \mid R)}{P(A_1 \mid \bar{R})} \times \cdots \times \frac{P(A_N \mid R)}{P(A_N \mid \bar{R})}$$

If one thinks of a query and document drawn at random, R is the event that the document is relevant to the query, while each clue A_i is some respect in which the document is similar or dissimilar to, or somehow related to, the query.

For most clue-types likely to be of interest this simplifying assumption is at best only approximately true. Though not as implausible as the strong conditional independence assumptions traditionally discussed in probabilistic IR, still it should be regarded with skepticism. In practice, when the number N of clues to be combined becomes large the assumption can become highly unrealistic, with a distorting tendency that often causes the probability of relevance estimates produced by the resulting retrieval model to be grossly overstated for documents near the top of the output ranking. But it will be expedient here to adopt the assumption anyway, on the understanding that later on we shall have to find a way to curb its probability-inflating tendencies.

From the Assumption of Linked Dependence it is possible to derive the basic equation underlying the probabilistic model:

Equation (1):

$$\log O(R \mid A_1, \ldots, A_N) =$$

$$\log O(R) + \sum_{i=1}^{N} [\log O(R \mid A_i) - \log O(R)]$$

where for any events E and E' the odds $O(E \mid E')$ is by definition $\frac{P(E \mid E')}{P(\bar{E} \mid E')}$. Using this equation, the evidence of N separate clues can be combined as shown in the right side to yield the logodds of relevance based on all clues, shown on the left. Query-document pairs can be ranked by this logodds estimate, and a ranking of documents for a particular query by logodds is of course a probability ranking in the IR sense. For further discussion of the linked dependence assumption and a formal derivation of Eq. (1) from it, see Cooper (1991) and Cooper, Dabney & Gey (1992). An empirical investigation of independence and dependence assumptions in IR is reported by Gey (1993).

Application to Term Properties

We shall consider as a first application of the modeling equation the problem of how to exploit the properties of match terms. By a 'match' term we mean a stem (or word, phrase, etc.) that occurs in the query and also, in identical or related form, in the document to which the query is being compared. The retrieval properties of a match term could include its frequency characteristics in the query, in the document, or in the collection as a whole, its grammatical characteristics, the type or degree of 'match' involved, etc. If there are M match terms that relate a certain query to a certain document, Eq. (1) becomes applicable with N set to M. Each of the properties $A_1, \ldots, A_M$ then represents a composite clue or set of properties concerning one of the query-document pair's match terms.

Suppose a 'learning set' of human judgements of relevance or nonrelevance is available for a sample of representative query-document pairs. (However, for the time being we assume no such learning data is available for the very queries on which the retrieval is to be performed.) Logistic regression can be applied to the data to develop an equation capable of using the match term clues to estimate, for any query-document pair, values for each of the expressions in the right side of Eq. (1). Eq. (1) then yields a probability estimate for the pair.

TABLE I: Equations to Approximate the Components of Eq. (1)

$$\log O(R) \approx [\, b_0 + b_1 M]$$

$$\log O(R \mid A_1) - \log O(R) \approx [\, a_0 + a_1 X_{1,1} + \cdots + a_K X_{1,K} + a_{K+1} M \,] - [\, b_0 + b_1 M]$$

$$\vdots \qquad \vdots \qquad \approx \qquad \vdots \qquad \vdots$$

$$\log O(R \mid A_M) - \log O(R) \approx [\, a_0 + a_1 X_{M,1} + \cdots + a_K X_{M,K} + a_{K+1} M \,] - [\, b_0 + b_1 M]$$

$$\log O(R \mid A_1, \ldots, A_M) \approx [\, a_0 M + a_1 \sum_{m=1}^{M} X_{m,1} + \cdots + a_K \sum_{m=1}^{M} X_{m,K} + a_{K+1} M^2 \,] - (M-1)[\, b_0 + b_1 M \,]$$

What must be done, then, is to find approximating expressions for the various components in the right side of Eq. (1). Table I (above) shows the interrelationships among the quantities involved. In this array, the material to the left of the column of approximation signs just restates Eq. (1), for Eq. (1) asserts that the expressions above the horizontal line sum to the expression below the line. On the right of each approximation sign is a simple formula that might be used to approximate the expression on the left. (More complex formulae could of course also be considered.) Each right-side formula involves a linear combination of K variables $X_{m,1}, \ldots, X_{m,K}$ corresponding to the K retrieval properties (frequency values, etc.) used to characterize the term in question. The idea is to apply logistic regression to find the values for the coefficients (the a's and b's) in the right side that produce the best regression fit to the learning sample. Once these constants have been determined, retrieval can be carried out by summing the right sides to obtain the desired estimate of $\log O(R \mid A_1, \ldots, A_N)$ for any given query-document pair.

It may not be immediately apparent why terms in M have been included in the approximating formulas on the right. In Eq. (1), the number N of available retrieval clues is actually part of the conditionalizing information, a fact that could have been stressed by writing $O(R \mid N)$ in place of $O(R)$ and $O(R \mid A_i, N)$ in place of $O(R \mid A_i)$. So the approximating formula for $\log O(R)$ is really an approximation for $\log O(R \mid M)$ and must involve a term in M. (Simple functions of M such as $\sqrt{M}$ or $\log(M)$ might also be considered.) Similar remarks apply to the formulas for approximating $\log(R \mid A_m)$.

There are two approaches to the problem of finding coefficients to use in the approximating expressions. The first is what might be called the 'triples-then-pairs' approach. It starts out with a logistic regression analysis performed directly on a sample of query-document-term triples constructed from the learning set. In the sample, each triple is accompanied by the clue-values associated with the match term in the query and document, and by the human relevance judgement for the query and document. The clue-values are used as the values of the independent variables in the regression, and the relevance judgements as the values of the the dependent variable. After the desired coefficients have been obtained, the resulting regression equation can be applied to evaluate the left-side expressions, which can in turn be summed down the M terms to obtain a preliminary estimate of $\log O(R \mid A_1, \ldots, A_M)$.

A second regression analysis is then performed on a sample of query-document pairs, using the preliminary logodds prediction from the first (triples) analysis as an independent variable. This second analysis serves the purpose of correcting for distortions introduced by the Assumption of Linked Dependence. It also allows retrieval properties not associated with particular match terms, such as document length, age, etc., to be introduced. The triples-then-pairs approach is the one used by the Berkeley group in its Trec-1 research (Cooper, Gey & Chen 1992). The theory behind it is presented in more detail in (Cooper, Dabney & Gey 1992).

The second way of determining the coefficients -- the one used in Trec-2 -- is the 'pairs-only' approach. It requires only one regression analysis, performed on a sample of query-document pairs. It is based on the trivial observation that in the right side of the above array, instead of adding across rows and then down the resulting column of sums, one can equivalently add down columns and across the resulting row of sums. Under either procedure the grand total value for $\log O(R \mid A_1, \ldots, A_M)$ will be the same.

Summing down the columns and then across the totals gives the expression shown in the bottom line of the array. It simplifies to

$$\log O(R \mid A_1, \ldots, A_N) \approx$$

$$a_1 \sum_{m=1}^{M} X_{m,1} + \cdots + a_K \sum_{m=1}^{M} X_{m,K} + (a_0 - b_0 + b_1)M + (a_{K+1} - b_1)M^2$$

Since there is no need to keep the a_j coefficients segregated from the b_j coefficients to get a predictive equation, this suggests the adoption of a regression equation of form

$$\log O(R \mid A_1, \ldots, A_M) \approx$$

$$c_0 + c_1 \sum_{m=1}^{M} X_{m,1} + \cdots + c_K \sum_{m=1}^{M} X_{m,K} + c_{K+1}M + c_{K+2}M^2$$

The coefficients $c_0, \ldots, c_{K+2}$ may be found by a logistic regression on a sample of query-document pairs constructed from the learning sample. In the sample each pair is accompanied by its K different $X_{m,k}$-values each already summed over all match terms for the pair, the values of M and M^2, and (to serve as the dependent variable in the regression) the human relevance judgement for the pair.

But if only one level of regression analysis is to be performed, where is the correction for the Assumption of Linked Dependence to take place? That assumption causes mischief because it creates a tendency for the predicted logodds of relevance to increase roughly linearly with the number of match terms, whereas the true increase is less than linear. This can be corrected by modifying the variables in such a way that their values rise less rapidly than the number of match terms as the number of match terms increases. The variables can, for instance, be multiplied by some function $f(M)$ that drops gently with increasing M, say $\frac{1}{\sqrt{M}}$ or $\frac{1}{1+\log M}$. The exact form of the function can be decided during the course of the regression analysis.

With such a damping factor included, the foregoing regression equation becomes
Equation (2):

$$\log O(R \mid A_1, \ldots, A_M) \approx$$

$$c_0 + c_1 f(M) \sum_{1}^{M} X_{m,1} + \cdots + c_K f(M) \sum_{m=1}^{M} X_{m,K} + c_{K+1}M + c_{K+2}M^2$$

In our experiments, this simple modification was found to improve the regression fit and the precision/recall performance. It would appear therefore to be a worth-while refinement of the basic model. Note, however, that this adjustment only removes a general bias. It does nothing to exploit the possibility of measuring dependencies between particular stems to improve retrieval effectiveness. To exploit individual dependencies would be desirable in principle, but would require a substantial elaboration of the model for what might well turn out to be an insignificant improvement in effectiveness (for discussion see Cooper (1991)).

Optimally Relativized Frequencies

The philosophy of the project called for the use of a few well-chosen retrieval clues. The most obvious clues to be exploited in connection with match terms are their frequencies of occurrence in the query and document. What is not so obvious is the exact form the frequencies should take. For instance, should they be absolute or relative frequencies, or something in between?

The IR literature mentions two ways in which frequencies might be relativized for use in retrieval. The first is to divide the absolute frequency of occurrence of the term in the query or document by the length of the query or document, or some parameter closely associated with length. The second is to divide the relative frequency so obtained by the relative frequency of occurrence of the term in the entire collection considered as one long running text. Both kinds of relativization seem potentially beneficial, but the question remains whether these relativizations, if they are indeed helpful, should be carried out in full strength, or whether some sort of blend of absolute and relative frequencies might serve better.

To answer this question, regression techniques were used in a side investigation to discover the optimal extent

to which each of the two kinds of relativization might best be introduced. To investigate the first kind of relativization -- relativization with respect to query or document length -- a semi-relativized stem frequency variable of the form

$$\frac{absolute\ \ frequency}{document\ length\ +\ C}$$

was adopted. If the constant C is chosen to be zero, one has full relativization, whereas a value for C much greater than the document length will cause the variable to behave in the regression analysis as though it were an absolute frequency. Several logistic regressions were run to discover by trial-and-error the value of C that produced the best regression fit to the TIPSTER learning data and the highest precision and recall. It was found that a value of around $C = 35$ for queries, and $C = 80$ for documents, optimized performance.

For the relativization with respect to the global relative frequency in the collection, the relativizing effect of the denominator was weakened by a slightly different method -- by raising it to a power less than 1.0. For the document frequencies, for instance, a variable of form

$$\frac{\dfrac{absolute\ frequency\ in\ doc}{document\ length\ +\ 80}}{[\ relative\ frequency\ in\ collection\ ]^{\,D}}$$

with $D < 1$ was in effect used. Actually, it was the logarithm of this expression that was ultimately adopted, which allowed the variable to be broken up into a difference of two logarithmic expressions. The optimal value of the power D was therefore obtainable in a single regression as the coefficient of the logged denominator.

Thus the variables ultimately adopted consisted in essence of sums over two optimally relativized frequencies ('ORF's) -- one for match stem frequency in the query and one for match stem frequency in the document. Because of the logging this breaks up mathematically into three variables. A final logistic regression using sums over these variables as indicated in Eq. (2) produced the ranking equation

Equation (3):

$$\log O(R|\, A_1, \ldots, A_M) \approx$$

$$-3.51 + \frac{1}{\sqrt{M}+1}\ \Phi + 0.0929\ M$$

where Φ is the expression

$$37.4 \sum_{m=1}^{M} X_{m,1} + 0.330 \sum_{m=1}^{M} X_{m,2} - 0.1937 \sum_{m=1}^{M} X_{m,3}$$

Here

$X_{m,1}$ = number of times the m'th stem occurs in the query, divided by (total number of all stem occurrences in query + 35);

$X_{m,2}$ = number of times the m'th stem occurs in the document, divided by (total number of all stem occurrences in document + 80), quotient logged;

$X_{m,3}$ = number of times the m'th stem occurs in the collection, divided by the total number of all stem occurrences in the collection, quotient logged;

M = number of distinct stems common to both query and document.

Although Eq. (2) calls for an M^2 term as well, such a term was found not to make a statistically significant contribution and so was eliminated.

Eq. (3) provided the ranking rule used in the ad hoc run (labeled 'Brkly3') and the first routing run ('Brkly4') submitted for Trec-2. The equation is notable for the sparsity of the information it uses. Essentially it exploits only two ORF values for each match stem, one for the stem's frequency in the query and the other for its frequency in the document. Other variables were tried including the inverse document frequency (both logged and unlogged), a variable consisting of a count of all two-stem phrase matches, and several variables for measuring the tendency of the match stems to bunch together in the document. All of these exhibited predictive power when used in isolation, but were discarded because in the presence of the ORF's none produced any detectable improvement in the regression fit or the precision/recall performance. Some attempts at query expansion using the WordNet thesaurus also failed to produce noticable improvement, even when care was taken to create separate variables with separate coefficients for the synonym-match counts as opposed to the exact-match counts.

The quality of a retrieval output ranking matters most near the top of the ranking where it is likely to affect the most users, and matters hardly at all far down the ranking where hardly any users are apt to search. Because of this it is desirable to adopt a sampling methodology that produces an especially good regression fit to the sample

data for documents with relatively high predicted probabilities of relevance. In an attempt to achieve this emphasis, the regression analysis was carried out in two steps. A logistic regression was first performed on a stratified sample of about one-third of the then-available TIPSTER relevance judgements to obtain a preliminary ranking rule. This preliminary equation was then applied as a screening rule to the entire available body of judged pairs, and for each query all but the highest-ranked 500 documents were discarded. The resulting set of 50,000 judged query-document pairs (100 topics × 500 top-ranked docs per topic) served as the learning sample data for the final regression equation displayed above as Eq. (3).

Application to Query-Specific Learning Data

To this point it has been assumed that relevance judgement data is available for a sample of queries typical of the future queries for which retrieval is to be performed, but not for those very queries themselves. We consider next the contrasting situation in which the learning data that is available is a set of past relevance judgements for the very queries for which new documents are to be retrieved. Such data is often available for a routing or SDI request for which relevance feedback has been collected in the past, or for a retrospective feedback search that has already been under way long enough for some feedback to have been obtained.

For a query so equipped with its very own learning sample, it is possible to gather separate data about each individual term in the query. Such data reflects the term's special retrieval characteristics in the context of that query. For instance, for a query term T_i we may count the number $n(T_i, R)$ of documents in the sample that contain the term and are also relevant to the query, the number of documents $n(T_i, \bar{R})$ that contain the term but are not relevant to the query, and so forth.

The odds that a future document will turn out to be be relevant to the query, given that it contains the term, can be estimated crudely as

$$O(R \mid T_i) \approx \frac{n(T_i, R)}{n(T_i, \bar{R})}$$

To refine this estimate, a Bayesian trick adapted from Good (1965) is useful. One simply adds the two figures used in expressing the prior odds of relevance (i.e. $n(R)$ and $n(\bar{R})$) into the numerator and denominator respectively with an arbitrary weight β as follows:

$$O(R \mid T_i) \approx \frac{n(T_i, R) + \beta\, n(R)}{n(T_i, \bar{R}) + \beta\, n(\bar{R})}$$

The smaller the value assigned to β, the more influence the sample evidence will have in moving the estimate away from the prior odds. The adjustment causes large samples to have a greater effect than small, as seems natural. It also forestalls certain absurdities implicit in the unadjusted formula, for instance the possibility of calculating infinite odds estimates.

Suppose now that a query contains Q distinct terms $T_1, \ldots, T_M, T_{M+1}, \ldots, T_Q$, numbered in such a way that the first M of them are match terms that occur in the document against which the query is to be compared. The fundamental equation can be applied by taking N to be Q, and interpreting the retrieval clues $A_1, \ldots, A_N$ as the presence or absence of particular query terms in the document. The pertinent data can be organized as shown in Table II (next page).

Thus we are led to postulate, for query-specific learning data, a retrieval equation of form

Equation (4):

$$\log O(R \mid A_1, \ldots, A_Q) \approx$$

$$c_0 + c_1 f(Q)\left[\Phi_1 + \Phi_2 - \Phi_3\right]$$

where

$$\Phi_1 = \sum_{m=1}^{M} \log \frac{n(T_i, R) + \beta\, n(R)}{n(T_i, \bar{R}) + \beta\, n(\bar{R})}$$

$$\Phi_2 = \sum_{m=M+1}^{Q} \log \frac{n(\overline{T_i}, R) + \beta\, n(R)}{n(\overline{T_i}, \bar{R}) + \beta\, n(\bar{R})}$$

$$\Phi_3 = (Q-1)\, \log \frac{n(R)}{n(\bar{R})}$$

and where $f(Q)$ is some restraining function intended as before to subdue the inflationary effect of the Assumption of Linked Dependence. The coefficients c_0, c_1 are found by a logistic regression over a sample of query-document pairs involving many queries.

Combining Query-Specific and Query-Nonspecific Data

If a query-specific set of relevance judgements is available for the query to be processed, a larger

TABLE II: Reinterpretation of the Components of Eq. (1) for Routing Retrieval

$$\log O(R) \approx \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad - \log \frac{n(R)}{n(\bar{R})}$$

$$\log O(R \mid A_1) - \log O(R) \approx \log \frac{n(T_1, R) + \beta\, n(R)}{n(T_1, \bar{R}) + \beta\, n(\bar{R})} \qquad\qquad - \log \frac{n(R)}{n(\bar{R})}$$

$$\cdot \quad \cdot \quad \approx \quad \cdot \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \cdot$$

$$\cdot \quad \cdot \quad \approx \quad \cdot \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \cdot$$

$$\cdot \quad \cdot \quad \approx \quad \cdot \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \cdot$$

$$\log O(R \mid A_M) - \log O(R) \approx \log \frac{n(T_M, R) + \beta\, n(R)}{n(T_M, \bar{R}) + \beta\, n(\bar{R})} \qquad\qquad - \log \frac{n(R)}{n(\bar{R})}$$

$$\log O(R \mid A_{M+1}) - \log O(R) \approx \qquad\qquad \log \frac{n(\bar{T}_{M+1}, R) + \beta\, n(R)}{n(\bar{T}_{M+1}, \bar{R}) + \beta\, n(\bar{R})} - \log \frac{n(R)}{n(\bar{R})}$$

$$\cdot \quad \approx \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \cdot$$

$$\cdot \quad \cdot \quad \approx \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \cdot$$

$$\cdot \quad \cdot \quad \approx \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \cdot$$

$$\log O(R \mid A_Q) - \log O(R) \approx \qquad\qquad \log \frac{n(\bar{T}_Q, R) + \beta\, n(R)}{n(\bar{T}_Q, \bar{R}) + \beta\, n(\bar{R})} - \log \frac{n(R)}{n(\bar{R})}$$

$$\log O(R \mid A_1, \ldots, A_Q) \approx \sum_{1}^{M} \log \frac{n(T_i, R) + \beta\, n(R)}{n(T_i, \bar{R}) + \beta\, n(\bar{R})} + \sum_{M+1}^{Q} \log \frac{n(\bar{T}_i, R) + \beta\, n(R)}{n(\bar{T}_i, \bar{R}) + \beta\, n(\bar{R})} - (Q-1) \log \frac{n(R)}{n(\bar{R})}$$

nonspecific set may well be available at the same time. If so the theory developed in the foregoing section can be applied in conjunction with the earlier theory to capture the benefits of both kinds of learning sets. The retrieval equation will then contain variables not only of the kind occurring in Eq. (4) but also of the Eq. (2) kind.

It is convenient to formulate this equation in such a way that it contains as one of its terms the entire ranking expression developed earlier for the nonspecific learning data. For the TIPSTER data the combined equation takes the form:

Equation (5):

$$\log O(R \mid A_1, \ldots, A_M, A'_1, \ldots, A'_Q) \approx$$
$$0.688\, \Phi_4 + 0.344 \left[\Phi_1 + \Phi_2 - \Phi_3 \right] + 0.0623$$

where Φ_4 is the entire right side of Eq. (3) and Φ_1, Φ_2, Φ_3 are as defined in Eq. (4). This form for the equation is computationally convenient if Eq. (3) is to be used as a preliminary screening rule to eliminate unpromising documents, with Eq. (5) in its entirety applied only to rank those that survive the screening.

Eq. (5) was used to produce the Trec-2 routing run 'Brkly5'. Its coefficients were determined by a logistic regression constrained in such a way as to make the query-specific variables contribute about twice as heavily as the nonspecific, when contribution is measured by standardized coefficient size. This emphasis was largely arbitrary; finding the optimal balance between the query-specific and the general contributions remains a topic for future research. A value of $20/n(\bar{R})$ was used for β. This choice too was arbitrary, and it would be interesting to try to optimize it experimentally for some typical collection (trying out, perhaps, numbers larger than 20, divided by the total number of documents in the query's learning set). No restraining function $f(Q)$ was used in the final form of Eq. (5) because none that were tried out produced any discernible improvement in fit or retrieval effectiveness in this context.

Calibration of Probability Estimates

The most important role of the relevance probability estimates calculated by a probabilistic IR system is to rank the output documents in as effective a search order as possible. For this ranking function it is only the relative sizes of the probability estimates that matter, not their absolute magnitudes. However, it is also desirable that the absolute sizes of these estimates be at least somewhat realistic. If they are, they can be displayed to provide guidance to the users in their decisions as to when to stop searching down the ranking. This capability is a potentially important side benefit of the probabilistic approach.

One way of testing the realism of the probability estimates is to see whether they are 'well-calibrated'. Good calibration means that when a large number of probability estimates whose magnitudes happen to fall in a certain small range are examined, the proportion of the trials in question with positive outcomes also falls in or close to that range. To test the calibration of the probability predictions produced by the Berkeley approach, the 50,000 query-document pairs in the ad hoc entry Brkly3 along with their accompanying relevance probability estimates were sorted in descending order of magnitude of estimate. Pairs for which human judgements of relevance-relatedness were unavailable were discarded; this left 22,352 sorted pairs for which both the system's probability estimates of relevance and the 'correct' binary judgements of relevance were available. This shorter list was divided into blocks of 1,000 pairs each -- the thousand pairs with the highest probability estimates, the thousand with the next highest, and so forth. Within each block the 'actual' probability was estimated as the proportion of the 1,000 pairs that had been judged to be relevance-related by humans. This was compared against the mean of all the system probability estimates in the block. For a well-calibrated system these figures should be approximately equal.

The results of the comparison are displayed in Table III. It can be seen that the system's probability predictions, while not wildly inaccurate, are generally somewhat higher than the actual proportions of relevant pairs. The same phenomenon of mild overestimation was observed when the runs Brkly4 and Brkly5 were tested for well-calibratedness in a similar way.

Since no systematic overestimation was observed when the calibration of the formula was originally tested against the learning data, it seems likely that the overestimation seen in the table is due mainly to the shift from learning data to test data. Naturally, predictive formulae that have been fine-tuned to a certain set of learning data will perform less well when applied to a new set of data to which they have not been fine-tuned. If this is indeed the root cause of the observed overestimation, it could perhaps be compensated for (at least to an extent sufficient for practical purposes) by the crude expedient of lowering all predicted probabilities to, say, around four fifths of their originally calculated values before displaying them to the users.

TABLE III: Calibration of Ad Hoc Relevance-Probability Estimates

Query-doc Pair Ranks	Mean System Probability Estimate	Proportion Actually Relevant
1 to 1,000	0.66	0.60
1,001 to 2,000	0.63	0.47
2,001 to 3,000	0.61	0.44
3,001 to 4,000	0.58	0.41
4,001 to 5,000	0.55	0.38
5,001 to 6,000	0.53	0.34
6,001 to 7,000	0.50	0.36
7,001 to 8,000	0.48	0.36
8,001 to 9,000	0.46	0.36
9,001 to 10,000	0.44	0.38
10,001 to 11,000	0.42	0.39
11,001 to 12,000	0.41	0.36
12,001 to 13,000	0.39	0.37
13,001 to 14,000	0.37	0.36
14,001 to 15,000	0.36	0.35
15,001 to 16,000	0.34	0.31
16,001 to 17,000	0.32	0.29
17,001 to 18,000	0.31	0.28
18,001 to 19,000	0.29	0.23
19,001 to 20,000	0.28	0.22
20,001 to 21,000	0.25	0.21
21,001 to 22,000	0.23	0.23
22,001 to 22,352	0.18	0.19

Computational Experience

The statistical program packages used in the course of the analysis included SAS, S, and BLSS. Of these,

SAS provides the most complete built-in diagnostics for logistic regression. BLSS was found to be especially convenient for interactive use in a UNIX environment, and ended up being the most heavily used.

The prototype retrieval system itself was implemented as a modification of the SMART system with SMART's vector-similarity subroutines replaced by the probabilistic computations of Eqs. (3) and (5). For the runs Brkly3 and Brkly4, which used only Eq. (3), only minimal modifications of SMART were needed, and at run time the retrieval efficiency remained essentially the same as for the unmodified SMART system. This demonstrates that probabilistic retrieval need be no more cumbersome computationally than the vector processing alternatives. For Brkly5, which used Eq. (5), the modifications were somewhat more extensive and retrieval took about 20% longer.

Retrieval Effectiveness

The Berkeley system achieved an average precision over all documents (an '11-point average') of 32.7% for the ad hoc retrieval run Brkly3, and 29.0% and 35.4% respectively for the routing runs Brkly4 and Brkly5. The distinct improvement in effectiveness of Brkly5 over Brkly4 suggests that in routing retrieval the use of frequency information about individual query stems is worth while.

At the '0 recall level' a precision of 84.7%, the highest recorded at the conference, was achieved in the ad hoc run. The high effectiveness of the Berkeley system for the first few retrieved documents may be explainable in terms of the practice, mentioned earlier, of redoing the regression analysis for the highest-ranked 500 documents for each query. This technique ensures an especially good regression fit for the query-document pairs that are especially likely to be relevant, thus emphasizing good performance near the top of the ranking where it is most important.

The generally high retrieval effectiveness of the Berkeley system should be interpreted in the light of the fact that the system probably uses less evidence -- that is, fewer retrieval clues -- than any of the other high-performing TREC-2 systems. In fact, the only clues used were the frequency characteristics of single stems (not even phrases were included). What this suggests is that the underlying probabilistic logic may have the capacity to exploit exceptionally fully whatever clues may be available.

Summary and Conclusions

The Berkeley design approach is based on a probabilistic model derived from the linked dependence assumption. The variables of the probability-ranking retrieval equation and their coefficients are determined by logistic regression on a judgement sample. Though the model is hospitable to the utilization of other kinds of evidence, in this particular investigation the only variables used were optimally relativized frequencies (ORF's) of match stems.

The approach was found to have the following advantages:

1. *Experimental Efficiency.* Since the numeric coefficients in a regression equation are determined simultaneously in one computation, trial-and-error experimentation involving the evaluation of retrieval output to optimize parameters is largely avoidable.
2. *Computational Simplicity.* For ad hoc retrieval and routing retrieval that does not involve individual stem statistics, the computational simplicity and efficiency achieved by the model at run time are comparable to that of simple vector processing retrieval models. For routing retrieval that exploits individual stem frequencies the programming is somewhat more complicated and runs slightly slower.
3. *Effective Retrieval.* The level of retrieval effectiveness as measured by precision and recall is high relative to the simple clue-types used.
4. *Potential for Well-Calibrated Probability Estimates.* In-the-ballpark estimates of document relevance probabilities suitable for output display would appear to be within reach.

Acknowledgements

We are indebted to Ray Larson and Chris Plaunt for helpful systems advice, as well as to the several colleagues already acknowledged in our Trec-1 report. The work stations used for the experiment were supplied by the Sequoia 2000 project at the University of California, a project principally funded by the Digital Equipment Corporation. A DARPA grant supported the programming effort.

References

Cooper, W. S. (1991). Inconsistencies and Misnomers in Probabilistic IR. *Proceedings Fourteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* Chicago, 57-62.

Cooper, W. S.; Dabney, D.; Gey, F. (1992). Probabilistic retrieval based on staged logistic regression. *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* Copenhagen, 198-210.

Cooper, W. S.; Gey, F. C.; Chen, A. (1993). Probabilistic retrieval in the TIPSTER Collections: An Application of Staged Logistic Regression. *The First Text REtrieval Conference (TREC-1).* NIST Special Publication 500-207. Washington, 73-88.

Fuhr, N.; Buckley, C. (1993). Optimizing Document Indexing and Search Term Weighting Based on Probabilistic Models. *The First Text REtrieval Conference (TREC-1).* NIST Special Publication 500-207. Washington, 89-100.

Fuhr, N.; Buckley, C. (1991). A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems,* 9, 223-248.

Gey, F. C. (1993). *Probabilistic Dependence and Logistic Inference in Information Retrieval* Ph.D. Dissertation, Univ. of California, Berkeley, California.

Good, I. J. (1965). *The estimation of probabilities; An essay on modern Bayesian methods.* Cambridge, Mass.; M.I.T. Press.

Hosmer, D. W.; Lemeshow, S. (1989). *Applied Logistic Regression.* New York: Wiley.

Robertson, S. E.; Maron, M. E.; Cooper, W. S. (1982). Probability of relevance: A unification of two competing models for document retrieval. *Information Technology: Research and Development* 1, 1-21.

Probabilistic Learning Approaches for Indexing and Retrieval with the TREC-2 Collection

Norbert Fuhr, Ulrich Pfeifer, Christoph Bremkamp, Michael Pollmann
University of Dortmund, Germany
Chris Buckley
Cornell University

Abstract

In this paper, we describe the application of probabilistic models for indexing and retrieval with the TREC-2 collection. This database consists of about a million documents (2 gigabytes of data) and 100 queries (50 routing and 50 adhoc topics). For document indexing, we use a description-oriented approach which exploits relevance feedback data in order to produce a probabilistic indexing with single terms as well as with phrases. With the adhoc queries, we present a new query term weighting method based on a training sample of other queries. For the routing queries, the RPI model is applied which combines probabilistic indexing with query term weighting based on query-specific feedback data. The experimental results of our approach show very good performance for both types of queries.

1 Introduction

The good TREC-1 results of our group described in [Fuhr & Buckley 93] have confirmed the general concept of probabilistic retrieval as a learning approach. In this paper, we describe some improvements of the indexing and retrieval procedures. For that, we first give a brief outline of the document indexing procedure which is based on description-oriented indexing in combination with polynomial regression. Section 3 describes query term weighting for adhoc queries, where we have developed a new learning method based on a training sample of other queries and corresponding relevance judgements. In section 4, the construction of the routing queries is presented, which is based on the probabilistic RPI retrieval model for query-specific feedback data. In the final conclusions, we suggest some further improvements of our method.

2 Document indexing

The task of probabilistic document indexing can be described as follows (see [Fuhr & Buckley 91] for more details): Let d_m denote a document, t_i a term and R the fact that a query-document pair is judged relevant, then $P(R|t_i, d_m)$ denotes the probability that document d_m will be judged relevant w.r.t. an arbitrary query that contains term t_i. Since these weights can hardly be estimated directly, we use the description-oriented indexing approach. Here term-document pairs (t_i, d_m) are mapped onto so-called *relevance descriptions* $\vec{x}(t_i, d_m)$. The elements x_i of the relevance description contain values of features of t_i, d_m and their relationship, like e.g.

tf	within-document frequency (wdf) of t_i,
logidf	= log(inverse document frequency),
lognumterms	= log(number of different terms in d_m),
imaxtf	= 1/(maximum wdf of a term in d_m)
is_single	=1, if term is a single word, =0 otherwise
is_phrase	=1, if term is a phrase, =0 otherwise.

(As phrases, we considered all adjacent non-stopwords that occurred at least 25 times in the $D_1 + D_2$ (training) document set.)

Based on these relevance descriptions, we estimate the probability $P(R|\vec{x}(t_i, d_m))$ that an arbitrary term-document pair having relevance description $\vec{x}$ will be involved in a relevant query-document relationship. This probability is estimated by a so-called *indexing function* $u(\vec{x})$. Different regression methods or probabilistic classification algorithms can serve as indexing function.

For our retrieval runs submitted to TREC-2, we used polynomial regression for developing an indexing function of the form

$$u(\vec{x}) = \vec{b} \cdot \vec{v}(\vec{x}), \tag{1}$$

where the components of $\vec{v}(\vec{x})$ are products of elements of $\vec{x}$.

The indexing function actually used has the form

$$
\begin{aligned}
u(\vec{x}) = \\
& b_0 & & & & & + \\
& b_1 \cdot is\_single & \cdot tf & \cdot logidf & \cdot imaxtf & + \\
& b_2 \cdot is\_single & \cdot tf & \cdot imaxtf & & + \\
& b_3 \cdot is\_single & \cdot logidf & & & + \\
& b_4 \cdot lognumterms & \cdot imaxtf & & & + \\
& b_5 \cdot is\_phrase & \cdot tf & \cdot logidf & \cdot imaxtf & + \\
& b_6 \cdot is\_phrase & \cdot tf & \cdot imaxtf & & + \\
& b_7 \cdot is\_phrase & \cdot logidf. & & &
\end{aligned}
$$

The coefficient vector $\vec{b}$ is computed based on a training sample of query-document-pairs with relevance judgements.

Since polynomial functions may yield results outside the interval $[0, 1]$, these values were mapped onto the corresponding boundaries of this interval.

For each phrase occurring in a document, indexing weights for the phrase as well as for its two components (as single words) were computed.

There are two major problems with this approach which we are investigating currently:

1. Which factors should be used for defining the indexing functions? We are developing a tool that supports a statistical analysis of single factors for this purpose.
2. What is the best type of indexing function? Previous experiments have suggested that regression methods outperform other probabilistic classification methods. As a reasonable alternative to polynomial regression, logistic regression seems to offer some advantages (see also [Fuhr & Pfeifer 94]). As a major benefit, logistic functions yield only values between 0 and 1, so there is no problem with outliers. We are performing experiments with logistic regression and compare the results to those based on polynomial regression.

3 Query term weighting for ad-hoc queries

3.1 Theoretical background

The basis of our query term weighting scheme for ad-hoc queries is the linear utility-theoretic retrieval function described in [Wong & Yao 89]. Let q_k^T denote the set of terms occurring in the query, and u_{im} the indexing weight $u(\vec{x}(t_i, d_m))$ (with $u_{im} = 0$ for terms t_i not occurring in d_m). If c_{ik} gives the utility of term t_i for the actual query q_k, then the utility of document d_m w.r.t. query q_k can be computed by the retrieval function

$$\varrho(q_k, d_m) = \sum_{t_i \in q_k^T} c_{ik} \cdot u_{im}. \tag{2}$$

For the estimation of the utility weights c_{ik}, we applied two different methods.

As a heuristic approach, we used tf weights (the number of occurrences of the term t_i in the query), which had shown good results in the experiments described in [Fuhr & Buckley 91].

As a second method, we applied linear regression to this problem. Based on the concept of polynomial retrieval functions as described in [Fuhr 89b], one can estimate the probability of relevance of q_k w.r.t. d_m by the formula

$$P(R|q_k, d_m) \approx \sum_{t_i \in q_k^T} c_{ik} \cdot u_{im}. \tag{3}$$

If we had relevance feedback data for the specific query (as is the case for the routing queries), this function could be used directly for regression. For the ad-hoc-queries, however, we have only feedback information about other queries. For this reason, we regard *query features* instead of specific queries. This can be done by considering for each query term the same features as described before in the context of document indexing. Assume that we have a set of features $\{f_0, f_1, \ldots, f_l\}$ and that x_{ji} denotes the value of feature f_j for term t_i. Then we assume that query term weight c_{ik} can be estimated by linear regression according to the formula

$$c_{ik} = \frac{1}{|q_k^T|} \sum_{j=0}^{l} a_j x_{ji}. \tag{4}$$

Here the factor $|q_k^T|$ serves for the purpose of normalization across different queries, since queries with a larger number of terms tend to yield higher retrieval status values with formula 2. The factors a_j are the coefficients that are to be derived by means of regression. Now we have the problem that regression cannot be applied to eqn 4, since we do not observe c_{ik} values directly. Instead, we observe relevance judgements. This leads us back to the polynomial retrieval function 3, where we substitute eqn 4 for c_{ik}:

$$
\begin{aligned}
P(R|q_k, d_m) &\approx \sum_{t_i \in q_k^T} \frac{1}{|q_k^T|} \left(\sum_{j=0}^{l} a_j x_{ji} \right) u_{im} \\
&= \sum_{j=0}^{l} a_j \sum_{t_i \in q_k^T} \frac{1}{|q_k^T|} x_{ji} u_{im} \\
&= \sum_{j=0}^{l} a_j y_j \qquad (5)
\end{aligned}
$$

with

$$y_j = \sum_{t_i \in q_k^T} \frac{1}{|q_k^T|} x_{ji} u_{im} \tag{6}$$

Equation 5 shows that we can apply linear regression of the form $P(R|q_k, d_m) \approx \vec{a} \cdot \vec{y}$ to a training sample

of query-document pairs with relevance judgements in order to determine the coefficients a_j. The values y_j can be computed from the number of query terms, the values of the query term features and the document indexing weights.

For the experiments, the following parameters were considered as query term features:

x_0 =1 (constant)

$x_1 = tf$ (within-query-frequency)

$x_2 = \log tf$

$x_3 = tf \cdot idf$

$x_4 = is\_phrase$

$x_5 = in\_title$ (=1, if term occurs in query title, =0 otherwise)

For most of our experiments, we only used the parameter vector $\vec{x} = (x_0, \ldots, x_4)^T$. The full vector is denoted as $\vec{x}'$. Below, we call this query term weighting method *reg*.

This method is compared with the standard SMART weighting schemes:

nnn: $c_{ik} = tf$

ntc: $c_{ik} = tf \cdot idf$

lnc: $c_{ik} = (1 + \log tf)$

ltc: $c_{ik} = (1 + \log tf) \cdot idf$

3.2 Experiments

In order to have three different samples for learning and/or testing purposes, we used the following combinations of query sets and document sets as samples: Q3/D12 was used as training sample for the *reg* method, and both Q1/D12 and Q2/D3 were used for testing. As evaluation measure, we consider the 11-point average of precision (i.e., the average of the precision values at $0.0, 0.1, \ldots, 1.0$ recall).

	sample	
QTW	Q1/D12	Q2/D3
nnn	0.2303	
ntc	0.2754	
lnc	0.2291	0.2601
ltc	0.2826	0.2783
reg	0.2698	0.2678

Table 1: Global results for single words

First, we considered single words only. Table 1 shows the results of the different query term weighting (QTW) methods. First, it should be noted that the *ntc* and *ltc* methods perform better than *nnn* and *lnc*. This finding is somewhat different from the results presented in [Fuhr & Buckley 91], where the *nnn* weighting scheme gave us better results than the *ntc* method for lsp indexing. However, in the earlier experiments, we used only fairly small databases, and the queries also were much shorter than in the TREC collection. These facts may account for the different results.

		test sample		
run	learning sample	features	Q1/D12	Q2/D3
1	every doc.	$\vec{x}$	0.2698	0.2678
2	every 100. doc.	$\vec{x}$	0.2700	0.2678
3	every 1000. doc.	$\vec{x}$	0.2662	
4	judged docs only	$\vec{x}$	0.2635	0.2677
5	every doc.	$\vec{x}'$	0.2654	
6	every 100. doc.	$\vec{x}'$	0.2677	

Table 2: Variations of the *reg* learning sample and the query features

	run					
factor	1	2	3	4	5	6
constant	-3.05	-2.96	1.04	-20.80	2.47	-1.71
tf	-7.54	-10.67	-2.40	14.58	-4.32	14.11
$\log tf$	-9.01	-5.69	-5.08	-81.58	-1.48	-0.70
$tf \cdot idf$	5.84	7.00	1.63	8.72	1.81	7.70
is_phrase	-8.06	-19.23	-2.73	19.81	-2.97	20.39
in_title					-1.70	3.44

Table 3: Coefficients of query regression

In a second series of experiments, we varied the sample size and the set of features of the regression method (table 2). Besides using every document from the learning sample, we only considered every 100th and every 1000th document from the database, as well as only those documents for which there were explicit relevance judgements available. As the results show almost no differences, it seems to be sufficient to use only a small portion of the database as training sample in order to save computation time. The additional consideration of the occurrence of a term in the query title also did not effect the results. So query titles seem to be not very significant. It is an open question whether or not other parts of the queries are more significant, so that the consideration as an additional feature would affect retrieval quality.

The coefficients computed by the regression process for the second series of experiments are shown in table 3. It is obvious that the coeffients depend heavily on the choice of the training sample — so it is quite surprising that retrieval quality is not affected by this factor. The only coefficient which does not change its sign through all the runs is the one for the $tf \cdot idf$ factor. This seems to confirm the power of this factor. The other factors can be regarded as being only minor modifications of the $tf \cdot idf$ query term weight.

Overall, it must be noted that the regression method does not yield an improvement over the *ntc* and *ltc* methods. This seems to be surprising, since the regression is based on the same factors which also go into the

ntc and *ltc* formulas. However, a possible explanation could be the fact that the regression method tries to minimize the quadratic error for all the documents in the learning sample, but our evaluation measure considers at most the top ranking 1000 documents for each query; so regression might perform well for most of the documents from the database, but not for the top of the ranking list. There is some indication for this explanation, since regression yields always slightly better results at the high recall end.

α	result
0.00	0.3199
0.10	0.3707
0.15	0.3734
0.20	0.3700
0.25	0.3656
0.30	0.3610
0.50	0.3451
1.00	0.3147

Table 4: Effect of downweighting of phrases (sample Q2/D12)

As described before, in our indexing process, we consider phrases in addition to single words. This leads to the problem that when a phrase occurs in a document, we index the phrase in addition to the two single words forming the phrase. As a heuristic method for overcoming this problem, we introduced a factor for downweighting query term weights for phrases. That is, the actual query term weight of a phrase is $c'_{ik} = \alpha c_{ik}$, where c_{ik} is the result of the regression process. In order to derive a value for α, we performed a number of test runs with varying values (see table 4). Obviously, weighting factors between 0.1 and 0.3 gave the best results. For the official runs, we choose $\alpha = 0.15$.

		sample	
QTW	α	Q1/D12	Q2/D3
ltc	0.15	0.3192	0.3131
ltc	0.2	0.3220	0.3056
reg	0.15	0.3080	0.3062

Table 5: Results for single words and phrases

In table 5, this method is compared with the *ltc* formula, where we also choose a weighting factor for phrases which gave the best results. One can see that with the sample Q2/D3, the differences between the methods are smaller than on sample Q1/D12, but still *ltc* seems to perform slightly better.

Finally, we investigated another method for coping with phrases. For that, let us assume that we have binary query weights only. Now as an example, the single words t_1 and t_2 form a phrase t_3. For a query with phrase t_3 (and thus also with t_1 and t_2), a document d_m containig the phrase would yield $u_{1m} + u_{2m} + u_{3m}$ as value of the retrieval function, where the weights u_{im} are computed by the lsp method described before. In order to avoid the effect of counting the single words in addition to the phrase, we modified the original phrase weight as follows:

$$u'_{3m} = u_{3m} - u_{1m} - u_{2m}$$

and stored this value as phrase weight. Queries with the single words t_1 or t_2 are not affected by this modification. For the query with phrase t_3, however, the retrieval function now would yield the value $u_{1m} + u_{2m} + u'_{3m} = u_{3m}$, which is what we would like to get.

QTW	α	result
reg	0.00	0.2724
reg	1.00	0.2596
ntc	0.00	0.2754
ntc	0.15	0.3110
ntc	1.00	0.2524

Table 6: Results for the subtraction method (sample Q1/D12)

Table 6 shows the corresponding results ($\alpha = 0$ means that single words only are considered). In contrast to what we expected, we do not get an improvement over single words only when phrases are considered fully. The result for the *ntc* method shows that still phrases should be downweighted. Possibly, there may be an improvement with this method when we would use binary query term weights, but it is clear that other query term weighting methods mostly give better results.

3.3 Official runs

As document indexing method, we applied the description-oriented approach as described in section 2. In order to estimate the coefficients of the indexing function, we used the training sample Q12/D12, i.e. the query sets Q_1 and Q_2 in combination with the documents from D_1 and D_2.

Two runs with different query term weights were submitted. Run `dortL2` is based on the *nnn* method, i.e. *tf* weights. Run `dortQ2` uses *reg* query term weights. For performing the regression, we used the query sets Q_1 and Q_2 and a sample of 400,000 documents from D_1.

Table 7 shows the results for the two runs (Numbers in parentheses denote figures close to the best/worst results.). As expected, `dortQ2` yields better results than `dortL2`. The recall-precision curves (see figure 1) show that there is an improvement throughout the whole recall range. For precision average and precision at 1000 documents retrieved, run `dortQ2` performs very well, while precision at 100 documents retrieved is less good. This confirms our interpretation from above, saying that

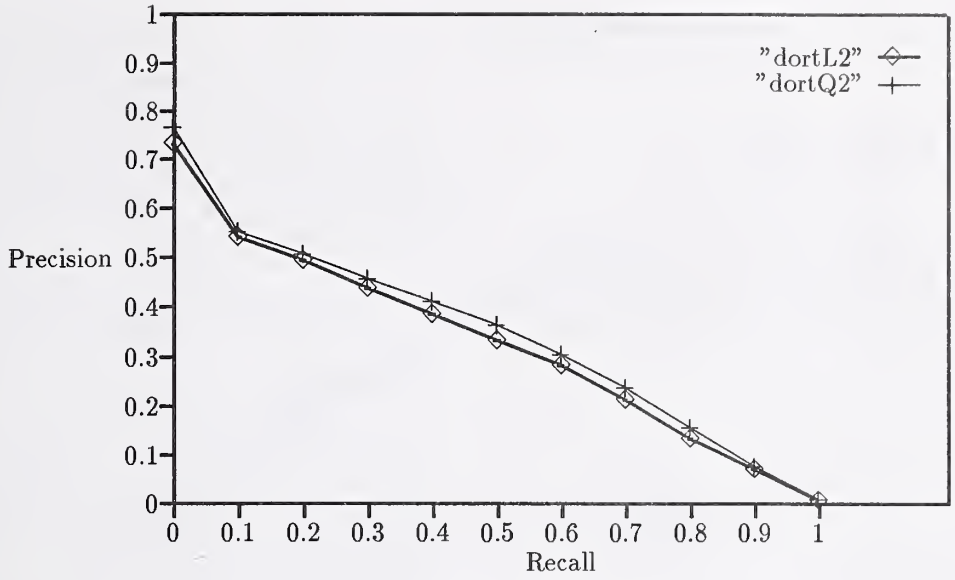


Figure 1: Recall-precision curves of ad-hoc runs

run query term weighting	dortL2 *nnn*	dortQ2 *reg*
average precision:		
Prec. Avg.	0.3151	0.3340
query-wise comparison with median:		
Prec. Avg.	37:2	45:4
Prec. @ 100 docs	35:11	34:7
Prec. @ 1000 docs	37:10	45:2
Best/worst results:		
Prec. Avg.	3/0	3(1)/0
Prec. @ 100 docs	3(2)/1	4(1)/0(2)
Prec. @ 1000 docs	6(1)/0	9(1)/0
dortL2 vs. dortQ2:		
Prec. Avg.	21:29	
Prec. @ 100 docs	22:24	
Prec. @ 1000 docs	17:29	

Table 7: Results for adhoc queries

regression optimizes the overall performance, but not necessarily the retrieval quality when only the top ranking documents are considered. With regard to the moderate results for the *reg* query term weighting method, the good performance of **dortQ2** obviously stems from the quality of our document indexing method.

4 Query term weighting for routing queries

4.1 Theoretical background

For the routing queries, the retrieval-with-probabilistic-indexing (RPI) model described in [Fuhr 89a] was applied. The corresponding retrieval function is based on the following parameters:

u_{im} indexing weight of term t_i in document d_m
D_k^R set of documents judged relevant for query q_k,
p_{ik} expectation of the indexing weight of term t_i in D_k^R,
D_k^N set of documents judged nonrelevant for query q_k,
r_{ik} expectation of the indexing weight of t_i in D_k^N.

The parameters p_{ik} and r_{ik} can be estimated based on relevance feedback data as follows:

$$p_{ik} = \frac{1}{D_k^R} \sum_{d_m \in D_k^R} u_{im}$$

$$q_{ik} = \frac{1}{D_k^N} \sum_{d_m \in D_k^N} u_{im}$$

Then the query term weight is computed by the formula

$$c_{ik} = \frac{p_{ik}(1 - r_{ik})}{r_{ik}(1 - p_{ik})} - 1$$

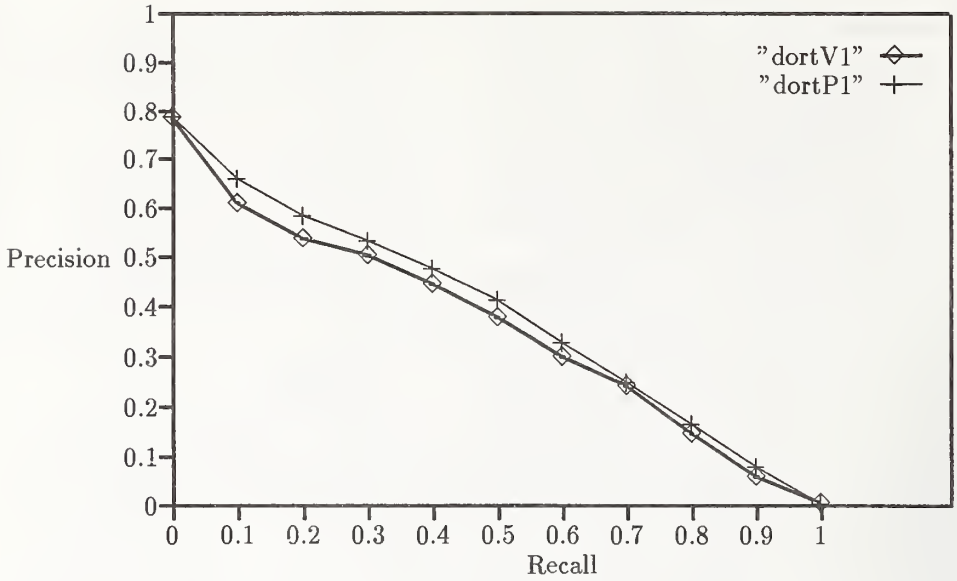


Figure 2: Recall-precision curves of routing runs

and the RPI retrieval function yields

$$\varrho(q_k, d_m) = \sum_{t_i \in q_k^T} \log(c_{ik} u_{im} + 1). \qquad (7)$$

4.2 Experiments

In principle, the RPI formula can be applied with or without query expansion. For our experiments in TREC1, we did not use any query expansion. The final results showed that this was reasonable, mainly with respect to the small amount of relevance feedback data available. In contrast, for TREC2 there were about 2000 relevance judgements per query, so there was clearly enough training data for applying query expansion methods.

As basic criterion for selecting the expansion terms, we considered the number of relevant documents in which a term ocurs, which gave us a ranking of candidates; document indexing weights were considered for tie-breaking. Then we varied the the number of terms which are added to the original query.

expansion	result
0	0.2909
10	0.3047
30	0.3035
50	0.3002
100	0.2832

Table 8: Effect of number of expansion terms

In a first series of experiments, we considered single word only. We used Q2/D1 (*lsp* document indexing) as training sample and Q2/D2 (*ltc* indexing) as test sample. As can be seen from table 8, query expansion clearly improves retrieval quality, but only for a limited number of expansion terms. For larger numbers, we get worse results. This effect seems to be due to parameter estimation problems.

	expansion		
phraseweight	single w.	phrases	result
0.5	0	0	0.3476
0.5	20	0	0.3713
1.0	20	0	0.3730
0.5	20	10	0.3728
1.0	20	10	0.3605
0.5	30	10	0.3729
1.0	30	10	0.3626

Table 9: Query expansion with phrases

In a second series of experiments, we looked at the combination of single words and phrases. These experiments were performed as retrospective runs, with Q2/D12 as training sample and Q2/D2 as test sample (both with *ltc* document indexing). For the number of expansion terms, we treated single words and phrases separately. Furthermore, similar to the adhoc runs, we used an additional factor for downweighting the query term weights of phrases. The different parameter combinations tested and the corresponding results are given in table 9. Obviously, phrases as expansion terms gave no improvement, so we decided to have only single words as expansion terms (but the phrases from the original query still are used for retrieval). Furthermore, the retrieval quality reaches its optimum at about 20 terms.

4.3 Official runs

Two different runs were submitted for the routing queries, both based on the RPI model.
Run `dortP1` uses the same document indexing function as for the adhoc queries. Query terms were weighted according to the RPI formula. In addition, each query was expanded by 20 single words. Phrases were not downweighted.
Run `dortV1` is based on *ltc* document indexing. Here no query expansion took place.

run	`dortV1`	`dortP1`
document indexing	*ltc*	*lsp*
query expansion	none	20 terms
average precision:		
Prec. Avg.	0.3516	0.3800
query-wise comparison with median:		
Prec. Avg.	38:10	46:4
Prec. @ 100 docs	31:11	40:5
Prec. @ 1000 docs	32:9	37:7
Best/worst results:		
Prec. Avg.	1/0	4(2)/0
Prec. @ 100 docs	3(3)/1(1)	7(5)/1(1)
Prec. @ 1000 docs	6(2)/0(1)	10(2)/0(1)
`dortV1` vs. `dortP1`:		
Prec. Avg.	10:39	
Prec. @ 100 docs	9:27	
Prec. @ 1000 docs	7:33	

Table 10: Results for routing queries

Table 10 shows the results for the two runs. The recall-precision curves are given in figure 2. Again, the results confirm our expectations that LSP indexing and query expansion yields better results.

5 Conclusions and outlook

The experiments described in this paper have shown that probabilistic learning approaches can be applied successfully to different types of indexing and retrieval. For the ad-hoc queries, there seems to be still room for further improvement in the low recall range. In order to increase precision, a passage-wise comparison of query and document text should be performed. For this purpose, polynomial retrieval functions could be applied.
In the case of the routing queries, we first have to investigate methods for parameter estimation in combination with query expansion. However, with the large number of feedback documents given for this task, other types of retrieval models may be more suitable, e.g. query-specific polynomial retrieval functions.
Finally, it should be emphasized that we still use rather simple forms of text analysis. Since our methods are flexible enough to work with more sophisticated analysis procedures, this combination seems to be a prospective area of research.

A Operational details of runs

A.1 Basic Algorithms

The algorithm A to find the coefficient vector $\vec{a}$ for the ad-hoc query term weights can be given as follows:

Algorithm A

1 For each query document pair $(q_k, d_m) \in (Q_1 \cup Q_2) \times D_s$ with D_s being a sample from $(D_1 \cup D_2)$ do
  1.1 determine the relevance value r_{km} of the document d_m with respect to the query q_k.
  1.2 For each term t_i occuring in q_k do
    1.2.1 determine the feature vector $\vec{x_i}$ and the indexing weight u_{im} of the term t_i w.r.t. to document d_m.
  1.3 For each feature j of the feature vectors $\vec{x}$ compute the value of y_j looping over the terms of the query.
  1.4 Add vector $\vec{x}$ and relevance value r_{km} to the least squares matrix.
2 Solve the least squares matrix to find the coefficient vector $\vec{a}$

The algorithm B to find the coefficient vector $\vec{b}$ for the document indexing is sketched here:

Algorithm B

1 Index $D_1 \cup D_2$ (the learning document set) and $Q_1 \cup Q_2$ (the learning query set).

2 For each document $d \in D_1 \cup D_2$
  2.1 For each $q \in Q_1 \cup Q_2$
    2.1.1 Determine the relevance value r of d to q
    2.1.2 For each term t in common between q^T (set of query terms) and d^T (set of document terms)
      2.1.2.1 Find values of the elements of the relevance description involved in this run and add values plus relevance information to the least squares matrix being constructed
3 Solve the least squares matrix to find the coefficient vector $\vec{b}$

The algorithm C to index a document set D can now be given as:

Algorithm C

1 For each document $d \in D$
  1.1 For each term $t \in d^T$
    1.1.1 Find values of the relevance description $\vec{x}(t,d)$ involved in run.
    1.1.2 Give t the weight $\vec{b} \cdot \vec{v}(\vec{x}(t,d))$
  1.1 Add d to the inverted file.

A.2 Ad-hoc runs

The algorithm D is used for indexing and retrieval for the ad-hoc runs. Steps numbered with a trailing "A" apply only for run `dortQ2`, steps with trailing "B" only to run `dortL2`.

Algorithm D

1 Run algorithm B to determine the coefficient vector $\vec{b}$ for document indexing.
1A Run algorithm A to determine the coefficient vector $\vec{a}$ for query indexing.
2 Call algorithm C for document set $D_1 \cup D_2$
3 For each query $q_k \in Q_3$ do
  3.1 For each term t_i occuring in q_k do
    3.1.1A Determine the feature vector x_{ik} and compute the query term weight c_{ik} by multiplying it to $\vec{a}$.
    3.1.1B Weight t_i w.r.t. q_k (test query set) with tf weights (nnn variant). Phrases where downweighted by multiplying the weights with $\alpha = 0.15$.
  3.2 Run an inner product inverted file similarity match of $\vec{c_k}$ against the inverted file formed in step 2, retrieving the top 1000.

A.3 Routing runs

Algorithm E is used for indexing and retrieval for the routing runs. Steps numbered with a trailing "A" apply only for run `dortP1`, steps with trailing "B" only to run `dortV1`..

Algorithm E

1A Index query set Q_2 and document set $D_1 \cup D_2$ with $tf \cdot idf$ weights.
1B Index query set Q_2 and document set $D_1 \cup D_2$ by calling algorithm C
2 For each query $q \in Q_2$
  2.1 For each term $t \in q^T$ (set of query terms)
    2.1.1 Reweight term t using the RPI relevance weighting formula andthe relevance information supplied.
3A Index document set D_3 by calling algorithm C.
3B Index document set D_3 with $tf \cdot idf$ weights.
Note that the collection frequency information used was derived from occurrences in $D_1 \cup D_2$ only (in actual routing the collection frequencies within D_3 would not be known).
4 Run the reweighted queries of Q_2 (step 2) against the inverted file (step 3), returning the top 1000 documents for each query.

References

Fuhr, N.; Buckley, C. (1991). A Probabilistic Learning Approach for Document Indexing. *ACM Transactions on Information Systems 9(3)*, pages 223–248.

Fuhr, N.; Buckley, C. (1993). Optimizing Document Indexing and Search Term Weighting Based on Probabilistic Models. In: Harman, D. (ed.): *The First Text REtrieval Conference (TREC-1)*, pages 89–100. National Institute of Standards and Technology Special Publication 500-207, Gaithersburg, Md. 20899.

Fuhr, N.; Pfeifer, U. (1994). Probabilistic Information Retrieval as Combination of Abstraction, Inductive Learning and Probabilistic Assumptions. *ACM Transactions on Information Systems 12(1).*

Fuhr, N. (1989a). Models for Retrieval with Probabilistic Indexing. *Information Processing and Management 25(1)*, pages 55–72.

Fuhr, N. (1989b). Optimum Polynomial Retrieval Functions Based on the Probability Ranking Principle. *ACM Transactions on Information Systems 7(3)*, pages 183–204.

Wong, S.; Yao, Y. (1989). A Probability Distribution Model for Information Retrieval. *Information Processing and Management 25(1)*, pages 39–53.

TREC-2 Routing and Ad-Hoc Retrieval Evaluation using the INQUERY System

Bruce Croft, James Callan, and John Broglio
Computer Science Department
University of Massachusetts
Amherst, MA. 01003

1 Project Goals

The ARPA TIPSTER project, which is the source of the data and funding for TREC, has involved four sites in the area of text retrieval and routing. The TIPSTER project in the Information Retrieval Laboratory of the Computer Science Department, University of Massachusetts, Amherst (which includes MCC as a subcontractor), has focused on the following goals:

- Improving the effectiveness of information retrieval techniques for large, full-text databases,
- Improving the effectiveness of routing techniques appropriate for long-term information needs, and
- Demonstrating the effectiveness of these retrieval and routing techniques for Japanese full text databases [4].

Our general approach to achieving these goals has been to use improved representations of text and information needs in the framework of a new model of retrieval. This model uses Bayesian networks to describe how text and queries should be used to identify relevant documents [6, 3, 7]. Retrieval (and routing) is viewed as a probabilistic inference process which compares text representations based on different forms of linguistic and statistical evidence to representations of information needs based on similar evidence from natural language queries and user interaction. Learning techniques are used to modify the initial queries both for short-term and long-term information needs (relevance feedback and routing, respectively).

This approach (generally known as the inference net model and implemented in the INQUERY system) emphasizes retrieval based on combination of evidence. Different text representations (such as words, phrases, paragraphs, or manually assigned keywords) and different versions of the query (such as natural language and Boolean) can be combined in a consistent probabilistic framework. This type of "data fusion" has been known to be effective in the information retrieval context for a number of years, and was one of the primary motivations for developing the inference net approach.

Another feature of the inference net approach is the ability to capture complex structure in the network representing the information need (i.e. the query). A practical consequence

of this is that complex Boolean queries can be evaluated as easily as natural language queries and produce ranked output. It is also possible to represent "rule-based" or "concept-based" queries in the same probabilistic framework. This has led to us concentrating on automatic analysis of queries and techniques for enhancing queries rather than on in-depth analysis of the documents in the database. In general, it is more effective (as well as efficient) to analyze short query texts than millions of document texts. The results of the query analysis are represented in the INQUERY query language which contains a number of operators, such as #SUM, #AND, #OR, #NOT, #PHRASE, and #SYN. These operators implement different methods of combining evidence and describing concepts.

Some of the specific research issues we are addressing are morphological analysis in English and Japanese, word sense disambiguation in English, the use of phrases and other syntactic structure in English and Japanese, the use of special purpose recognizers (for example, company, country and people name recognizers) in representing documents and queries, analyzing natural language queries to build structured representations of information needs, learning techniques appropriate for routing and structured queries, techniques for acquiring domain knowledge by corpus analysis, and probability estimation techniques for indexing.

The first TREC evaluation and the two previous TIPSTER evaluations have made it clear that a lot remains to be learned about retrieval in large, full-text databases based on complex information needs. Issues as phrases, relevance feedback, and probability estimation have proven to be quite difficult in such environments. On the other hand, the effectiveness levels achieved have been quite good. The experiments done in the TREC-2 evaluation, together with the 24 month TIPSTER evaluation which followed it, were designed to improve our understanding about which IR techniques work and why.

2 System Description

The document retrieval and routing system that has been developed on the basis of the inference net model is called INQUERY [2]. The main processes in INQUERY are **document indexing**, **query processing**, **query evaluation** and **relevance feedback**.

In the document indexing process, documents are parsed and index terms representing the content of documents are identified. INQUERY supports a variety of indexing techniques including simple word-based indexing, indexing based on part-of-speech tagging and phrase identification, and indexing by domain-dependent features such as company names, dates, locations, etc. The last type of indexing is a first step towards integrating detection and extraction systems.

In more detail, the document structure is used to identify which parts will be used for indexing. The first step of this process is then to scan for word tokens. Most types of words (including numbers) are indexed, although a stopword list is used to remove very common words. Stopwords can be indexed, however, if they are capitalized (but not at the start of sentences) or joined with other words (e.g. "the The-1 system"). Words are then stemmed to conflate variants. Although the Porter stemmer was used for the TREC-2

experiments, we have developed a new stemming algorithm that has a number of advantages for operational systems. A number of recognizers written in *flex* are then used to identify objects such as company names and mark their presence in the document using "meta" index terms. A company name such as IBM in the text, for example, will result in a meta term #COMPANY being recorded at that position in the text. The use of these meta terms extends the range of queries that can be specified. This completes the usual processing for document text.

The document indexing process also involves building the compressed inverted files that are necessary for efficient performance with very large databases. Since positional information is stored, overhead rates are typically about 40% of the original database size.

The query processing process involves a series of steps to identify the important concepts and structure describing a user's information need. INQUERY is unique in that it can represent and use complex structured descriptions in a probabilistic framework. Many of the steps in query processing are the same as those done in document indexing. In addition, a part-of-speech tagger is to used to identify candidate search phrases. Domain-dependent features are recognized and meta-terms inserted into the query representation. The relative importance of query concepts is also estimated, and relationships between concepts are suggested based on simple grammar rules. An evaluation of some of the query processing techniques is presented in [1].

INQUERY also has the capability of expanding the query using relationships between concepts found by either using manually specified domain knowledge in the form of a simple thesaurus or by corpus analysis. The WORDFINDER system is a version of INQUERY that retrieves concepts that are related to the query. WORDFINDER is constructed by identifying noun groups in the text and representing them by the words that are closely associated with them (i.e. occur in the same text windows). Concept "documents" are then stored in INQUERY. This technique of query expansion was not tested in TREC-2.

The query evaluation process uses the inverted files and the query represented as an inference net to produce a document ranking. The evaluation involves probabilistic inference based on the operators defined in the INQUERY language. These operators define new concepts and how to calculate the belief in those concepts using linguistic and statistical evidence. We are constantly experimenting with and refining these operators (for example, the operator defining a phrase-based concept) in order to improve retrieval performance.

The relevance feedback process uses information from user evaluations of retrieved documents to modify the original query in detection or routing environments. The INQUERY system, because it can represent structured queries, supports a wide range of learning techniques for query modification [5]. In general, new words and phrases are identified in the sample of relevant documents. These are added to the original query and all the terms in the query are then reweighted. With the amount of relevance information available in TIPSTER, relatively simple automatic techniques appear to produce good levels of effectiveness. We are also investigating the effect of using more limited information and more complex learning techniques, such as neural networks.

3 Query Processing

In order to clarify the query processing done for the TREC and TIPSTER experiments with INQUERY, the following sections give more detailed descriptions.

There are two main kinds of query styles: a natural language query and a keyword or key concept query. For example, the <desc> and <narr> fields of a TIPSTER query represent natural language queries of varying levels of abstraction. The <con>, <title> and <fac> fields represent key concepts in the query. The main difference between the two types of processing is that the key concept query has more controlled information. The phrasing and emphasis are already given and do not have to be conjectured from the language structure. It is valuable to discover how to treat both styles of query, because a good user interface will make it easy for a user to input both styles. For example, a user may enter a prose query and then highlight the important words and phrases in the query in some convenient manner. These highlighted words would then be treated as key concepts in the query processing.

3.1 Prose query processing

Natural language query fields are tagged for syntactic category by a part-of-speech (POS) tagger. Currently we use the tagger developed by Ken Church. We have developed our own POS tagger, and we expect to begin using it in the fall of 1993. There are some pre-tagging and post-tagging "housekeeping" operations, such as removing parentheses. (The current version of INQUERY does not permit parentheses except as part of an operator, and we do not yet make any inferences from the presence of parentheses during the text processing.) Additionally, we change operator phrases to single words in order to simplify later processing. An example of this simplification is replacing the phrase *in order to* with the infinitive particle *to* or replacing *with respect to* with the word *regarding*. The goal of this replacement is to remove phrases which resemble noun phrases syntactically but which are really syntactic operators (e.g., phrasal prepositions) with no substantive content. At this stage, *stop phrases* are also removed.

3.1.1 Noun and adjective phrase capture: orthographic and syntactic clues.

When the text is tagged and the potentially irrelevant material has been removed, syntactically-based noun group capture is performed. Certain kinds of noun phrase patterns are enfolded in a #PHRASE operator:

1. A noun phrase which contains more than one modifying adjective and noun is enclosed in a #PHRASE operator;

2. A head noun with no premodifiers and followed by a prepositional phrase is enclosed in a #PHRASE operator with the head noun of the prepositional phrase;

3.1.2 Constraint capture

All text in the query is searched for constraint expressions. Among these expressions are the words *company*, *not U. S.* or a restriction in the nationality section of the <fac> field to U.S. or other nationality. A restriction to U.S. nationality as the area of interest is implemented by penalizing documents for references to foreign countries. A restriction to other nationalities is implemented by repeating that country as a term. This asymmetry depends on the fact that the document collection is drawn solely from U.S. sources, and therefore the U.S., as the default area of interest, is rarely referred to unless the government or foreign policy implementation is under discussion.

There is some recognition of simple time expressions, such as *since 1984* which are expanded to the set of years which might be intended by the phrase in question.

Countries are recognized as such and are handled so that expressions like *South Africa* are phrased as `#1( south africa )` even when they appear in the middle of a larger group of capitalized words. In addition, proper names such as country names are moved out of the scope of `#PHRASE` operators, since it generally increases the effectiveness of a `#PHRASE` to reduce the number of words in it. Nationality constraints can better be maintained within the scope of the larger and more tolerant `#SUM` operator. For example the phrase

"import ban on South African diamonds"

becomes by stages,

```
#PHRASE (import ban on #SYN (#1 (south african) #1 (south africa)) diamonds)
```

and finally

```
#SUM (#SYN (#1(south african) #1(south africa))
#PHRASE(import ban on diamonds)).
```

3.2 Key concept query processing

Key concept query processing is different from prose query processing since the concept separation provided by the user can presumably be trusted. Instead of using a part-of-speech tagger, we rely on comma delimitation of concepts, and #PHRASE the words found between each pair of delimiters.

Additionally, if any constraints were found anywhere else in the query, e.g., a mention of the word *company* or an exclusionary geographical constraint (e.g., *not USA* or *only USA*), the query will be modified according to these constraints. For example,

only USA ⇒ `#NOT (#FOREIGNCOUNTRY )`

and

not USA ⇒ `#NOT ( #USA )`.

If the word company is found in a query, then a second copy of the key concepts (the <con> field), is produced where each item in the field appears in an unordered window operator with the special concept #COMPANY. For example, if the word *South Africa* appears as a key concept (and *company* appears somewhere in the query), then the pre-processor would produce the term `#UW50( #COMPANY #1( south africa))` which would match any document which had a company name within fifty words of *South Africa.*

4 The TREC Experiments

Four experiments were submitted to the TREC evaluation, two "ad-hoc" and two "routing". In these experiments, we emphasized automatic query processing and automatic feedback algorithms for routing. The following is a summary:

- AdHoc: topics 101-150 against TIPSTER volumes 1 and 2.

  INQ001 Created automatically from TIPSTER topics. Contains phrases. Details of query processing used are described below.

  INQ002 INQ001 queries, modified manually. Modifications restricted to eliminating words and phrases, and adding paragraph-level operators around existing words and phrases. The method for doing this was done somewhat differently than last year's TREC conference, as discussed below.

- Routing: topics 51-100 against TIPSTER volume 3.

  INQ003 Created automatically from TIPSTER topics and relevance judgements from Volumes 1 and 2. Baseline queries (from a previous TIPSTER evaluation) were modified by reweighting and adding single-word terms. The term weighting and selection function used was df.idf, as described in [5]. Only the top 120 relevant documents found by INQUERY were used for feedback, and 30 terms were added to each query.

  INQ004 Formed by combining (using the #SUM operator) INQ001 queries and INQRYP queries (used in TIPSTER 18 month evaluation). The INQRYP queries were produced automatically and then modified manually. Modifications restricted to eliminating words and phrases, and adding paragraph-level operators around existing words and phrases.

Query Type	**Average Precision**			
	5 Docs	**30 Docs**	**100 Docs**	**11-Pt Avg**
INQ001	.62	.57	.49	.36
INQ002	.60 (−2.6%)	.59 (+3.5%)	.51 (+4.1%)	.36 (0%)

Table 1: Results for Adhoc queries

Table 1 gives the results for the adhoc queries. These show that there is little difference in effectiveness between the automatically processed queries and the semi-automatically processed queries. The query processing for the automatically processed queries has been significantly improved as described in the previous section, but there is another effect. Compared to the manual query run in the last TREC conference, paragraph-level concepts were formed in a much more mechanistic way and were constrained by the language of the description and the narrative. In the previous conference, the only constraint was the vocabulary used in the queries, and the user's "world knowledge" was used to group concepts.

This resulted in considerably better retrieval performance. Additional experiments using manually edited queries are discussed in the next section.

Query Type	Average Precision			
	5 Docs	30 Docs	100 Docs	11-Pt Avg
INQ003	.64	.56	.45	.35
INQ004	.67 (+3.7%)	.58 (+2.7%)	.45 (0%)	.36 (+2.4%)

Table 2: Results for Routing queries

The routing results show that some improvement is obtained by combining the manual queries with the queries that were automatically modified using relevance feedback techniques. The difference in performance between the two types of queries is considerably less than last year, however. Our own experiments have also shown that no additional gains in performance were obtained by using more than the top 150 documents from the INQUERY output. This is a significant result from a practical viewpoint, since in an operational environment we will not want to rely on having output from other systems or need thousands of relevance judgements before performance improves.

5 Other Experiments

In the TIPSTER 24 month evaluation, which took place soon after the TREC-2 evaluation, we did a number of experiments that complement those done in TREC. In particular, we evaluated paragraph-based retrieval, expansion using an automatically generated thesaurus, feedback techniques that use phrases, and Japanese indexing techniques. In this section, we report some of the most interesting results. The precision figures given here are calculated using the TREC-2 relevance judgements, rather than the TIPSTER judgements.

The first two experiments were with adhoc queries. INQ041 (the numbers are consistent with those used in TIPSTER and other publications) is a run that used a different manually modified version of INQ001. That is, the manual modifications were the same as those done in the first TIPSTER and TREC evaluations, rather than the more restricted modifications done for INQ002. INQ042 is a run that combines INQ041 with INQ001.

Query Type	Average Precision			
	5 Docs	30 Docs	100 Docs	11-Pt Avg
INQ041	.68	.60	.50	.36
INQ042	.65 (−4.6%)	.61 (+1.7%)	.51 (+2.0%)	.38 (+5.6%)

Table 3: Results for TIPSTER adhoc queries

These results show that the manually modified queries can achieve significantly better precision at low recall levels. For example, at the 5 document cutoff level, the average precision for INQ041 is 9.7% higher than INQ001. The overall average is the same, however. This is a much smaller difference than was seen in the first TREC and TIPSTER evaluations

of the INQUERY system and it indicates that the automatic query processing has improved considerably.

The combination search (INQ042) is slightly worse than INQ041 at the 5 document cutoff level, but overall is better than either the automatic or manual queries on their own. An adhoc search that incorporates automatic paragraph-level matching was also tested in TIPSTER and this resulted in a further 5% improvement.

INQ023 and INQ024 are routing query sets that were created automatically using relevance judgements from volumes 1 and 2. In addition to the single-word terms added in INQ003, 10 phrase-level concepts and 20 paragraph-level concepts were added to the query. A phrase-level concept is a #UW5 two-word pattern that occurs frequently in the relevant documents, and a paragraph-level concept is a #UW50 two-word pattern. The #UWn operator looks for co-occurrence in any order in a text window of size n. The difference between INQ023 and INQ024 is that INQ023 contains the original query terms in addition to terms extracted from relevant documents, whereas INQ024 contains only terms from relevant documents.

Query Type	**Average Precision**			
	5 Docs	**30 Docs**	**100 Docs**	**11-Pt Avg**
INQ023	.67	.60	.47	.38
INQ024	.68 (+1.5%)	.59 (−1.7%)	.46 (−2.2%)	.39 (+2.6%)

Table 4: Results for TIPSTER routing queries

These results show that there is little difference between using the original query or just the relevant documents. This is probably due to the large number of relevance judgements available in this routing experiment. In a relevance feedback situation, where there are far fewer relevant documents, the original query is very important. It is clear that the addition of phrase and paragraph-level structure to the routing has improved performance. The average precision for INQ023 is 8.6% higher than INQ003. Combining these new runs with manually modified routing queries produced further improvements.

6 Summary

The TREC-2 runs, both in the adhoc and routing categories, provided further evidence that manually generated queries are not, in general, superior to automatically processed natural language queries. In the case of routing, in fact, the manual queries are significantly less effective. They do, however, improve the effectiveness of retrieval when used in combination with the automatic queries. This combination of query types has been a theme of the research at the University of Massachusetts and has been established as effective in a number of experiments.

The additional TIPSTER runs showed that learning structure in the form of phrases and paragraph-level co-occurrences is effective for routing. They also showed that learning techniques significantly improve performance (the best routing runs were more than 20%

higher in terms of average precision than the best queries that were not modified using relevance judgements). It is becoming apparent that techniques that may not work well in relevance feedback situations with few identified relevant documents, may be very effective in routing where there are many more relevant documents identified. We are currently doing experiments with different forms of weighting, including the use of identified non-relevant documents.

With regard to improving the performance of adhoc queries, we are continuing to carry out experiments with different ways of estimating the probabilities (or tf.idf weights) needed for the inference net, and with different forms of paragraph-level matching. Finally, as mentioned earlier, we have seen some significant improvements using automatic query expansion based on corpus analysis.

References

[1] J. P. Callan and W.B. Croft. An evaluation of query processing strategies using the TIPSTER collection. In *Proceedings of ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 347–356, 1993.

[2] J. P. Callan, W.B. Croft, and S.M. Harding. The INQUERY retrieval system. In *Proceedings of the 3rd International Conference on Database and Expert Systems Applications*, pages 78–83, 1992.

[3] W. Bruce Croft and Howard R. Turtle. Text retrieval and inference. In P. Jacobs, editor, *Text-Based Intelligent Systems*, pages 127–156. Lawrence Erlbaum, 1992.

[4] Hideo Fujii and W. Bruce Croft. A comparison of indexing techniques for japanese text retrieval. In *Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 237–246. ACM, 1993.

[5] David Haines and W. Bruce Croft. Relevance feedback and inference networks. In *Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 2–11. ACM, 1993.

[6] H.R. Turtle and W.B. Croft. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems*, 9(3):187–222, 1991.

[7] H.R. Turtle and W.B. Croft. A comparison of text retrieval models. *Computer Journal*, 1992. To appear.

DR-LINK: A System Update for TREC-2

Elizabeth D. Liddy
Sung H. Myaeng
School of Information Studies
Syracuse University
Syracuse, New York 132100-4100

liddy@mailbox.syr.edu; shmyaeng@mailbox.syr.edu

1. Overview of DR-LINK's Approach

The theoretical goal underlying the DR-LINK System is to represent and match documents and queries at the various linguistic levels at which human language conveys meaning. Accordingly, we have developed a modular system which processes and represents text at the lexical, syntactic, semantic, and discourse levels of language. In concert, these levels of processing permit DR-LINK to achieve a level of intelligent retrieval beyond more traditional approaches. In addition, the rich annotations to text produced by DR-LINK are replete with much of the semantics necessary for document extraction.

The system was planned and developed in a modular fashion and functional modularity has been achieved, while a full integration of these multiple levels of linguistic processing is within reach. As currently configured, DR-LINK performs a staged processing of documents, with each module adding a meaningful annotation to the text. For matching, a Topic Statement undergoes analogous processing to determine its relevancy requirements for documents at each stage. Among the many benefits of staged processing are: improvements and changes can be easily made within any module; the contribution of the various stages can be empirically tested by simply turning them on or off; modules can be re-ordered (as was done within the last six months) in order to utilize document annotations in various ways, and; individual modules can be incorporated in other evolving systems.

The purpose of each of the processing modules will be briefly introduced here (also see Figure 1) in the order in which the system is currently run, with fuller explanations provided in the section below: 1) the Text Structurer labels clauses or sentences with a text-component tag which provides a means for responding to the discourse level Topic Statement requirements of time, source, intentionality, and state of completion; 2) the Subject Field Coder provides a subject-based, summary-level vector representation of the content of each text; 3) the Proper Noun Interpreter and 4) the Complex Nominal Phraser provide precise levels of content representation in the form of concepts and relations, as well as controlled expansion of group nouns and content-bearing nominal phrases; 5) the Relation-Concept Detector produces concept-relation-concept triples with a range of semantic relations expressed via various syntactic classes, e.g. verbs, nominalized verbs, complex nominals, and proper nouns; 6) the Conceptual Graph Generator combines the triples to form a CG and adds Roget International Thesaurus (RIT) codes to concept nodes, and; 7) the Conceptual Graph Matcher determines the degree of overlap between a query graph and graphs of those documents which surpass a statistically predetermined criterion of likelihood of relevance based on ranking by the integrated processing of the first four system modules.

2. Detailed System Description

In the following system description, emphasis is placed on work accomplished within the last year, plus a basic overview description of each module. The more rudimentary processing details of each module plus fuller description of earlier development are available in the TREC-1 Proceedings (Harman, 1993).

2. A. Text Structurer

Since human interpretation of text is influenced by expectations regarding the text to be read, discourse level analysis is required for a system to approximate the same level of meaningful representation and matching. DR-LINK's Text Structurer is based on discourse linguistic theory which suggests that texts of a particular type have a predictable

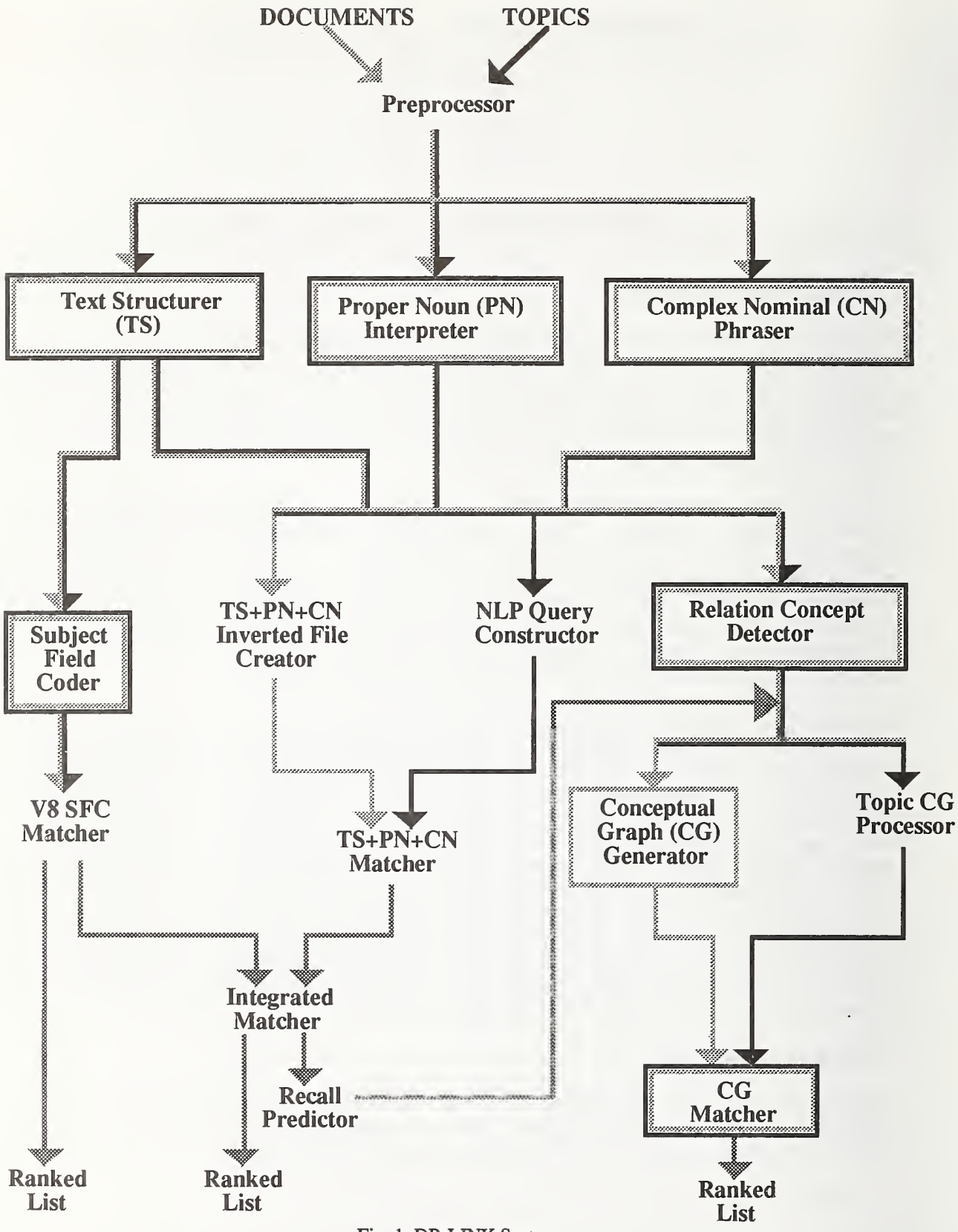


Fig. 1: DR-LINK System

text-level structure which is used by both producers and readers of that text-type as an indication of how and where certain information endemic to that text-type will be conveyed. We have implemented a Text Structurer for the newspaper text-type, which produces an annotated version of a news article in which each clause or sentence is tagged for the specific slot it instantiates in the news-text model, an extension of van Dijk's earlier model (1988). The structural annotations are used to respond more precisely to information needs expressed in Topic Statements, where some aspects of relevancy can only be met by understanding a Topic Statement's discourse requirements. For example, Topic Statement 75, states that:

> *Document will identify an instance in which automation has clearly paid off, or conversely, has failed.*

which contains the implicit discourse requirement that relevant instances should occur in the CONSEQUENCE component of a news article. DR-LINK extracts this requirement from the Topic Statement and will only assign a similarity value for the discourse-level of relevance to those documents in which the sought information occurs in a CONSEQUENCE component.

The current news-text model consists of thirty-eight recognizable components of information observed in a large sample of training texts (e.g. MAIN EVENT, VERBAL REACTION, EVALUATION, FUTURE CONSEQUENCE, PREVIOUS EVENT). The Text Structurer assigns these component labels to document clauses or sentences on the basis of lexical clues learned from text, which now comprise a special lexicon. We considered expanding the lexicon via available lexical resources such as Roget's International Thesaurus or WordNet, but our analysis of these resources suggested that they do not capture the particularities of lexical usage in the sublanguage of newspaper reporting.

The Text Structurer has recently been improved to assign structural tags at the clause level, a refinement which has corrected most of the anomalies that were observed in earlier testings of the Text Structurer. For example, given the new clause-level structuring, the following sentence is correctly interpreted as containing both future-oriented information in the LEAD-FUTURE segment and some nested information regarding a past situation in the LEAD-HISTORY segment.

> <LEAD-FUT> *South Korea's trade surplus,* <LEAD-HIST> *which more than doubled in 1987 to $6.55 billion,* </LEAD-HIST> *is expected to narrow this year to above $4 billion.* </LEAD-FUT>

We have recently implemented new matching techniques which more fully realize the Text Structurer's potential contribution to the system's performance. This was achieved as one outcome of a study which greatly increased our understanding of how text structure requirements in Topic Statements should be used for matching documents to Topic Statements. Analysis of relevant and non-relevant documents retrieved for a test sample of Topic Statements indicated that most of the errors in the Text Structurer's matching were not serious errors, but only slight mismatches in terms of the conceptual definitions of some of the text model's components. This suggested that our model was overly specific for the task of responding to discourse aspects of information requirements, and that matching Text Structure needs from a Topic Statement to structured documents called for a more generalized model. That is, Topic Statement text-structure requirements are not expressed at the same level of specificity at which Text Structure components are recognizable in documents.

Given this, we reduced the matching complexity via a function that maps the thirty-eight news-text components to seven meta-components. These are: LEAD-MAIN, HISTORY, FUTURE, CONSEQUENCE, EVALUATION, ONGOING, and OTHERS. The new approach allows the system to continue to impose the finer-level, 38-component structure on the newspaper articles themselves with excellent precision, but maps this fuller set of text components to the seven meta-components at the matching stage, as the Topic Statements' text structure requirements are coded at the meta-component level. Unofficial experimental results indicate that this new scheme has significantly increased the Text Structurer's contribution to an improved level of precision in the retrieval of relevant documents.

While the Text Structurer module processes documents as described above, the analysis of Topic Statements for their Text Structure requirements is done by the Natural Language Query Constructor (QC) which also analyzes the proper noun and complex nominal requirements of Topic Statements. The QC, as well as the matching and ranking of documents using these sources of linguistic information, is described below.

2. B. Subject Field Coder

The Subject Field Coder (SFC), as reported at TREC-1, has been producing consistently reliable semantic vectors to represent both documents and Topic Statements using the semantic codes assigned to word senses in a machine-readable dictionary. Details regarding this process are reported in detail in Liddy et al, 1993. Our more recent efforts on this module have focused on multiple ways to exploit SFC-based similarity values between document and query vectors. One implementation is the use of the ranked vector-similarity values for predicting a cut-off criterion of potentially relevant documents when the module is used as an initial filter. This is to replace the earlier practice used in the eighteenth month TIPSTER testing, where documents were ranked by their SFC-vector similarity to a query SFC-vector and the top two thousand documents were passed to the CG Matcher, since CG matching is too computationally expensive to handle all documents in the collection. To report the SFC's performance, at that time we reported how far down the ranked list of documents the system would need to process documents in order to get all the judged relevant documents. Although the results were highly promising (all relevant documents were, on average, in the top 37% of the ranked list based on SFC similarity values), this figure varies considerably for individual Topic Statements. Therefore, we needed to devise a method for predicting a priori for individual Topic Statements, the cut-off criterion for any desired level of recall. We first developed a method that could successfully predict a cut-off criterion based on just SFC similarity values. We then extended the algorithm to incorporate the similarity values produced when proper noun, complex nominal, and text structure requirements are considered as well, to produce an integrated ranking based on these varied sources of linguistic information.

The SFC-based cut-off criterion uses a multiple regression formula which was developed on the odd-numbered Topic Statements from 1 to 50 and a training corpus of Wall Street Journal articles. The regression formula takes into account the distribution of similarity values for documents in response to a particular query by incorporating the mean and standard deviation of the similarity value distribution, the similarity of the top-ranked document, and the desired recall level. The cut-off criterion was tested on the held-out, twenty-five Topic Statements. The averaged results, when a user is striving for 100% recall, showed that only 39.65 % of the 173,255 documents would need to be processed further. And this document set, in fact, contained 92% of the judged-relevant documents.

The advantage of the cut-off criterion is it's sensitivity to the varied distributions of SFC similarity values for individual Topic Statements, which appears to reflect how "appropriate" a Topic Statement is for a particular database. For many queries, a relatively small portion of the database, when ranked by similarity to the Topic Statement, will need to be further processed. For example, for Topic Statement forty-two, when the goal is 100% recall, the regression formula predicts a cut-off criterion similarity value which requires that only 13% of the ranked output be further processed, and the available relevance judgments show that this pool of documents contains 99% of the documents judged relevant for that query.

2. C. V-8 Matching

Given the complete modularity of the first four modules in the system, for the twenty-four month TIPSTER testing, we reordered two modules so that Text Structuring is done prior to Subject Field Coding. This allowed us to implement and test a new version of matching which combines in a unique way the Text Structurer and the Subject Field Coder. We refer to this version as the V-8 model, since eight SFC vectors are produced for each document, one for each of the seven meta-categories, plus one for all of the categories combined. The V-8 model, therefore, provides multiple SFC vectors for each document, thereby representing the distribution of SFCs over the various meta-text components that occur in a news-text document. This means, in the V-8 matching, that if certain content areas of the Topic Statement are required to occur in a document in one meta-text component, e.g. CONSEQUENCE, and other content is required to occur in another meta-text component, e.g. FUTURE, this proportional division can be matched against the V-8 vectors produced for each document at a fairly abstract, subject level. For the TIPSTER twenty-four month evaluation, we have experimented with several formulas for combining the similarity values of

the multiple SFC vectors produced for each document, including both a Dempster-Shafer combination and a straight averaging. Although official results are not yet available, our internal test results indicate that the combination of Text Structuring and Subject Field Coding produces an improved ranking of documents, especially when using the Dempster-Shafer method.

2. D. Proper Noun Interpreter

Our earlier work with the SFCoder, suggested that the most important factor in improving the performance of this upstream ranking module, would be to integrate the general subject-level representation provided by SFCodes with a level of text representation that enabled more refined discrimination. Analysis of earlier test results suggested that proper noun (PN) matching that incorporated both particular proper nouns (e.g. Argentina, FAA) as well as 'category' level proper nouns (e.g. third-world country, government agency) would improve precision performance. The Proper Noun Interpreter (Paik et al, 1993) that we developed provides: a canonical representation of each proper noun; a classification of each proper noun into one of thirty-seven categories, and; a means for expanding group nouns into their constituent members (e.g. all the countries comprising the Third World). Recent work on our proper noun algorithms, context-based rules, and knowledge bases, has improved the module's ability to recognize and categorize proper nouns to 93% correct categorization using 37 categories as tested on a sample set of 545 proper nouns from newspaper text. The improved performance has a double impact on the system's retrieval performance, as proper nouns contribute both to the downstream relation-concept representation used in CG matching as well as to the upstream proper noun, complex nominal, and text structure ranking of documents in relation to individual queries. Details of processing Topic Statements for their PN requirements and the use of this similarity value in document ranking is described in the later section on the Query Constructor.

2. E. Complex Nominal Phraser

A new level of natural language processing has been incorporated in the DR-LINK System with the implementation of the Complex Nominal (CN) Phraser. The motivation behind this addition was our recognition that either, in addition to proper nouns, or in the absence of proper nouns, most of the substantive content requirements of Topic Statements are expressed in complex nominals (i. e. noun + noun, e.g. "debt reduction", "government assistance", "health hazards"). Complex nominals provide a linguistic means for precise conceptual matching, as do proper nouns. However, the conceptual content of complex nominals can be expressed in synonymous phrases, in a different way than can the conceptual content of proper nouns, which are more particularized. Therefore, for complex nominals, a controlled expansion step was incorporated in the CN matching process in order to accomplish the desired goals of improved recall, as well as improved precision.

For input to the CN Phraser, the complex nominals in Topic Statements are recognizable as adjacent noun pairs or non-predicating adjective + noun pairs in the output of the part-of-speech tagger. Having recognized all CNs, the substitutable phrases for each complex nominal are found by computationally determining the overlap of synonymous terms suggested by RIT and statistical corpus analysis. These processes serve to identify all second order associations between each complex nominal constituent and terms in the database. Second order associations exist between terms that are used interchangeably in certain contexts. The premise here is that if, for example, terms *a* and *b* are both frequently premodified by the same set of terms in a corpus, it is highly likely that terms *a* and *b* are substitutable for each other within these phrases. The use of both corpus and RIT information appears to limit the over-generation that frequently results from automatic term expansion. Ongoing experiments on this new addition to the system will help us further refine the process and will be reported more extensively in the near future.

The terms that exhibit second order associations are compiled into equivalence classes. These equivalence classes provide substitutable synonymous phrases for Topic Statement complex nominals and are used by the matching algorithms in the same manner that the original complex nominals are used. The complex nominals and their substitutes are first used in the upstream matching of Topic Statements to documents as one contributing factor to the integrated similarity value, to be further explained in the section on the Query Constructor.

In addition, each complex nominal and its assigned relation provides a CRC to the RCD module for use in the final round of matching. For that module, semantic relations between the constituent nouns of each complex nominal are

assigned manually, using an ontology of forty-three relations. Some example complex nominals plus relation are:

[press] <- (SOURCE) <- [commentaries]
[growth] -> (MEASURE) -> [rate]
[electronic] <- (MEANS) <- [theft]
[campaign] <- (USED_FOR) <- [finances]

The development of the complex nominal CRC knowledge base was an intellectual effort for the twenty four month testing, but our current task is the full automation of the semantic relation assignment. Although difficult, our experience with the intellectual process has encouraged us to pursue appropriate NLP-based machine-learning techniques which will enable the system to automatically recognize and code semantic relations in complex nominals.

In CG matching, the existence of both case-frame relations and complex nominal relations make it possible for the system to detect conceptual similarity even if expressed in different grammatical structures, such as a verb + arguments in a Topic Statement and a complex nominal in a document, e.g.:

"reduce the debt" =[reduc*] -> (OBJECT) -> [debt]
"debt reduction" =[debt] <- (OBJECT) <- [reduc*]

To achieve the fullest exploitation of relational information despite grammatical realization, a further step was necessary in order to match on CRCs produced by verb-based analysis and CRCs produced by complex nominal analysis. This required the determination of the degree of relation-similarity across the two relation sets. There are approximately sixty relations used in case frames, while there are approximately forty relations used in complex nominals. A relation-similarity table was constructed that assigns a degree of similarity between twenty-eight pairs across the two grammatically-distinguished sets, and a degree of similarity between pairs within the same set. The relation-similarity table is used in the final CG matching to allow concepts that are linked by a relation in a document that is different from the relation that links the same two concepts in the Topic Statement, to still be awarded some degree of similarity. The quality and appropriateness of the similarity table will be determined by the results of the twenty-four month testing which will also provide empirical evidence of the Complex Nominal Phraser's impact on performance. Sample runs have indicated that the inclusion of complex nominals has a strongly positive impact on our results in both of its incorporations in the system.

2. F. Natural Language Query Constructor

We have implemented a Natural Language Query Constructor (QC) for DR-LINK which takes as input a Topic Statement which has been pre-processed by straight-forward techniques, such as part-of-speech tagging as well as SGML-tagging of the meta-language which reflects the typical request-presentation language used in Topic Statements (e.g. "A relevant document will ..." or "To be relevant..."). The QC produces a query which reflects the appropriate logical combinations of the text structure, proper noun, and complex nominal requirements of a Topic Statement. The basis of the QC is a sublanguage grammar which is a generalization over the regularities exhibited in the Topic, Description, and Narrative fields of the one hundred fifty TIPSTER Topic Statements. It should be noted that the sublanguage grammar, with minor modifications, is capable of handling non-TIPSTER queries, so its generalized utility is promising. Earlier work (Liddy et al, 1991) demonstrated that the sublanguage approach is an effective and efficient approach to natural language processing tasks within a particular text-type, here Topic Statements.

For the twenty-four month runs, the QC sublanguage grammar detects the required logical combination of text structure components, proper nouns, and complex nominals. These are the specific entities which we consider to be particularly revealing indicators of relevant documents. In most cases, matching on these classes produces high-precision ranked results, although there are some instances in which single common nouns may also be needed. After analyzing the twenty-four month results, we will determine whether to expand the range of linguistic types which can be used to instantiate the variables in the QC's logical assertions.

The QC sublanguage grammar relies on function words (e.g. conjunctions, prepositions, relative pronouns), meta-level phrases (e.g. "such as", "examples of", "as well as"), and punctuation (e.g. commas, semi-colons) to recognize and extract the relevancy requirements of Topic Statements. These linguistic features serve as clues to the 'organizing' structure of a Topic Statement and present each Topic Statement's unique thematic content in a recognizable frame. The QC sublanguage interprets a Topic Statement into pattern-action rules which are used to reduce each sentence in a Topic Statement into a first order logic assertion, reflecting the boolean-like requirements of Topic Statements, including NOT'd assertions. In addition, definite noun phrase anaphors are recognized and resolved by sublanguage grammar processing rules.

2. G. Integrated Matcher

Each logical assertion produced by the QC for a Topic Statement is evaluated against the entries in the document inverted file and a weight is assigned to each segment of text (either a clause or a sentence) which has any similarity. The weighting scheme we are currently using evolved from iterative testing. Each segment of text is indexed in the inverted file with a text structure component label and will be assigned a weight if it contains any proper nouns or complex nominals that match the Topic Statement's requirements. The following weights are assigned:

proper noun	=	1.00
complex nominal	=	1.00
proper noun category	=	0.50

This means, for example, that if, in response to the following requirement from a Topic Statement:

> *A relevant document will provide data on Japanese laws, regulations, and/or practices which help the foreigner understand how Japan controls, or does not control, stock-market practices which could be labeled as insider trading.*

a document text-segment contains 'Japanese law', and 'stock-market practice' (or one of its synonymous phrases), and 'insider trading' (or one of its synonymous phrases), that segment is assigned a preliminary value of 3.00. Depending on which field in the Topic Statement the assertion came from, and whether the document text-segment matches the Topic Statement's Text Structure requirement, the preliminary value will be multiplied by one of the following co-efficients:

Topic field and **required** Text Structure component	=	1.00
Desc, Narr, or Concept field and **required** Text Structure component	=	0.75
Topic field and **non-required** Text Structure component	=	0.50
Desc, Narr, or Concept field and **non-required** Text Structure component	=	0.25

So if 'Japanese law' and 'stock-market practice' and 'insider trading' were conceptual requirements from a Topic field assertion that also required them to occur in an EVALUATION or LEAD-MAIN text component, and they occurred in a document text segment which has been tagged by the Text Structurer as EVALUATION, the value of 3 would be multiplied by 1; whereas if that assertion came from the Description field in the Topic Statement and the three required phrases occurred in a document text segment labelled CONSEQUENCE by the Text Structurer, the value of 3 would be multiplied by .25.

Since the QC interprets each sentence in the Topic, Description, Narrative, and Concept fields in a Topic Statement, multiple, sometimes overlapping, sometimes repetitive assertions are produced for a single Topic Statement. In the current implementation, each of these Topic Statement assertions is compared to the inverted document file, and the highest similarity value for a single assertion in the document is used as that document's integrated similarity value for that Topic Statement.

The similarity value which results from the QC module matching is combined with the SFC similarity value of the document, and an integrated similarity score for each document is produced. This similarity value can be used in several ways. Firstly, the two similarity values can be used to provide a full ranking of all the documents which

takes into account the lexical, semantic and discourse sources of linguistic information in both documents and queries. Secondly, it can serve as input to a filter which uses a more complex version of the original cut-off criterion to determine how many documents should be further processed by the system's final modules.

For the Integrated Matcher to produce a combined ranking, each document's similarity value for a given Topic Statement can be thought of as being composed of two elements. One element is the SFC similarity value and one element is the similarity value that represents the combined proper noun, complex nominal, and text structure similarities. Additionally, the system will have computed the regression formula, the mean, and standard deviation of the distribution of the SFC similarity values for the individual Topic Statement. Using these statistical values, the system produces the cut-off criterion value. Since we know from the eighteen-month results, that 74% of the relevant documents had what we refer to as a k-value (then PN value; now PN, CN, TS values) and the remaining 26% of the relevant documents had no k-value, we use this information to predict what proportion of the predicted relevant documents should come from which segment of the ranked documents for full recall. The combined ranking can be envisioned as consisting of four segments, as shown in Figure 2.

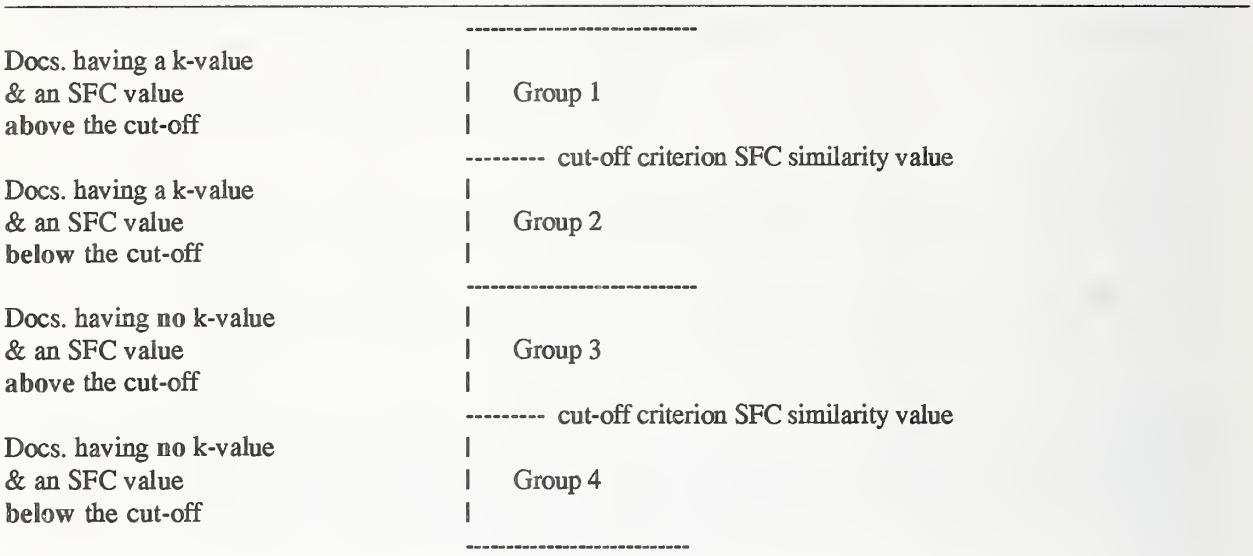


Fig. 2: Schematic of Segmented Ranks from SFC & Integrated Ranking (k-value)

Four groups are required to reflect the two-way distinction mentioned above. The first distinction is between those groups which have a k-value and which should contain 74% of the relevant documents and those documents without a k-value, which should contribute 26% of the relevant documents. The second distinction is between those documents whose SFC similarity value is above the predicted cut-off criterion and those whose SFC similarity value is not.

When a cut-off criterion is the application desired, the system will produce the ranked list in response to a desired recall level, by concatenating the documents above the appropriate cut-off for that level of recall from Group 1; then documents above the appropriate cut-off for that level of recall from Group 3. However, since our test results show that there is a potential 8% error in the predicted cut-off criterion for 100% recall, we use extrapolation to add the appropriate proportion of the top ranked documents from Group 2 to Group 1, before concatenating documents from Group 3. These same values are used to produce the best end-to-end ranking of all the documents using the various segments.

Document ranks are produced by the Integrated Matcher and the cut-off criterion is used either by an individual user who requires a certain recall level for a particular information need, or, as in the twenty four month TIPSTER test situation, by the system to determine how many documents from the Integrated Matcher ranking will be passed on to the final modules for further processing.

2. H. Topic Statement Processing for Conceptual Graph Generation

The processing of topic statements for CG generation does not make use of the output of the Natural Language Query Constructor, but instead the current system first applies the same RCD and CG generator modules to produce topic statement (TS) CGs. Several TS-specific processing requirements have been identified, some of which have been implemented as post-processing routines and others are under development.

- Elimination of concept and relation nodes corresponding to contentless meta-phrases (e.g. "Relevant document must identify ..."). If both of the concept nodes in a concept-relation-concept triple belong to a meta-phrase, the CRC is ignored. When only one of them is a meta-phrase concept, the triple is not removed blindly unless the other concept occurs in another triple.

- Handling of negated parts of topic statements. The weights are adjusted in such a way that an occurrence of the negated concept in a document will contribute to the negative evidence that the document will be relevant. In effect, the two weights for the concept are switched.

- Automatic assignment of weights to concept and relation nodes. There are several factors we consider: the conventional way of determining the importance of terms using inverse document frequency (IDF) and total frequency; the location of terms occurring in topic statements; the part of speech information for each term; and indications in the topic statement sublanguage (e.g. the document MUST contain ...). Although we have implemented a program that tags individual words with the degree of importance based on the sublanguage patterns, we assigned concept weights based on IDF values of terms in the collection for the evaluation, due to time constraints.

- Merging common concept appearing in different sections of topic statements. Although it is not safe in general to assume that two concepts sharing the same concept name actually refer to the same concept instantiation and merge them blindly, we have observed that this is not the case in the topic statements. In fact, we believe that it is desirable to merge CG fragments using common concept nodes. This is an important process that eliminates undesirable effects on scoring. Without this, a document containing a concept occurring repeatedly in <desc>, <narr>, and <con> fields would be ranked unnecessarily high (or low if it is negated) because each occurrence of the concept would make an independent contribution to the overall score.

Since an integrated automatic topic processing module was not available, the mechanical aspects of the process were hand-simulated with some parts done automatically and other done manually.

2. I. Relation Concept Detector (RCD)

The output of the Complex Nominal Phraser and the Proper Noun Interpreter modules described above provide concept-relation-concept triples directly to the Relation-Concept Detector (RCD) module. In addition, the following RCD handlers are operative.

One of the more distinct aspects of the DR-LINK system is its capability of extracting and using relations in the final representation of documents and topic statements in their CG representations. This module provides building blocks for the CG representation by generating concept-relation-concept triples based on the domain-independent knowledge bases we have been constructing with machine-readable resources and corpus statistics. In this module, there are several handlers that are activated selectively depending on the input sentence.

2. I. 1. Case Frame (CF) Handler

The main function of the CF Handler is to generate concept-relation-concept triples where one of the concepts comes typically from a verb. It identifies a verb in a sentence and connects it to other constituents surrounding the verb. Since the relations (about 50 we use currently) included in our representation are originated from the theories of linguistic case roles (Somers, 1987, and Cook, 1989) and are all semantic in nature, this module consults the

knowledge base containing 13,786 case frames we have constructed, each of which prescribes a pattern involving a verb and the corresponding concept-relation-concept triples.

Given a set of case frames for different senses of `decline', for example,

```
(decline 1      ((PATIENT subject ? obligatory)))

(decline 2      ((AGENT subject human obligatory)
                 (PATIENT object ? optional)))

(decline 3      ((AGENT subject human obligatory)
                 (ACTIVITY infinitive ? obligatory)
                 (link infinitive subject AGENT)))
```

AGENT, PATIENT, and ACTIVITY are the relations that connect the verb to other constituents. The second components (e.g. subject) prescribe the syntactic categories of the constituents and the third components (e.g. human) semantic restrictions that the subject and object should satisfy. The last components (e.g. obligatory) indicate whether the constituent must exist in a sentence in order for the particular case frame to be instantiated. The last line of the third case frame instructs the CF handler to link the subject to the infinitive verb with the AGENT relation. This kind of linking instructions allow the CF handler to produce triples containing non-verbal constituents.

The CF Handler selects the best case frame by attempting to instantiate each case frame and determine which one is satisfied most by the sentence at hand. This can be seen as a sense disambiguation process using both syntactic and semantic information. The semantic restriction information contained in the case frames were obtained from LDOCE, and when the sentence is processed, the CF handler also consults LDOCE to get semantic restriction information for individual constituents surrounding the verb in the sentence and compares it with the restrictions in the case frames of the verb as a way to determine which case frame is likely to be the correct one.

With the following sentence fragment,

... the chairman declined to elaborate on the disclosure ...

the CF handler chooses the third case frame and produces

```
[decline] -> (AGENT) -> [chairman]
[decline] -> (ACTIVITY) -> [elaborate]
[elaborate] -> (AGENT) -> [chairman]
```

In the current implementation, the input text to the CF handler is first tagged with part-of-speech information and bracketed for constituent boundaries. BBN's POST tagger (Metter et al., 1991) has been used to attach a part-of-speech tag to individual words. The constituent boundary bracketer we developed then marks boundaries of grammatical constituents such as infinitives, noun phrases, prepositional phrases, clauses, etc.

At the time of writing, the case frame knowledge base contains 13, 786 case frames, of which 13,444 are for all the verb entries (5,206) in LDOCE, and the rest are for 342 verbs that appear in the Wall Street Journal collection but are not in the LDOCE as a headword. While we have constructed case frames for most of the phrasal verbs in LDOCE, the capability of processing phrasal verbs has not
been implemented in the current CF Handler.

2. I. 2. Nominalized Verb (NV) Handler

The nominalized verb handler has been implemented for the DR-LINK system we ran for the TIPSTER 24th month evaluation. Its main function is to consult the NV case frames to identify a NV in a sentence and create

concept-relation-concept triples based on the rule. At the same time, it converts the NV into its verb form. In this way, we can allow for a match between a CG fragments generated from a phrase containing verb and another fragment generated from a noun phrase containing the corresponding nominalized verb. For example, the NV Handler converts the sentence fragment

... the company's investigation of the incident ...

into

[investigate] -> (AGENT) -> [company]
[investigate] -> (PATIENT) -> [incident].

This process is much more than a sophisticated way of performing stemming in that we canonicalize concept-relation-concept triples rather than just concept nodes.

For NV processing, 15, 053 case frames have been generated for 1,593 nominalized verbs. Most of the case frames for NVs were automatically generated from the corresponding verb case frames. This process was also facilitated by identifying potential NVs from LDOCE.

No explicit testing of the impact of NVs in information retrieval has been done yet although we have convinced ourselves with anecdotal evidence that this would improve the retrieval performance. More semantic processing of nominalized verbs in determining the relations to the surrounding constituents is on the future research agenda. More rigorous study on the impact of NVs on information retrieval should be done, too.

2. I. 3. Noun Phrase (NP) and Prepositional Phrase (PP) Handler

The noun phrases that are not handled by the complex-nominal handler or by the nominalized verb handler are analyzed so that the head noun is connected to the concepts outside the noun phrase (e.g. a verb concept in the CF Handler). In addition, this module identifies individual concepts corresponding to adjectives and other nouns in a compound noun and connects them with CHARACTERISTIC, ATTRIBUTE, or LINK relations. LINK is the most generic relation in our system.

Once noun phrases are handled this way, this module handles prepositional phrases by connecting the head noun concept of the noun phrase to the preceding constituent (e.g. a verb or a noun). The preposition attachment problem is a difficult one, and the current implementation takes the simple- minded approach with general relations such as LINK, which can match with many of other semantically more specific relations. Our preliminary analysis indicates that this approach correctly handles about 75% of the prepositional phrase cases in the Wall Street Journal collection. More accurate and finer-level processing will be done with more semantically oriented rules that check the semantic restrictions and use more specific relations. The role of this handler will be diminished when we process phrasal verbs as part of the CF handler, for which we have constructed case frames.

2. I. 4. Ad-hoc Handler

This module looks for lexical patterns not covered by any of the other special handlers discussed above. Its processing is also driven by its own knowledge base of patterns to infer relations between concepts. For example, a sentence fragment

... bought the item for the purpose of satisfying ...

contains a pattern

[VERB] ... for the (ADJ) purpose of [NP]

in the knowledge base, and hence results in a triple

[buy] -> (GOAL) -> [satisfy]

The knowledge base contains a small number of simple patterns involving BE verbs and more than 350 pattern rules for phrasal patterns across phrase boundaries, by which important relations are extracted. The pattern rules specify certain lexical patterns and the order of occurrences of words belonging to certain part-of-speech categories, and the concept-relation-concept triples to be generated. These patterns require a processing capability no more powerful than a finite state automaton. Due to the time constraints, however, the current ad-hoc handler has not been generalized to process all the patterns, and about 30% of the patterns in the knowledge base are recognized and handled correctly.

2. J. Conceptual Graph (CG) Generator

After individual RCD modules have generated concept-relation-concept triples for a document, the CG generator merges them to form a set of conceptual graphs, each corresponding to a clause in most cases. Since more than one handler can generate different triples for the same concept pairs (e.g. a prepositional phrase handled by the CF handler and the NP/PP handler) based on independently constructed rules and on independent processes, a form of conflict resolution is necessary. In the current implementation, we simply order the execution of different handlers based on the general quality of the rules and the resulting triples so that more reliable handlers have higher precedence.

The concept nodes in the resulting CGs can not only contain general concept names but also some instantiations (referents) of the concepts. Such a concept can be derived either from a proper noun such as a company name or from a sub-ordinate clause. In the latter case, the instantiation is a CG itself to produce a CG like

[country: {US}] <- (SOURCE-OF-INFO) <- [C#: [[pact] <- (PATIENT) <- ...]

In the current implementation, concepts with the same instantiation are merged across sentences to form a larger CG, but concept with the same label but without any referents across sentences are treated as separate concepts and are not merged. A pronoun resolution method is being implemented to merge a pronoun to its antecedent as a way to increase the connectivity of CGs and hence increase the usefulness of relation nodes.

As a way to make our current representation more "conceptual", we have implemented a module that adds RIT (Roget's International Thesaurus) codes to individual concept nodes so that the label on the nodes is not a word but a position of the hierarchy of RIT. The lowest level position beyond individual lexical items in the RIT hierarchy is called a semi-colon group consisting of several terms within the delimiter of semi-colons, which represents a concept.

The mapping from a word (called target) in text to a position in RIT requires sense disambiguation, and our approach is to use the words surrounding the target word as the context within which the sense of the target word is determined and one or more RIT codes are selected. The algorithm selects minimal number (i.e. one or more) of RIT codes, not just the best one, for target words since we feel that some of the sense distinctions made in RIT are unnecessarily subtle, and it is unlikely that any attempts to make such fine distinctions would be successful and hence contribute to information retrieval.

We have produced RIT-coded documents and topic statements for the San Jose Mercury collection and the routing queries. All the concept nodes derived from nouns now have RIT codes selected using the surrounding text as the context. Those concept nodes derived from verbs also have RIT codes but in a different way. Instead of using the surrounding text as the context and trying to disambiguate senses (we concluded that this method is not reliable for verbs), we first assign RIT codes to each sense of LDOCE verb entries using the same method. In this case the context become the definition text in LDOCE. Once we select the right case frame by Case Frame Handler while text is processed, the RIT codes attached to the case frame are automatically assigned to the target verb.

2. K. Conceptual Graph (CG) Matcher

The main function of the CG matcher is to determine the relevance of each document against a topic statement CG

and produce a ranked list of documents as the third and final output of the system. Using the techniques necessary to model plausible inferences with CGs (Myaeng and Khoo, 1992), this module computes the degree to which the topic statement CG is covered by the CGs in the document (see Myaeng and Liddy (1993) and Myaeng & Lopez-lopez (1992) for details).

While the most obvious strength of the CG approach is its ability to enhance precision by exploiting the structure of the CGs and the semantics of relations in document and topic statement CGs, and by attempting to meet the specific semantic constraints of topic statements, we also attempt to increase recall by allowing flexibility in node-level matching. Concept labels can be matched partially (e.g. between `Bill Clinton' and `Clinton'), and both relation and concept labels can be matched inexactly (e.g. between `aid' and `loan' or between `AGENT' and `EXPERIENCER'). For both inexact and partial matches, we determine the degree of matching and apply a multiplication factor less than 1 to the resulting score. For inexact matching cases, we have used a relation similarity table that determines the degree of similarity between pairs of relations. Although this type of matching slows down the matching time, we feel that until we have a more accurate way of determining the conceptual relations and a way to represent at a truly conceptual level (e.g. our attempt to use RIT codes), it is necessary. More importantly, the similarity table reflects our ontology of relations and allows for matching between relations produced by different RCD handlers whose operations in turn are heavily dependent on the domain-independent knowledge bases.

We have done a series of matching experiments internally to evaluate various strategies in CG matching/scoring and document representation with the goal of selecting the best one for the final TIPSTER 24th month runs. The first question we had was how to "normalize" the score assigned to a document based on the current scoring scheme. As described above, the scoring algorithm is query-oriented in the sense that the score reflects to what extent the query CG is covered by the document CG. While this approach is theoretically justifiable, one potential drawback is that a document containing the entire query CG is not ranked higher than one that contains fragments of the query CG scattered in the document as long as they cover the same query CG. That is "connectivity" or `coherence" of matching document CG is not fully taken into account.

With the intuitive notion that the number of matching CG fragments in a document would be inversely proportional to "connectivity", we have been experimenting with various normalization factors that are a function of the number of matching CG fragments. At the time of writing, our experimental data show that when we consider 12 sentential CGs as a unit (called "paragraph") and use the number of units containing one or more matching CG fragments in the normalization function, we obtain the best result. Among all the functions we have tried, the best normalization factor we have found experimentally so far is:

$$1.05 \wedge (1-x)$$

where x is the number of text units that contain one or more matching CG fragments. When this is combined with the maximum of the scores assigned to individual "paragraph" as follows:

$$S*1.05\wedge(1-x) + 0.4*M$$

where S is for the unnormalized score and M for the maximum "paragraph" score, we obtained the best results. Since we determined the constants incrementally, it is entirely possible that different combination of the constants can give better results. It is relatively clear based on these experiments that the first or the second term alone are always inferior to the combination. The number of sentential CGs for "paragraphs", 12, seems also pretty stable.

We have produced TIPSTER runs using the RIT-coded documents and topic statements. The current matching program attempts to match on RIT codes only when the concept names (words) don't match. Because of this conservative approach, the RIT codes do not block a match between two different polysemous words and thus have any direct impact on the word ambiguity problems in IR. With the disambiguation process employed when RIT codes are chosen for a noun or verb, however, the net effect is analogues to term expansion with sense disambiguation. It should be noted that since RIT codes are used for both document and query concepts, this amounts to sense-disambiguated term expansion on both queries and documents.

While the original motivation was to represent documents and topic statements at more conceptual level using RIT codes, we are also testing the effectiveness of RIT-based term expansion in IR environments. Using the scheme we have developed for term clustering using contextual information in the corpus (Myaeng & Li, 1992), we have three methods to evaluate: RIT-based expansion, term-cluster based expansion, and a combination of the two so that we can eliminate the problem of using a general- purpose thesaurus and the errors made by the term-clustering method.

For TIPSTER evaluation, we have submitted two sets of four runs: one with RIT codes and the other without them. Each set consists of three runs for different scoring schemes and the last one for the combination of the three runs which appears to produce the best result in our internal experiment.

3. Test Runs

The DR-LINK group elected to put their efforts into continued work for the twenty-four month TIPSTER testing, and as a result we lost our opportunity to have TREC-compatible results to discuss at this time. Although our twenty-four month TIPSTER runs have been submitted, many of our top-ranked documents were not amongst those submitted by TREC participants, so it is virtually impossible to make even unofficial reports on our system's performance. We trust that in the near future there will be some comparable groups and/or runs to measure ourselves against after the results from both TIPSTER and TREC-2 are available.

4. Summary

As the above descriptions should convey, we have made a great deal of progress in the development and integration of the DR-LINK System since TREC-1. Unfortunately, the absence of quantified results of our performance limits our convincing power. However, we are pleased to have demonstrated that a system implementation of our original notion of integrating multiple levels of linguistic processing so that retrieval can be conducted at a conceptual rather than word-based level is nearly achieved.

Many rich research and implementation ideas remain to be explored in all of the DR-LINK modules, particularly those which have only been in existence for a few months.

5. Acknowledgments

The research reported herein was funded by ARPA's TIPSTER Program. We are grateful to both Longmans' for access to the machine readable tape of LDOCE and to BBN for the loan of their POST tagger.

Without the untiring efforts of the following individuals, neither the DR-LINK System nor our very promising results would have been possible: Margot Clark, Bob Del Zoppo, Saleh Elmohammed, Chris Khoo, Tracey Lemon, Ming Li, Mary McKenna, Ken McVearry, Carin Obad, Woojin Paik, Ching-cheng Shih, Joe Woelfel, Edmund Yu, Ahsan Zhia.

References

Cook, W. (1989). Case Grammar Theory. Washington, D.C. : Georgetown University Press.

Harman, D. (Ed.) (1993). The first Text REtrieval Conference (TREC-1). National Institute of Standards and Technology.

Liddy, E.D., Jorgensen, C.L., Sibert, E. & Yu, E.S. (1991). Sublanguage grammar in natural language processing. Proceedings of RIAO '91 Conference. Barcelona.

Liddy, E.D., Paik, W. & Yu, E.S. (1993). Document filtering using semantic information from a machine readable dictionary. Proceedings of the ACL Workshop on Very Large Corpora.

Meteer, M., Schwartz, R. & Weischedel, R. (1991). POST: Using probabilities in language processing. Proceedings of the Twelfth International Conference on Artificial Intelligence. Sydney, Australia.

Myaeng, S, H. & Khoo, Chris (1992). On uncertainty handling in plausible reasoning with conceptual graphs. Proceedings of the 7th Conceptual Graphs Workshop. Las Cruces, NM.

Myaeng, S. H. & Li, Ming (1992). Building a database-specific concept hierarchy for information retrieval by acquiring lexical semantics from a corpus. Proceedings of the First International Conference in Information and Knowledge Management. Baltimore, MD.

Myaeng, S. H. & Liddy, E. D. (1993). Information retrieval with semantic representation of texts. Proceedings of Symposium on Document Analysis and Information Retrieval. Las Vegas.

Myaeng, S. H. & Lopez-Lopez, A. (1992). Conceptual graph matching: A flexible algorithm and experiments. Journal of Experimental and Theoretical Artificial Intelligence. Vol. 4, 107-126.

Paik, W., Liddy, E.D., Yu, E.S., & McKenna, M. (1993). Categorizing and standardizing proper nouns for efficient information retrieval. Proceedings of the Workshop on Acquisition of Lexical Knowledge from Text. Association for Computational Linguistics. pp. 154-60.

Somers, H. L. (1987). Valency and Case in Computational Linguistics. Edinburgh: Edinburgh University Press.

van Dijk, T. A. (1988). News as discourse. HillsdaleLawrence Erlbaum Associates.

Feedback and Mixing Experiments with *MatchPlus*

Stephen I. Gallant* William R. Caid† Joel Carleton† Todd W. Gutschow†
Robert Hecht-Nielsen† Kent Pu Qing† David Sudbeck†

HNC Inc

Abstract

We briefly review the *MatchPlus* system and describe recent developments with learning word representations, experiments with relevance feedback using neural network learning algorithms, and methods for combining different output lists.

1 Introduction

HNC is developing a neural network related approach to document retrieval called *MatchPlus*[1]. Goals of this approach include high precision/recall performance, ease of use, incorporation of machine learning algorithms, and *sensitivity to similarity of use.*

To understand our notion of sensitivity to similarity of use, consider the four words: 'car', 'automobile', 'driving', and 'hippopotamus'. 'Car' and 'automobile' are synonyms and they very often occur together in documents; 'car' and 'driving' are related words (but not synonyms) that sometimes occur together in documents; and 'car' and 'hippopotamus' are essentially unrelated words that seldom occur within the same document. We want the system to be sensitive to such similarity of use, much like a built-in thesaurus, yet without the drawbacks of a thesaurus, such as domain dependence or the need for hand-entry of synonyms. In particular we want a query on 'car' to prefer a document containing 'drive' to one containing 'hippopotamus', and we want the *system itself* to be able to figure this out from the corpus.

The implementation of *MatchPlus* is motivated by neural networks, and designed to interface with neural network learning algorithms. High-dimensional ($\approx$ 300) vectors, called context vectors, represent word stems, documents, and queries in the same vector space. This representation permits one type of neural network learning algorithm to generate stem context vectors that are sensitive to similarity of use, and a more standard neural network algorithm to perform routing and automatic query modification based upon user feedback, as described below.

Queries can take the form of terms, full documents, parts of documents, and/or conventional Boolean expressions. Optional weights may also be included.

The following sections give a brief overview of our implementation, and look at some recent improvements and experiments. For a previous description of the approach and comments on complexity considerations see [1]; a longer journal article is in preparation.

2 The Context Vector Approach

One of the most important aspects of *MatchPlus* is its representation of words (stems), documents, and queries by high ($\approx$ 300) dimensional vectors called *context vectors.* By representing all objects in the *same* high dimensional space we can easily:

1. Form a document context vector as the (weighted) vector sum of the context vectors for those words (stems) contained in the document.
2. Form a query context vector as the (weighted) vector sum of the context vectors for those words (stems) contained in the query.
3. Compute the distance of a query Q to any document. Moreover if document context vectors are normalized, the closest document d (in Euclidean distance) has the context vector V^d that gives highest dot product with the query context vector V^q:

   $<\text{closest } d> = \{d | V^d \cdot V^Q \text{ is maximized for } d \in D\}$

   (proof: $\|V^d - V^Q\|^2 = \|V^d\|^2 + \|V^Q\|^2 - 2(V^d \cdot V^Q) = \text{const} - 2(V^d \cdot V^Q)$.)

*124 Mt Auburn St, Suite 200. Cambridge, MA 02138
†5501 Oberlin Drive, San Diego, CA 92121.
[1]Patents pending.

4. Find the closest document to a given document d by treating $\mathbf{v}^d$ as a query vector.

5. Perform relevance feedback. If d is a relevant document for query Q, form a new query vector

$$\widehat{\mathbf{v}^Q} = \mathbf{v}^Q + \alpha \mathbf{v}^d$$

where α is some suitable positive number (eg 3). (See also [8].) Note that search with $\widehat{\mathbf{v}^Q}$ takes the same amount of time as search with $\mathbf{v}^Q$.

2.1 Context Vector Representations

Context vector representations (or feature space representations) have a long history in cognitive science. Work by Waltz & Pollack [10] had an especially strong influence on the work reported here. They described a neural network model for word sense disambiguation and developed context vector representations (which they termed micro-feature representations). See Gallant [2] for more background on context vector representations and word sense disambiguation.

We use context vector representations for document retrieval, with all of the representation being learned from an unlabeled corpus. A main constraint for all of this work is to keep computation and storage reasonable, even for very large corpora.

2.2 Bootstrap Learning

Bootstrapping is a machine learning technique that begins with vectors having randomly generated positive and negative components, and then uses an unlabeled training corpus to modify the vectors so that similarly used terms have similar representations. Previously we had used partially hand-entered components as described in [2], but we have dispensed with all hand entry in current implementations.

Although there are important proprietary details, the basic idea for bootstrapping is to make a stem's vector more like its neighbors by adding a fraction of their vectors to the stem in question. We make use of a key property of high-dimensional vectors: the ability to be 'similar to' a multitude of vectors. This is the same property that allows the vector sum that represents a document to be similar to individual term vector summands. (Similarity between normalized vectors is measured by their inner product.)

Note that bootstrapping takes into account *local word positioning* when assigning the context vector representation for stems. Moreover it is nearly invariant with respect to document divisions within the training corpus. This contrasts with those methods where stem representations are determined solely by those *documents* in which the stem lies.

2.3 Context Vectors for Documents

Once we have generated context vectors for stems it is easy to compute the context vector for a document. We simply take a weighted sum of context vectors for all stems appearing in the document[2] and then normalize the sum. This procedure applies to documents in the training corpus as well as to new documents. When adding up stem context vectors, we can use term frequency weights similar to conventional IR systems.

2.4 Context Vectors for Queries; Relevance Feedback

Query context vectors are formed similarly to document context vectors. For each stem in the query we can apply a user-specified weight (default 1.0). Then we can sum the corresponding context vectors and normalize the result.

Note that it is easy to implement traditional relevance feedback. The user can specify documents (with weights) and the document context vectors are merely added in with the context vectors from the other terms. We can also find documents close to a given document by using the document context vector as a query context vector.

2.5 Retrieval

The basic retrieval operation is simple; we find the document context vector closest to the query context vector and return it. There are several important points to note.

1. As many documents as desired may be retrieved, and the distances from the query context vector give some measure of retrieval quality.

2. Because document context vectors are normalized, we may simply find the document d that maximized the dot product with the query context vector, $\mathbf{v}^q$:

$$\max_d \{\mathbf{v}^d \cdot \mathbf{v}^Q\}.$$

3. It is easy to combine keyword match with context vectors. We first use the match as a filter for documents and return documents in order by closeness to the query vectors. If all matching

[2] Stopwords are discarded.

documents have been retrieved, *MatchPlus* can revert to context vectors for finding the closest remaining document.

4. *MatchPlus* requires only about 300 multiplications and additions to search a document. Moreover it is easy to decompose the search for a corpus of documents with either parallel hardware or, less expensively, several networked conventional machines (or chips). Each machine can search a subset of the document context vectors and return the closest distances and document numbers in its subset. The closest from among the distances returned by *all* the processors then determines the documents chosen for retrieval.

   We also plan to investigate a *cluster tree* pruning procedure that finds nearest neighbor document context vectors without having to compute dot products for all document context vectors. This data organization affects retrieval speed, but does not change the order in which documents are retrieved.

3 Experiments and Runs Submitted

The four runs (two ad-hoc, two routing) submitted to TREC-II are described in the following sections.

3.1 Totally Automated

This run used the entire topic as a query, with weight of 2 applied to the topic section. We also employed a $Match filter that put documents having at least 4 of the concept terms before other documents. Context vectors determined the actual order of the documents (as well as which documents were in the 1000-document submission list.) Note that these ad hoc retrieval runs were totally automated from topic to retrieval list.

3.2 Relevance Feedback From the First 20 Retrievals

Here we took the top 20 documents from the previous section, read them to estimate relevance, and then modified the initial query for the remaining 980 retrievals. To change a query we added and normalized all relevant document context vectors from the first 20 retrievals and then added this vector to 0.7 times the original query vector. (The 0.7 factor was obtained from experiments with a different corpus.)

3.3 Routing Using Neural Network Learning and Output Mixing

Both routing runs used two types of neural network learning.

3.3.1 Stem Weight Learning

The Stem Weight Learning approach uses neural network learning to compute weights for terms in a query; there is one neural network input for every query term.[3]

Every judged document for a query provides a training example as follows. Let $\mathbf{V}^n$ be the context vector for judged document n. If query term i has context vector $\mathbf{V}^i$, then the i^{th} network input for training example n is given by $\mathbf{V}^i \cdot \mathbf{V}^n$, the inner product of $\mathbf{V}^i$ and $\mathbf{V}^n$. For training example n, the desired network output is ± 1, according to the relevance of document n.

A fast single-cell learning algorithm, the pocket algorithm with ratchet [3, 4], generated weights. (More complex algorithms did not produce better results.) These weights were then used as term weights for normal query processing.

Note that Wong, Yao, *et al* [9] previously used a similar approach, applying a variant of perceptron learning to learn weights with the SMART vector space model.

3.3.2 Full Context Vector Learning

This approach is similar to the previous Stem Weight approach, except we compute an entire query context vector rather than weights for stems. Here we use document context vectors directly as inputs to the network; for example the i^{th} network input for training example n is given by $\mathbf{V}^n_i$. The network weights produced by learning are directly interpreted as a query context vector.

For most topics, this approach was not as good as the previous approach for two apparent reasons. First the Stem Weight approach makes use of the original query terms, a valuable piece of user input. Second, the Stem Weight approach uses fewer (5–50) trainable parameters, possibly avoiding a tendency with the Full Context Vector approach ($\approx$ 300 parameters) to 'overfit the data'.

3.3.3 Routing Run #1: Best Candidate

Our first routing run was to take the best candidate from 4 sources: the two neural network approaches,

[3] Only terms in the concept section of the topic were used for routing experiments.

a fully automated query run (as with the first ad hoc submission), and a top 20 feedback run (as with the second ad hoc submission). For each topic, the best method was estimated from retrievals on a separate test corpus (not the corpus used for submissions). The winning method's list of 1000 retrievals was then selected as the retrievals for this topic.

3.3.4 Routing Run #2: Mix and Match

Our second routing run was to mix retrievals from the same 4 sources, using a 'quality estimate' consisting of 11-point recall/precision scores determined from a run on a separate test corpus. Each document in each of the four approaches was given points proportional to the run quality estimate and inversely proportional to its position number on an output list. Documents appearing on more than one list received points for each appearance.

Mix and Match worked better than the previous Best Candidate approach.

4 Comments

We are generally pleased with the performance from our one-year-old system's results. (For final figures, see the appendix of this proceedings.)

In examining the data, one interesting aspect is that *MatchPlus* does better when measured by 11-point averages than by number of relevant documents retrieved. This means a comparatively higher percentage of documents in *early retrievals* were judged relevant.[4]

We are now running initial experiments with word sense disambiguation using context vectors and clustering. It will be interesting to see whether word sense disambiguation can further improve retrieval performance.

References

[1] Gallant, S. I. Context Vector Representations for Document Retrieval. AAAI-91 Natural Language Text Retrieval Workshop, Anaheim, CA, July 15, 1991.

[2] Gallant, S. I. A Practical Approach for Representing Context And for Performing Word Sense Disambiguation Using Neural Networks. Neural Computation, Vol. 3, No. 3, 1991, 293-309.

[3] Gallant, S. I. Perceptron-Based Learning Algorithms. IEEE Transactions on Neural Networks, Volume 1, Number 2.

[4] Gallant, S. I. *Neural Network Learning and Expert Systems.* MIT Press, 1993.

[5] Hinton, G. E. Distributed Representations. Technical Report CMU-CS-84-157, Carnegie-Mellon University, Department of Computer Science. Revised version in Rumelhart, D. E. & McClelland, J. L. (Eds.) *Parallel Distributed Processing: Explorations in the Microstructures of Cognition, Vol. 1.* MIT Press, 1986.

[6] Salton, G. *The SMART retrieval system – Experiments in automatic automatic document processing.* Englewood Cliffs, NJ: Prentice-Hall, 1971.

[7] Salton, G. & Buckley, C. Term-Weighting Approaches in Automatic Text Retrieval. Information Processing & Management, Vol. 24, No. 5, 1988, pp. 513-523.

[8] Salton, G. & Buckley, C. Improving Retrieval Performance by Relevance Feedback. Journal of the American Society for Information Science, 41(4):288–297, 1990.

[9] Wong, S.K.M, Yao, Y.Y, Salton, G. & Buckley, C. Evaluation of an Adaptive Linear Model. Journal of the American Society for Information Science, 42(10):723–730, 1991.

[10] Waltz, D. L. & Pollack, J. B. Massively Parallel Parsing: A Strongly Interactive Model of Natural Language Interpretation. Cognitive Science 9, 51-74 (1985).

[4] This also raises the question as to whether comparatively fewer total documents were scored by the readers, perhaps due to the *MatchPlus* system's different approach. Computing the median number of documents scored per topic would resolve this question, as well as providing an indication of "novelty" for TREC participants.

Latent Semantic Indexing (LSI) and TREC-2

Susan T. Dumais
Bellcore
445 South St.
Morristown, NJ 07960
std@bellcore.com

1. Overview of Latent Semantic Indexing

Latent Semantic Indexing (LSI) is an extension of the vector retrieval method (e.g., Salton & McGill, 1983) in which the dependencies between terms are explicitly taken into account in the representation and exploited in retrieval. This is done by simultaneously modeling all the interrelationships among terms and documents. We assume that there is some underlying or "latent" structure in the pattern of word usage across documents, and use statistical techniques to estimate this latent structure. A description of terms, documents and user queries based on the underlying, "latent semantic", structure (rather than surface level word choice) is used for representing and retrieving information. One advantage of the LSI representation is that a query can be very similar to a document even when they share no words.

Latent Semantic Indexing (LSI) uses singular-value decomposition (SVD), a technique closely related to eigenvector decomposition and factor analysis (Cullum and Willoughby, 1985), to model the associative relationships. A large term-document matrix is decomposed it into a set of k, typically 100 to 300, orthogonal factors from which the original matrix can be approximated by linear combination. Instead of representing documents and queries directly as sets of independent words, LSI represents them as continuous values on each of the k orthogonal indexing dimensions. Since the number of factors or dimensions is much smaller than the number of unique terms, words will not be independent. For example, if two terms are used in similar contexts (documents), they will have similar vectors in the reduced-dimension LSI representation. The SVD technique can capture such structure better than simple term-term or document-document correlations and clusters. LSI partially overcomes some of the deficiencies of assuming independence of words, and provides a way of dealing with synonymy automatically without the need for a manually constructed thesaurus. LSI is a completely automatic method. (The Appendix provides a brief overview of the mathematics underlying the LSI/SVD method. Deerwester et al., 1990, and Furnas et al., 1988 present additional mathematical details and examples.)

One can also interpret the analysis performed by SVD geometrically. The result of the SVD is a vector representing the location of each term and document in the k-dimensional LSI representation. The location of term vectors reflects the correlations in their usage across documents. In this space the cosine or dot product between vectors corresponds to their estimated similarity. Retrieval typically proceeds by using the terms in a query to identify a point in the space, and all documents are then ranked by their similarity to the query. However, since both term and document vectors are represented in the same space, similarities between any combination of terms and documents can be easily obtained.

The LSI method has been applied to many of the standard IR collections with favorable results. Using the same tokenization and term weightings, the LSI method has equaled or outperformed standard vector methods and other variants in almost every case, and was as much as 30% better in some cases (Deerwester et al., 1990). As with the standard vector method, differential term weighting and relevance feedback both improve LSI performance substantially (Dumais, 1991). LSI has also been applied in experiments on relevance feedback (Dumais and Schmitt, 1991), and in filtering applications (Foltz and Dumais, 1992).

The recent MatchPlus system described by Gallant et al. (1992) is related to LSI. Both systems model the relationships between terms by looking at the similarity of the contexts in which words are used, and exploit these associations to improve retrieval. Both systems use a reduced dimension vector

representation, but differ in how the term, document and query vectors are formed.

2. LSI and TREC-1

We used the TREC-1 conference as an opportunity to "scale up" our tools, and to explore the LSI dimension-reduction ideas using a very rich corpus of word usage. We were pleased that we were able to use many of the existing LSI/SVD tools on the large collection. (See Dumais, 1993 for details.) We were able to compute the SVDs of 50k docs x 75k words matrices without numerical or convergence problems on a standard Dec5000 or Sparc10 workstation. Because of these limits on the size of the matrices we could handle, we divided the TREC-1 documents into 9 separate subcollections (AP1, DOE1, FR1, WSJ1, ZIFF1, AP2, FR2, WSJ2, ZIFF2) and worked with these. There are also some theoretical reasons why working with subcollections makes sense. By using topically coherent subcollections one can get better discrimination among documents. When the ZIFF subcollection is analyzed separately, for example, 200 or so dimensions can be devoted to differences among computer-related topics. When these same documents are part of a large corpus, many fewer indexing dimensions will be devoted to discriminating among them.

In terms of accuracy, LSI performance in TREC-1 was about average. There were some obvious problems with our initial pre-processing of documents (e.g., "U.S." and "A.T.T." were omitted), and there were some unanticipated problems in combining across subcollections. Since many of the top performing automatic systems used SMART's preprocessing, we chose to do so as well for TREC-2. In addition, using the SMART software for this purpose allows us to compare LSI with the comparable vector method, so that we can examine the contribution of LSI per se. We also planned on comparing an LSI analysis using subcollections (from TREC-1) with an LSI analysis of the entire collection for TREC-2.

We had high hopes of being able to build on our TREC-1 work for TREC-2. In practice, however, the changes in the pre-processing algorithm, and the decision to use a single combined LSI analysis resulted in our "starting from scratch" in many respects. We have completed some experiments for TREC-2, but we did not get as far as we would have liked, especially for the adhoc topics.

3. LSI and TREC-2

3.1 Pre-processing

We used the SMART system[1] for pre-processing the documents and queries. Some markups (e.g. <> delimiters) were removed, and all hand-indexed entries were removed from the WSJ and ZIFF collections. Upper case characters were translated into lower case, punctuation was removed, and white spaces were used to delimit terms. The SMART stop list of 571 words was used as is. The SMART stemmer (a modified Lovins algorithm) was used without modification to strip words endings. We did not use: phrases, proper noun identification, word sense disambiguation, a thesaurus, syntactic or semantic parsing, spelling checking or correction, complex tokenizers, a controlled vocabulary, or any manual indexing.

The result of this pre-processing can be thought of as a term-document matrix, in which each cell entry indicates the frequency with which a term appears in a document. The entries in the term-document matrix were then transformed using an "ltc" weighting. The "ltc" weighting takes the log of individual cell entries, multiplies each entry for a term (row) by the IDF weight of the term, and then normalizes the document (col) length.

We began by processing the 742358 documents from CD-1 and CD-2. Using the minimal pre-processing described above, there were 960765 unique tokens, 512251 unique stems, and 81901331 non-zero entries in the term-document matrix. 742331 documents contained at least one term. To decrease the matrix to a size we thought we could handle, we removed tokens occurring in fewer than 5 documents. This resulted in 781421 unique tokens, 104533 unique stemmed words, and 81252681 non-zero entries. We used the resulting 742331 document x 104533 term matrix as the starting point for results reported in this paper. The "ltc" weights were computed on this matrix.

3.2 SVD analysis

The "ltc" matrix described above was used as input to the SVD algorithm. The SVD program takes the ltc

1. The SMART system (version 11.0) was made available through the SMART group at Cornell University. Chris Buckley was especially generous in consultations about how to get the software to do somewhat non-standard things.

transformed term-document matrix as input, and calculates the best "reduced-dimension" approximation to this matrix. The result of the SVD analysis is a reduced-dimension vector for each term and each document, and a vector of the singular values. The number of dimensions, k, was between 200 and 300 in our experiments. This reduced-dimensional representation is used for retrieval. The cosine between term-term, document-document, or term-document vectors is used as the measure of similarity between them.

We were recently able to compute the SVD analysis of the full 742k x 104k matrix described above, but not in time to include the results in this paper. For the runs submitted, we used a sample of documents from the above matrix. When appropriate, the documents that were not sampled were "folded in" to the resulting reduced-dimension LSI space. In all cases, we used the weights from the 742k x 104k matrix (and did not recompute them for our samples).

For the **routing** experiments, we used the subset of documents for which we had relevance judgements. There were 68385 unique documents with relevance judgements. The SVD analysis was computed on the relevant 68385 document x 88112 term subset of the above matrix, containing 14461782 non-zero cells. A 204 reduced-dimensional approximation took 18 hrs of CPU time to compute on a Sparc10 workstation. This 204-dimensional representation was used for matching and retrieval.

For the **adhoc** experiments, we took a random sample of 70000 documents. A reduced-dimensional SVD approximation was computed on a 69997 document x 82968 term matrix (7666044 non-zeros). The resulting 199 reduced-dimensional representation was used for retrieval. The 672331 documents not included in this sample were "folded in" to the 199-dimension LSI space, and the adhoc queries were compared against all 742k documents.

These "folded in" documents were located at the weighted vector sum of their constituent terms. That is, the vector for a new document was computed using the term vectors for all terms in the document. For documents that are actually present in the term-document matrix, this derived vector corresponds exactly to the document vector given by the SVD. New terms can be added in an analogous fashion. The vector for new terms is computed using the document vectors of all documents in which the term appears. When adding documents and terms in this manner, we assume that the derived "semantic space" is fixed and that new items can be fit into it. In general, this is not the same space that one would obtain if a new SVD were calculated using both the original and new documents. In previous experiments, we found that sampling and scaling 50% of the documents, and "folding in" the remaining documents resulted in performance that was indistinguishable from that observed when all documents were scaled. Here, however, the scaling is based on less than 10% of the total corpus.

We also had (from TREC-1) LSI analyses of the 9 subcollections in CD-1 and CD-2.

3.3 Queries and retrieval

Queries were automatically processed in the same way as documents. For queries derived from the topic statement, we began with the full text of each topic (all topic fields), and stripped out the SGML field identifiers. For feedback queries, we used the full text of relevant documents. A query vector (or new document in the case of routing) indicating the frequency with which each term appears in the query was automatically generated for each topic. The query was transformed using SMART's "ltc" weighting.

Note that we did **not** use any Boolean connectors or proximity operators in query formulation. (The implicit connectives, as in ordinary vector methods, fall somewhere between ORs and ANDs, but with an additional kind of "fuzziness" introduced by the dimension-reduced association matrix representation of terms and documents.)

The terms in the query are used to identify a vector in the LSI space; recall that each term has a vector representation in the space. A query is simply located at the weighted vector sum of its constituent term vectors. The cosine between the query vector and every document vector is computed, and documents are ranked in decreasing order of similarity to the query. (Although there are many fewer dimensions than in standard vector retrieval, the entries are almost all non-zero so inverted indices are not useful. This means that each query must be compared to every document.)

For 200-dimensional vectors, about 60,000 cosines can be computed per minute on a Sparc2. This time includes both comparison time and ranking time, and assumes that all document vectors are pre-loaded into memory. For the adhoc queries, the time to compare a query to the 743k documents was about 10 minutes if

all comparisons were sequential. It is, however, straightforward to split this matching across several machines or to use parallel hardware since all documents are independent. Preliminary experiments using a 16,000 PE MasPar showed that 60,000 cosines could be computed and sorted in less than 1 second.

It is important to note that all step in the LSI analysis are **completely automatic** and involved no human intervention. Documents are automatically processed to derive a term-document matrix. This matrix is decomposed by the SVD software, and the resulting reduced-dimension representation is used for retrieval. While the SVD analysis is somewhat costly in terms of time for large collections, it need is computed only once at the beginning to create the reduced-dimension database. (The SVD takes only about 2 minutes on a Sparc10 for a 2k x 5k matrix, but this time increases to about 18 hours for a 60k x 80k matrix.)

3.4 TREC-2: Routing experiments

For the routing queries, we created a filter or profile for each of the 50 training topics. We submitted results from two sets of routing queries. In one case, the filter was based on just the topic statements - i.e., we treated the routing queries as if they were adhoc queries. The filter was located at the vector sum of the terms in the topic. We call these the **routing_topic** (**lsir1**) results. This method makes **no** use of the training data, representing the topic as if it was an adhoc query. In the other case, we used information about relevant documents from the training set. The filter in this case was derived by taking the vector sum or centroid of all relevant documents. We call these the **routing_reldocs** (**lsir2**) results. There were an average of 328 relevant documents per topic, with a range of 40 to 896. This is a somewhat unusual variant of relevance feedback; we *replace* (rather than combine) the original topic with relevant documents, and we do not downweight terms that appear in non-relevant documents. These two extremes provide baselines against which to compare other methods for combining information from the original query and feedback about relevant documents. In both cases, the filter was a single vector. New documents were matched against the filter vector and ranked in decreasing order of similarity.

The new documents (336306 documents from CD-3) were automatically processed as described in section 3.2 above. It is important to note that only terms from the CD-1 and CD-2 training collection were used in indexing these documents. Each new document is located at the weighted vector sum of its constituent term vectors in the 204-dimension LSI space (in just the same way as queries are handled). New documents were compared to each of the 50 routing filter vectors using a cosine similarity measure in 204-dimensions. The 1000 best matching documents for each filter were submitted to NIST for evaluation.

3.4.1 Results

The main results of the lsir1 and lsir2 runs are shown in Table 1. The two runs differ only in how the profile vectors were created - using the weighted average of the words in the topic statement for **lsir1 routing_topic**, and using the weighted average of all relevant documents from the training collection (CD-1 and CD-2) for **lsir2 routing_reldocs**. Not surprisingly, the lsir2 profile vectors which take advantage of the known relevant documents do better than the lsir1 profile vectors that simply use the topic statement on all measures of performance. The improvement in average precision is 31% (.2622 vs. .3442). Users would get an average of 1 additional relevant document in the top 10 returned using the lsir2 method for filtering.

Table 1

	lsir1 (topic wds)	lsir2 (rel docs)	r1+r2 (sum r1 r2)
Rel_ret	6522	7155	7367
Avg prec	.2622	.3442	.3457
Pr at 100	.3799	.4524	.4394
Pr at 10	.5480	.6660	.6620
R-prec	.3050	.3804	.3786
Q >= Median	27 (4)	40 (9)	42 (6)
Q < Median	23 (0)	10 (0)	8 (0)

Table 1: LSI Routing Results. Comparison of topic words vs. relevant documents as routing filters.

Compared to other TREC-2 systems, LSI does reasonably well, especially for the **routing_reldocs** (**lsir2**) run (and the r1+r2 run to be discussed below). In the case of lsir2, LSI is at or above the median performance for 40 of the 50 topics, and has the best score for 9 topics. LSI performs about average for the **routing_topic** (**lsir1**) run even though no information from the training set was used in forming the routing vectors in this case (except, of course, for the global term weights).

We have also performed similar comparisons between

query vectors representing the words in queries and the centroid of all relevant documents for some of the standard IR test collections (Med, CISI, Cranfield, CACM, Time). In these cases, we found an average improvement of 107% when the query was replaced by the centroid of all relevant documents. The improvement was 67% when the top three relevant documents were used, and 33% when just the first relevant document was used. The smaller advantages observed in TREC-2 are partially due to statistical artifacts, and partially to the TREC topics which are much richer need statements than the usual IR queries. (We also examined topic and reldocs profiles in TREC-1. Somewhat surprisingly, the query using just the topic terms was about 25% more accurate than the query using relevant documents from training. This is attributable to the small number and inaccuracy of relevance judgements in the initial training set for TREC-1. This had substantial impact on performance for some topics because our reldocs queries were based only on the relevant articles and ignored the original topic description.)

The lsir1 and lsir2 runs provide baselines against which various combinations of query information and relevant document information can be measured. We have tried a simple combination of the lsir1 and lsir2 profile vectors, in which both components have equal weight. That is, we took the sum of the lsir1 and lsir2 profile vectors for each of the topics and used this as a profile vector. The results of this analysis are shown in the third column of the table labeled r1+r2. This combination does somewhat better than the centroid of the relevant documents in the total number of relevant documents returned and in average precision. (We returned fewer than 1000 documents for 5 of the topics and not all documents returned by the r1+r2 method had been judged for relevance, so we suspect that performance could be improved a bit more.) For 27 of the topics, r1+r2 was better than the maximum of the other two methods. It was never more than about 10% worse than the best method. Thus it appears that this combination takes advantage of the best of both methods.

The r1+r2 method which combines a query vector with a vector representing the centroid of all relevant documents is a kind of *relevance feedback*. This is an unusual variant of relevance feedback since *all* the words in relevant documents are used, words in non-relevant documents are not down-weighted, and query terms are not re-weighted. Interestingly, this method appears to produce improvements that are comparable to those obtained by Buckley, Allan and Salton (1993) using more traditional relevance feedback methods. Average precision for the r1+r2 method is 31% better than for lsir1 which used only the topic words (.3457 vs. .2622), and this is quite similar to the 38% improvement reported by Buckley, Allan and Salton (1993) for their richest routing query expansion method.

The lsir2 method is generally better than the lsir1 method, but there is substantial variability across topics. The topics on which there are the largest differences are generally those in which the cosine between the the lsir1 and lsir2 topic vectors are smallest. The cosines between corresponding topic vectors range from .87 to .54. The lsir2 method is substantially better on topics: 71 (incursions by foreign military or guerrilla groups), 73 (movement of people from one country to another), 87 (criminal actions against officers of failed financial institution), 94 (crime perpetrated with the aid of a computer), 98 (production of fiber optics equipment). There are a few topics for which lsir1 is substantially better than lsir2: 63 (machine translation system), 65 (information retrieval system), 85 (actions against corrupt public officials), 95 (computer application to crime solving). It is not entirely clear what distinguishes between these topics, especially topics 94 and 95, for example.

We have not yet had time to look in detail at the failures of the LSI system. We will examine both *misses* and *false alarms* in more detail. A preliminary examination of a few topics suggests that lack of specificity is the main reason for false alarms (highly ranked but irrelevant documents). This is not surprising because LSI was designed as a recall-enhancing method, and we have not added precision-enhancing tools although it would be easy to do so.

We would also like to examine some query splitting ideas. We have previously conducted experiments which suggest that performance can be improved if the filter is represented as several separate vectors. We did not use this method for the TREC-2 results we submitted, but would like to do so. (See also Kane-Esrig et al., 1991 or Foltz and Dumais, 1992, for a discussion of multi-point interest profiles in LSI.)

3.5 TREC-2: Adhoc experiments

We submitted two sets of adhoc queries - lsiasm and lsia1. We had intended to compare the new SMART pre-processing (lsiasm) and a single LSI space (lsia1) with our old TREC-1 pre-processing and 9 separate subcollection spaces. Unfortunately, there were some serious errors in our translation between internal document numbers and the <DOCNO> labels

(documents not in the LSI scaling were mislabeled), so the lsia1 run results are incomplete and misleading. We have corrected this translation problem, and the correct results are labeled lsia1*. These results are summarized in Table 2. We have not yet completed the comparison against the 9 separate subspaces from TREC-1.

Table 2

	lsiasm	lsia1 error	lsia1* correct
Rel_ret	7869	4756	6987
Avg prec	.3018	.1307	.2505
Pr at 100	.4306	.2664	.3922
Pr at 10	.5020	.3340	.5100
R-prec	.3580	.1937	.3069
Q >= Median	37 (2)	16 (1)	25 (1)
Q < Median	13 (0)	34 (7)	25 (0)

Table 2: LSI Adhoc Results. Comparison of standard vector method with LSI (corrected version, but missing relevance judgements).

In terms of absolute levels of performance, both lsiasm and lsia1* are about average. The SMART results (lsiasm) are somewhat worse than the TREC-2 SMART results reported by Buckley et al., Fuhr et al., or Voorhees, but this is because we used slightly different pre-processing options and did not include phrases. Although it is generally difficult to compare across systems, the SMART (lsiasm) and LSI (lsia1*) runs can meaningfully be compared since both use the same pre-processing. The starting term-document matrix was the same in both cases. Much to our disappointment, the reduced-dimension LSI performance appears to be somewhat worse than the comparable SMART vector method. However, it is important to realize that many of the documents returned by lsia1* were not judged for relevance because they were not submitted as an official run. Table 3 shows the number of documents for which there are no judgements. Consider the results for just the top100 documents for each query (i.e., the documents judged by the NIST assessors). For lsiasm, all 5000 documents were judged since this was an official run, and 2153 were relevant. For lsia1*, only 4073 documents were judged and almost as many, 2122, were relevant. Thus, if only 31 of the 927 unjudged lsia1* documents are relevant LSI performance would be comparable to SMART performance, and if more than 31 were relevant LSI performance would be somewhat better. Similarly for the top1000 documents, lsia1* had more than 4000 more documents without relevance judgements than did lsiasm.

Table 3

	lsiasm top100	lsia1* top100	lsiasm top1000	lsia1* top1000
relevant	2153	2122	15559	12230
not-relevant	2847	1961	7869	6987
not-judged	0	927	26572	30694

Table 3: Summary of missing relevance judgements for standard vector method and LSI.

Because the missing relevance judgements make direct comparisons between SMART and LSI difficult, we decided to look at performance for just the documents for which we had relevance judgements. That is, we looked at performance considering just the 38175 unique documents for which we have adhoc relevance judgements. These results are shown in Table 4.

Table 4

	lsiasm 38175	lsia1* 38175
Rel_ret	9493	9596
Avg prec	.3700	.3789
Pr at 100	.4306	.4466
Pr at 10	.5020	.5220
R-prec	.3977	.3995

Table 4: LSI Adhoc Results. Comparison of standard vector method with LSI using only documents for which relevance judgements were available.

The most striking aspect of these results is the higher overall levels of performance. This is to be expected since we are only considering the 38175 documents for which we have relevance judgements, and there are 700k fewer documents than in the official results. Considering only this subset of documents, there is a small advantage for LSI compared to the SMART vector method. Taken together with the results for just the top 100 documents these results suggest that LSI can outperform a straightforward vector method. We were somewhat disappointed at the relatively

small difference between LSI and a comparable vector method in the TREC environment, given that we have consistently observed larger advantages previously. The most likely reason for this is that the TREC topics are much richer and more detailed descriptions of searchers' needs than are found in typical IR requests. The average TREC query has 51 unique words in it, and many of these are very specific names. Since LSI is primarily a recall enhancing method it has little effect on these already very rich queries. This is much the same conclusion that groups who tried various kinds of query expansion (another recall enhancing method) reached - e.g., see Voorhees 1993.

We tried one analysis using the new "summary" field for each topic. This field alone is used as a much shorter topic statement (often quite similar to the description field) that covers the relevance assessments. These results are summarized in Table 5. As expected, overall performance is lower than with the complete topics. More interestingly, the difference between LSI and the standard vector method is now larger - 16% in average precision. This is still a somewhat smaller advantage than we have seen in previous experiments with smaller test collections, but even the summary queries have an average of 11 unique words in them.

Table 5

	lsiasm 38175	**lsia1* 38175**
Rel_ret	8043	8676
Avg prec	.2589	.3008
Pr at 100	.3344	.3710
Pr at 10	.3420	.3680
R-prec	.3114	.3386

Table 5: LSI Adhoc Results - Summary Topics. Comparison of standard vector method with LSI using only documents for which relevance judgements were available, using only the Summary field.

The reduced dimension LSI vector retrieval method can offer performance advantages compared to a standard vector method for the large TREC collection. The advantages are larger with shorter queries, as expected. The exact nature of this advantage (e.g., which documents are retrieved by LSI but not the standard vector method) needs to be examined in more detail.

4. Improving performance

4.1 Improving performance - Speed

The LSI/SVD system was built as a research prototype to investigate many different information retrieval and interface issues. Retrieval efficiency was not a central concern because we first wanted to assess whether the method worked before worrying about efficiency, and because the initial applications of LSI involved much smaller databases of a few thousand documents. Almost no effort went into re-designing the tools to work efficiently for the large TREC databases.

4.1.1 SVD

SVD algorithms get faster all the time. The sparse, iterative algorithm we now use is about 100 times faster than the method we used initially (Berry, 1992). There are the usual speed-memory tradeoffs in the SVD algorithms, so time can probably be decreased some by using a different algorithm and more memory. Parallel algorithms will help a little, but probably only by a factor of 2. Finally, all calculations are now done in double precision, so both time and memory could be decreased by using single precision computations. Preliminary experiments with smaller IR test collections suggest that this decrease in precision will not lead to numerical problems for the SVD algorithm. It is important to note that the pre-processing and SVD analyses are one-time-only costs for relatively stable domains.

4.1.2 Retrieval

Query vectors are compared to *every* document. This process is linear in the number of documents in the database, and can be quite slow for large databases. Although there are no practical and efficient algorithms for finding nearest neighbors in 200- or 300-dimensional spaces, there are several methods which could be used to speed retrieval. a) Document clustering could be used to reduce the number of comparisons, but accuracy would probably suffer some. We have explored several heuristics for clustering, but none are particularly effective when high levels of accuracy are maintained. b) HNC's MatchPlus system (Gallant et al., 1993) uses another approach to reduce the number of alternative documents that must be matched. They use an initial keyword match to eliminate many documents and calculate reduced-dimension vector scores for only the subset of documents meeting the initial keyword restriction. This may be a reasonable alternative for long queries (like TREC), but we believe that recall would be reduced for short queries. In addition two

data structures need to be maintained. c) Query matching can also be improved tremendously by simply using more than one machine or parallel hardware. Using a 16,000 PE MasPar, with no attempt to optimize the data storage or sorting, we decreased the time required to match a 200-dimensional query vector against all document vectors and sort by a factor of 60 to 100.

4.2 Improving Performance - Accuracy

We have only begun to look at a large number of parametric variations that might improve LSI performance. One important variable for LSI retrieval is the number of dimensions in the reduced dimension space. In previous experiments we have found that performance improves as the number of dimensions is increased up to 200 or 300 dimensions, and decreases slowly after that to the level observed for the standard vector method (Dumais, 1991). We have examined TREC-2 performance using *fewer dimensions* than reported above (204 for the routing queries and 199 for the adhoc queries) and consistently found worse performance. Thus, it looks like we could improve performance simply by increasing the number of dimensions some. Unfortunately, this requires rerunning the SVD.

We also noticed that many of the adhoc queries contained "NOTS". Since LSI does not use any Boolean logic and represents a query as the vector sum of its constituent terms, we thought that removing this information might help. We modified the topic statements by hand to remove negated phrases. Performance improved by less that 2%.

We still need to experiment with different term weighting methods. For the routing and adhoc experiments we used SMART's "ltc" weighting for both the corpus of documents and the queries. Buckley and Salton's TREC-1 paper suggests that alternative weightings may be more effective for the large TREC document collection. Reweighting the query vectors is easy. Reweighting the document collection is more difficult, because this changes the term-document matrix and a new SVD is required.

For the routing queries we would like to try several alternative methods of combining information from the original query and the relevant documents to take better advantage of the good training data that is available. We expect term re-weighting and the use of negative information (e.g., down weighting terms from non-relevant documents) to improve performance some.

In order to better understand retrieval performance we have begun to examine two kinds of retrieval failures: false alarms, and misses. *False alarms* are documents that LSI ranks highly that are judged to be irrelevant. *Misses* are relevant documents that are not in the top 1000 returned by LSI.

4.2.1 False Alarms.

The most common reason for false alarms was lack of specificity. These highly ranked but irrelevant articles were generally about the topic of interest but did not meet some of the restrictions described in the topic statement. Many topics required this kind of detailed processing or fact-finding that the LSI system was not designed to address. Precision of LSI matching can be increased by many of the standard techniques - proper noun identification, use of syntactic or statistically-derived phrases, or a two-pass approach involving a standard initial global matching followed by a more detailed analysis of the top few thousand documents. Buckley and Salton (1992, SMART's global and local matching), Evans et al. (1992, CLARIT's evoke and discriminate strategy), Nelson (1992, ConQuest's global match followed by the use of locality of information), and Jakobs, Krupka and Rau (1992, GE's pre-filter followed by a variety of more stringent tests) all used two-pass approaches to good advantage in TREC-1 or TREC-2. We would like to try some of these methods, and will focus on general-purpose, completely automatic methods that do not have to be modified for each new domain or query restriction.

Another possible reason for false alarms appears to be the result of inappropriate query pre-processing. The use of negation is the best example of this problem. 32 of 50 adhoc queries contain some negation in the topic statement. Some preliminary experiments (described briefly above) found only a small improvement in performance when negated information was manually removed from the topics. Another example of inappropriate query processing involved the use of logical connectives. LSI does not handle Boolean combinations of words, and often returned articles covering only a subset of ANDed topics. Often one aspect of the query appears to dominate (typically the one described by the terms with high weights). Limiting the contribution of any one term to the overall similarity score might help this problem.

Finally, it is not at all clear why about 20% of the false alarms were returned by LSI. Since LSI uses a statistically-derived "semantic" space and not surface-level word overlap for matching queries to

documents, it is sometimes difficult to understand why a particular document was returned. One advantage of the LSI method is that documents can match queries even when they have no words in common; but this can also produce some spurious hits. Another reason for false alarms could be inappropriate word sense disambiguation. LSI queries are located at the weighted vector sum of the words, so words are "disambiguated" to some extent by the other query words. Similarly, the initial SVD analysis used the context of other words in articles to determine the location for each word in the LSI space. However, since each word has only one location, it sometimes appears as if it is "in the middle of nowhere". A related possibility concerns long articles. Lengthy articles which talk about many distinct subtopics were averaged into a single document vector, and this can sometimes produce spurious matches. Breaking larger documents into smaller subsections and matching on these might help.

4.2.2 Misses.

For this analysis we will examine a random subset of relevant articles that were not in the *top 1000* returned by LSI. Many of the relevant articles were fairly highly ranked by LSI, but there were also some notable failures that would be seen only by the most persistent readers. So far, we have not systematically distinguished between misses that "almost made it" and those that were much further down the list.

Most of the misses we examined, represent articles that were primarily about a different topic than the query, but contained a small section that was relevant to the query. Because documents are located at the average of their terms in LSI space, they will generally be near the dominant theme, and this is a desirable feature of the LSI representation. Some kind of local matching should help in identifying less central themes in documents.

Some misses were also attributable to poor text (and query) pre-processing and tokenization.

4.3 Open issues

On the basis of preliminary failure analyses we would like to exploring some precision-enhancing methods. We would also like to explore three additional areas.

4.3.1 Separate vs. combined scaling

We used 9 separate subscalings for the TREC-1 experiments. For TREC-2 we used a single scaling (based on a very small sample). We have also recently finished a complete scaling and will compare this with the subcollection scalings and the sampled full scaling.

4.3.2 Centroid query vs. many separate points of interest

A single vector was used to represent each query. In some cases the vector was the average of terms in the topic statement, and in other cases the vector was the average of previously identified relevant documents. A single query vector can be inappropriate if interests are multifacted and these facets are not near each other in the LSI space. We have developed techniques that allow us to match using a controllable compromise between averaged and separate vectors (Kane-Esrig et al., 1991). In the case of the routing queries, for example, we could match new documents against each of the previously identified relevant documents separately rather than against their average.

4.3.3 Interactive interfaces

All LSI evaluations were conducted using a non-interactive system in essentially batch mode. It is well known that one can have the same underlying retrieval and matching engine, but achieve very different retrieval success using different interfaces. We would like to examine the performance of real users with interactive interfaces. A number of interface features could be used to help users make faster (and perhaps more accurate) relevance judgements, or to help them explicitly reformulate queries. (See Dumais and Schmitt, 1991, for some preliminary results on query reformulation and relevance feedback.) Another interesting possibility involves returning something richer than a rank-ordered list of documents to users. For example, a clustering and graphical display of the top-k documents might be quite useful. We have done some preliminary experiments using clustered return sets, and would like to extend this work to the TREC collections.

The general idea is to provide people with useful interactive tools that let them make good use of their knowledge and skills, rather than attempting to build all the smarts into the database representation or matching components of the system.

5. Onward to TREC-3

We were quite pleased that we were able to use many of the existing LSI/SVD tools on the TREC-1 and TREC-2 collections. The most important finding in this regard was that the large, sparse SVD problems could be computed without numerical or convergence problems. We modified the preprocessing substantially for TREC-2, now have many of the basic tools in place and should be able to conduct more experiments comparing various indexing and query matching ideas using the same underlying LSI engine.

Bigger SVDs, faster query matching, improving precision, and interactive interface issues are the major areas targeted for improvement.

6. References

[1] Berry, M. W. Large scale singular value computations. *International Journal of Supercomputer Applications*, 1992, *6(1)*, 13-49.

[2] Buckley, C., Allan, J., and Salton, G. Automatic routing and ad-hoc retrieval using SMART: TREC 2. To appear in: *Proceedings of TREC-2*.

[3] Buckley, C. and Salton, G. Automatic retrieval with locality information using SMART. In D. Harman (Ed.) *The First Text REtrieval Conference (TREC-1), NIST Special Publication 500-207*, 1992, 59-72.

[4] Cullum, J.K. and Willoughby, R.A. *Lanczos algorithms for large symmetric eigenvalue computations - Vol 1 Theory*, (Chapter 5: Real rectangular matrices). Brikhauser, Boston, 1985.

[5] Deerwester, S., Dumais, S. T., Landauer, T. K., Furnas, G. W. and Harshman, R. A. Indexing by latent semantic analysis. *Journal of the Society for Information Science*, 1990, *41(6)*, 391-407.

[6] Dumais, S. T. Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments and Computers*, 1991, *23(2)*, 229-236.

[7] Dumais, S. T. LSI meets TREC: A staus report. In D. Harman (Ed.) *The First Text REtrieval Conference (TREC-1)*. NIST special publication 500-207, 137-152.

[8] Dumais, S. T. and Schmitt, D. G. Iterative searching in an online database. In *Proceedings of Human Factors Society 35th Annual Meeting*, 1991, 398-402.

[9] Evans, D., Lefferts, R., Grefenstette, G., Handerson, S., Hersh, W., and Archbold, A. CLARIT TREC Design, experiments and results. In D. Harman (Ed.) *The First Text REtrieval Conference (TREC-1), NIST Special Publication 500-207*, 1992, 251-286.

[10] Foltz, P. W. and Dumais, S. T. Personalized information delivery: An analysis of information filtering methods. *Communications of the ACM*, Dec. 1992, *35(12)*, 51-60.

[11] Furnas, G. W., Deerwester, S., Dumais, S. T., Landauer, T. K., Harshman, R. A., Streeter, L. A., and Lochbaum, K. E. Information retrieval using a singular value decomposition model of latent semantic structure. In *Proceedings of SIGIR*, 1988, 465-480.

[12] Gallant, S., Hecht-Nielson, R., Caid, W., Qing, K., Carleton, J., Sudbeck, D. TIPSTER Panel - HNC's MatchPlus System. In D. Harman (Ed.) *The First Text REtrieval Conference (TREC-1), NIST Special Publication 500-207*, 1992, 107-112.

[13] Jacobs, P, Krupka, G., and Rau, L. A Boolean approximation method for query construction and topic assignment in TREC. In D. Harman (Ed.) *The First Text REtrieval Conference (TREC-1), NIST Special Publication 500-207*, 1992, 297-308.

[14] Kane-Esrig, Y., Streeter, L., Dumais, S. T., Keese, W. and Casella, G. The relevance density method for multi-topic queries in information retrieval. In *Proceedings of the 23rd Symposium on the Interface*, E. Keramidas (Ed.), 1991, 407-410.

[15] Nelson, P. Site report for the Text REtrieval Conference. In D. Harman (Ed.) *The First Text REtrieval Conference (TREC-1), NIST Special Publication 500-207*, 1992, 287-296.

[16] Salton, G. and McGill, M.J. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[17] Voorhees, E. On expanding query vectors with lexically related words. To appear in: *Proceedings of TREC-2*.

7. Appendix

Latent Semantic Indexing (LSI) uses singular-value decomposition (SVD), a technique closely related to eigenvector decomposition and factor analysis (Cullum and Willoughby, 1985). We take a large term-document matrix and decompose it into a set of k, typically 100 to 300, orthogonal factors from which the original matrix can be approximated by linear combination.

More formally, any rectangular matrix, X, for example a $t \times d$ matrix of terms and documents, can be decomposed into the product of three other matrices:

$$\underset{t \times d}{X} = \underset{t \times r}{T_0} \cdot \underset{r \times r}{S_0} \cdot \underset{r \times d}{D_0'},$$

such that T_0 and D_0 have orthonormal columns, S_0 is diagonal, and r is the rank of X. This is so-called *singular value decomposition* of X.

If only the k largest singular values of S_0 are kept along with their corresponding columns in the T_0 and D_0 matrices, and the rest deleted (yielding matrices S, T and D), the resulting matrix, $\hat{X}$, is the unique matrix of rank k that is closest in the least squares sense to X:

$$\underset{t \times d}{X} \approx \underset{t \times d}{\hat{X}} = \underset{t \times k}{T} \cdot \underset{k \times k}{S} \cdot \underset{k \times d}{D'}.$$

The idea is that the $\hat{X}$ matrix, by containing only the first k independent linear components of X, captures the major associational structure in the matrix and throws out noise. It is this reduced model, usually with $k \approx 100$, that we use to approximate the term to document association data in X. Since the number of dimensions in the reduced model (k) is much smaller than the number of unique terms (t), minor differences in terminology are ignored. In this reduced model, the closeness of documents is determined by the overall pattern of term usage, so documents can be near each other regardless of the precise words that are used to describe them, and their description depends on a kind of consensus of their term meanings, thus dampening the effects of polysemy. In particular, this means that documents which share no words with a user's query may still be near it if that is consistent with the major patterns of word usage. We use the term "semantic" indexing to describe our method because the reduced SVD representation captures the major associative relationships between terms and documents.

One can also interpret the analysis performed by SVD geometrically. The result of the SVD is a k-dimensional vector representing the location of each term and document in the k-dimensional representation. The location of term vectors reflects the correlations in their usage across documents. In this space the cosine or dot product between vectors corresponds to their estimated similarity. Since both term and document vectors are represented in the same space, similarities between any combination of terms and documents can be easily obtained. Retrieval proceeds by using the terms in a query to identify a point in the space, and all documents are then ranked by their similarity to the query. We make no attempt to interpret the underlying dimensions or factors, nor to rotate them to some intuitively meaningful orientation. The analysis does not require us to be able to describe the factors verbally but merely to be able to represent terms, documents and queries in a way that escapes the unreliability, ambiguity and redundancy of individual terms as descriptors.

Choosing the appropriate number of dimensions for the LSI representation is an open research question. Ideally, we want a value of k that is large enough to fit all the real structure in the data, but small enough so that we do not also fit the sampling error or unimportant details. If too many dimensions are used, the method begins to approximate standard vector methods and loses its power to represent the similarity between words. If too few dimensions are used, there is not enough discrimination among similar words and documents. We find that performance improves as k increases for a while, and then decreases (Dumais, 1991). That LSI typically works well with a relatively small (compared to the number of unique terms) number of dimensions shows that these dimensions are, in fact, capturing a major portion of the meaningful structure.

An Information Retrieval Test-bed on the CM-5

Brij Masand and Craig Stanfill
Thinking Machines Corporation
245 First Street
Cambridge MA 02142

I. INTRODUCTION

For many years, research on information retrieval was mostly confined to a few relatively small test collections such as the Cranfield collection [1], the NPL Collection [2], and the CACM Collection [3]. Over the years, results on those collections accumulated, with the aim of determining which technique or combination of techniques resulted in the best precision/recall figures on those collections. Gradually, a "standard model" more-or-less emerged: for the test collections under study, consistently good results are obtained by vector-model retrieval using a cosine similarity measure, *tf.idf* weighting, and a stemming algorithm (e.g. Chapter 9 of [4], [5]).[1] Out-performing this model on the old test collections has proved extremely difficult. This has led to a danger of stagnation in the field of IR, and a feeling that the majority of what can be learned from precision-recall experiments on the old collections has been learned.

Fortunately for the field, a number of recent developments have led to new challenges for the field. Among these:

- The Tipster project [6] has led to the construction of a test collection of unprecedented size [7]. This leads to challenges related to scaling up IR methods.
- IR methods are now being used on full documents (e.g. news stories), rather than on abstracts. This leads to new challenges relating to the structure of large documents, particularly effects relating to term proximity.
- The on-line database vendors have shown interest in full-text IR. This leads to challenges relating to the integration of IR methods with traditional boolean methods which, in the operational environment, must continue to be supported if only for the benefit of the existing, entrenched user community.
- Network-oriented IR protocols have become increasingly popular as a basic tool for navigating across the network. This leads to the challenge of designing systems which give uniformly interpretable results across a distributed database.

The work reported in this paper represents some steps along the way to solving these problems. In essence, the challenge we are facing is to design a system which delivers high precision-recall figures on large databases of long, complex documents. Furthermore, the system must be compatible with existing boolean retrieval methods, and it must be possible to use this system in a distributed database in a manner which yields consistently interpretable results. The specific steps reported at this point consist of the following:

- A database architecture which is suitable for use on large collections of structured documents, which supports both base-line IR and boolean retrieval methods.
- An in-memory implementation of this architecture on a massively parallel computer (the Connection Machine model CM-5), which is used as test-bed.
- Precision-Recall figures for this test-bed applied to the Tipster collection, as part of TREC.

It must be emphasized that this work is in an early stage and, at this point, has not reached the point where these methods are demonstrable on extremely large databases. Furthermore, work relating to taking advantage of document structure has only barely begun.

II. DATABASE ARCHITECTURE

A. Choice of a Representation

There are three basic approaches to representing databases for retrieval applications:

- **Text Files.** In this method, the database is stored in full-text form, and scanned sequentially. Such methods are often implemented by special-purpose hardware [8].
- **Signature Files.** In this family of methods, a compressed surrogate for the text file is searched instead of the text file itself. Overlap encoding is usually employed to construct the surrogate. There are many variants on signature files [9].
- **Inverted Files.** In this family of methods, an index containing the locations of every word in the database is constructed. The search is then accomplished by reference to

1. Other approaches yield results which are nearly as good and, in rare cases better. Approaches based on Bayesian inference networks are particularly notable in that respect. However, the above model has come to be accepted as a de-facto standard against which new methods are measured.

the index. Again, there are many variants on inverted file representations [10].

Inverted files have proved the only technique which supports interactive access to very large databases; this is primarily due to the excessive I/O requirements of the other methods. Within the family of inverted file methods, several variants are possible:

- **Simple Inverted Files.** In a simple inverted file, the index consists of a list of documents in which each word occurs.
- **Inverted Files with Weights.** In an inverted file with weights, the above information is supplemented with weights, in support of vector retrieval methods.
- **Structured Inverted Files.** In a structured inverted file, the index captures structural information (typically the paragraph/sentence/word coordinates of each occurrence of each term), in support of boolean retrieval methods which rely on proximity operators (e.g. word adjacency).

In this research, we have decided to explore the use of structured inverted files for the following reasons:

- They support the proximity operations required as part of boolean retrieval systems. This support makes architectures based on structured inverted files more likely to be adopted by the major on-line vendors.
- They provide a fundamentally richer representation of document structure than is available with the other methods.
- They are collection-independent and retrieval-method independent.

This last point requires some explanation. In a distributed environment, it would be useful to be able to search text files at multiple locations as if they were a single text file. Once weights — which are both collection-dependent and retrieval-method dependent — are incorporated into the index, transparent distributed access becomes impossible. The product of this is that the results of a single query applied to multiple databases cannot be meaningfully combined, since there is no way to compare the ranks or scores applied to the documents returned.

One of the primary challenges associated with the use of structured inverted files is their bulk: there are approximately as many index entries in the inverted file as there are words in the database, and each entry must represent a document, paragraph, sentence, and word coordinates for that word. This problem has been substantially solved by use of a novel combination of compression techniques [11], which allow structured indexes having a bulk on the order of 1/3 the size of the full text to be constructed.

The second challenge associated with the use of structured inverted files is to implement the standard information retrieval model using them. The results of this effort are reported in this paper.

The final challenge associated with structured inverted files is using them to implement methods which go beyond the standard model, taking into account the added richness the structured representation to improve retrieval system performance for databases having long, structured documents. This final challenge remains a topic for future research, and there are no results to report at this time.

B. The Database Architecture

The database consists of the following structures:

- **A Compressed Structured Posting File.** The first component of the database is an array of compressed postings. Each posting gives the location of a word expressed as a document-paragraph-sentence-word 4-tuple. The postings are sorted by word ID, in ascending document order.
- **A Lexicon/Index.** The second component of the database is a lexicon/index. For each word in the database, it stores the number of times it occurs plus the location of its postings in the posting file. The lexicon is structured so that the information pertaining to a given word may be quickly located.
- **Document Information.** The final component of the database is an array of document-related information. Each entry contains a mapping from an internal document identifier (an integer) to an external document identifier (a string). Additional information, such as the length of the document or normalization information, may be added as necessary.

The following operations are supported:

- **Adding Postings.** A 5-tuple consisting of a word plus its four coordinates may be added to the database.
- **Extracting Postings.** A word is presented to the database. An array having the decompressed coordinates of all occurrences of that word is returned. The array is sorted by ascending document coordinate, with paragraph, sentence, and word coordinates used as additional sort keys as needed.
- **Extracting Lexicon Information.** Lexicon information such as the number of occurrences of a word may be extracted.
- **Setting Lexicon Information.** Supplemental lexicon information (e.g. part-of-speech information) may be added to the lexicon. This feature is not used in the work reported herein.
- **Defining a Document.** Defines document-specific information such as external document identifier, document location on disk, etc., and associates it with a document coordinate.
- **Extracting Document Information.** Extracts the document-specific information mentioned above, given a document coordinate.

We believe that this architecture suffices to implement boolean retrieval methods, standard IR methods, and novel methods making use of document structure, and that it can be scaled to very large databases and to massively parallel computers.

C. The Test-bed

The above architecture has been implemented in test-bed form on the Connection Machine model CM-5. The purpose of this implementation is to permit various retrieval methods to be evaluated, rather than to support on-line services on very large files. As such, the focus was on simplicity of implementation, speed of database construction, and speed of query execution, rather than on handling files larger than a few Gigabytes.

The CM-5 [12] is a general-purpose parallel computer constructed from commodity microprocessors. Each processing node consists of a SPARC microprocessor, 32 megabytes of RAM, and a network interface. The CM-5 has two user-accessible networks: a *Data Router*, which is used for point-to-point transmission of data packets, and a *Control Network* which is used to implement global operations such as barrier synchronization, broadcast, and global-maximum. The data router provides each processor with 5 MB/second (full duplex) worth of point-to-point bandwidth. The control network is constructed using a fan-in/fan-out tree so that global operations complete within a few microseconds of their initiation.

The CM-5 I/O system consists of a high-performance mass storage system called the *scalable disk array* (*SDA*). In this system, disk controllers are connected directly to the CM-5's data router. This allows all processors to have equal access to all disks in the system, providing the image of a scalable shared disk environment. The file system implements a UNIX file system on top of this hardware, such that file systems may be striped across up to 256 disks [13]. The result is a file system which can sustain transfer rates exceeding 150 MB/second in large configurations.

The basic approach taken in this implementation is to begin by partitioning the document set among the processors. This is done by having each processor read a fixed-size contiguous chunk (1 MB) of data from the input file. In general, this will result in some documents spanning processor boundaries, so document fragments will then be re-distributed. Once this is done, each processor parses and lexes its own documents, using conventional programming techniques. The postings are then inserted into the inverted file.

One novel implementation trick is used in this phase of processing: rather than sorting the postings, which would be very time consuming owing to the size of the uncompressed postings, the database is indexed in two passes. On the first pass, called the "dry run", the postings are generated, counted, and discarded. At the end of this pass, the system knows how much space is required to store the posting list for each word. Space is then allocated and carved up. The second "production" phase then begins. The database is scanned, lexed, and indexed again from scratch, but this time the postings are compressed and stored into the space allocated at the end of the dry run. This strategy doubles indexing time but, by eliminating the expense of sorting the posting file, it ends up both simplifying the software and reducing overall database construction time.

At the end of this phase, the data structures noted above (posting file, lexicon/index, and document information) have been constructed in-memory, and the database is ready for querying. Using these methods on a 64 processor CM-5, the TREC-2 training database (2.2 Gigabytes) required 20 minutes to index. The speed of the indexing software permits the database to be re-indexed for every retrieval experiment, allowing both indexing methods and query methods to be conveniently explored. Ignoring stop words, the size of the compressed inverted file index for the TREC database is 24% of the raw text. Details of the compression algorithm can be found in [11].

The first step in query evaluation is to broadcast the query to all processors. In a boolean system, the query would then be processed locally by each processor, then the results produced by the processors concatenated to return the final answer.

In an information retrieval system based on term weighting and document ranking, slightly more work is required. First, query-term weights are generally based on some sort of term-frequency observations. These cannot be done locally, but require the use of some simple global operations. For example, to determine the number of documents in which a word occurs, each processor would count document occurrences for itself, then the *global sum* operation (supported in hardware by the Control Network) would be used to produce a machine-wide count. The second problem which arises comes after the documents have been scored: an algorithm is required to extract the highest-ranking documents in the collection. Several parallel algorithms for this task have been described in [14]. The test-bed used a variant on the iterative extraction algorithm: each processor locally determines its highest-ranking documents, then repeated application of the *global maximum* operation are used to find the best documents in the collection.

III. THE COSINE SIMILARITY MEASURE

Results of using the classical cosine similarity measure on the TREC collection have already been reported elsewhere [15], so those results have not been replicated. This section will briefly describe how the cosine measure may be implemented within our architecture.

Cosine similarity measures generally involve constructing an inverted file incorporating document-term weights. The structured inverted file architecture does not provide this information, partly because it is inconsistent with the structured representation, and partly because of the difficulties noted above with regard to distributed databases. Except for

the cosine measure's document normalization factor, we find that it is possible to implement document-term weighting based only in information in the structured posting file.

In general, term weights (both in queries and documents) may be computed based on the following factors [5], all of which are available at run-time in our architecture.

- **Term Frequency.** This is the number of times a term occurs in the database. This information is retained in the lexicon.
- **Document-Term Frequency.** This is the number of documents in which a term occurs. It may be computed at run-time by traversing posting list for the term, counting initial occurrences of the term.
- **In-Document Frequency.** This is the number of times a term occurs in a given document. This may be computed at run-time by traversing the posting list for the term, counting the number of occurrences of the term having the same document coordinate.
- **Maximum In-Document Frequency.** This is the maximum number of times a given term occurs in any document. It may be computed directly from the in-document frequencies.

Most conventional weighting schemes (e.g. tf.idf) may be computed from these run-time factors. The advantage of doing these computations at run-time is that it eliminates the need to incorporate collection-specific information into the database at indexing time. This is important for dynamic collections as well as for distributed databases, as described above.

The difficulty with these methods when applied to the cosine norm is that the total-document-weight term cannot be conveniently computed at run-time using an inverted file. It must, therefore, be computed at index time and stored separately (e.g. in the document-specific information data structure).

There are two solutions to this difficulty. The first is to insist on using the cosine norm, accepting the difficulties that this implies. The second is to look for alternatives to the cosine norm. We believe that this is the more promising approach, but this is in the realm of work-not-yet-completed. In any event, the value of the cosine norm for large structured documents has not been established at this time.

IV. RETRIEVAL EXPERIMENTS:

Most of the time (about 3 to 4 person months) for our TREC participation was spent on building the test bed. However we did finish some runs with automatically constructed routing and adhoc queries for which we report the results here. The experiments were done using the entire collection. We compare words and phrases, mixed case vs. lower case and also explore document length normalization and proximity measures.

A. Retrieval for Routing Topics:

All queries were constructed and optimized automatically. The terms consisted of words (ignoring stop words) as well as all phrases consisting of adjacent words. Numbers were ignored and case was preserved. No stemming or query expansion with thesauri etc. was attempted.

Routing queries were constructed by looking at each word and (adjacent) phrase from the whole text of the topic templates and determining a weight based on the number of relevant documents present in the first 100 retrieved documents, by using just that term. An initial weighted query was constructed for each topic by the above process. Then each topic query was optimized by choosing thresholds (per topic) for the weights and rejecting all weighted query terms below the threshold. The optimum threshold for each topic was chosen by straightforward incremental search. Table 1 shows the results for the routing experiments. Routing queries with both

Table 1: Routing Queries

Method	Precision at 100 docs	Average precision
tmc6- routing-words-phrases	.3396	.2553
tmc7-routing-words	.2920	.2045
tmc6-routing-words-phrases-**ip**	.3750	.2716
tmc6-routing-words-phrases-**doc-length-sent-prox**	.3782	.2792
tmc6-routing-words-phrases-**ip-sent-prox**	.3856	.3344

weighted words and phrases (queryid tmc6) did better than queries using just words (queryid tmc7). Using the same (official) queries, but adding sentence level proximity (**sent-prox**), document length scaling (**doc-length**) and inverse weights based on which paragraph the term appears in (**ip**), seem to improve results (see the next section for more details about the techniques).

B. Retrieval for Adhoc Topics

Adhoc queries were automatically constructed by using words and phrases from different sections of the topic templates and using tf.idf weights (as derived from the training collection). The "best" sections for the new topics were chosen by experimenting with the training topics. Queries derived from the description-concept sections were used for most of the experiments. A threshold for weights was used to select terms for the final queries. Table 2 shows the results for the adhoc queries.

Table 2: Adhoc Queries

Method	Precision at 100 docs	Average precision
Case:		
tmc8-adhoc-dcwp-idf-**caps-wt**	.2002	.1276
tmc8-adhoc-dcwp-idf-**lower-wt**	.1734	.1157
Document-length and inverse-para-scaling:		
tmc8-adhoc-dcwp-idf-**lower-doc-length-wt**	.3308	.1904
tmc8-adhoc-dcwp-idf-**caps-ip-wt**	.3432	.1939
tmc8-adhoc-dcwp-idf-**lower-ip-wt**	.3422	.2027
tmc9-adhoc-etwp-idf-**caps-ip-wt**	.3144	.1736
Stemming:		
tmc8-adhoc-dcwp-idf-**lower-stem-wt**	.1670	.1152
tmc8-adhoc-dcwp-idf-**lower-stem-ip-wt**	.3240	.1980
Proximity:		
tmc8-adhoc-dcwp-idf-**caps-doc-length-sent-prox-wt**	.3436	.2012
tmc8-adhoc-dcwp-idf-**lower-doc-length-sent-prox-wt**	.3518	.2146
tmc8-adhoc-dcwp-idf-**caps-ip-para-prox-wt**	.2892	.1681
tmc8-adhoc-dcwp-idf-**lower-ip-para-prox-wt**	.3006	.1772
tmc8-adhoc-dcwp--idf-**caps-ip-sent-prox-wt**	.3476	.1988
tmc8-adhoc-dcwp-idf-**lower-ip-sent-prox-wt**	.3602	.2164

The query tmc8 (dcwp) consisted of words and phrases from the description and concept sections of the topic templates. Query tmc9 (etwp) used words and adjacent phrases from the entire topic. Bold-face acronyms emphasize particular experiments with case (**caps** and **lower**), sentence and paragraph level proximity (**sent-prox** and **para-prox**) document length scaling (**doc-length**), inverse weights based on paragraph position (**ip**) and weight thresholds (**wt**). The queries were not changed for the different experiments.

Idf weighted terms from the description and concept sections taken together (query tmc8) seem to do better than those derived from the entire topic (query tmc9).

C. Case

For the adhoc queries, we compared indexing with and without preserving case (similar treatment for the queries). Except for the simplest experiment with weight thresholds, converting everything to lower case seems to yield comparable or better results than upper case. A similar experiment for routing queries wasn't attempted because that would have required reformulating and reoptimizing the routing queries.

D. Stemming

Using the Porter Algorithm software for stemming from [16] we experimented with stemming at index time (and stemming the queries). We found that stemming reduces performance when compared with similar experiments using lower case -- since the software we had used lower case. We are not sure yet why there is such a decrease.

E. Document length and Term position

Document length scaling was used to explore the effect of emphasizing shorter documents. A linear decreasing scaling for longer documents, with a tail was used. An inverse weight based on the paragraph the term appears in, was also explored. Both the document length scaling and the inverse paragraph scaling increase performance significantly and seem to be comparable to each other.

F. Proximity

The postings for the inverted file allow use of term position. Experiments are underway to define proximity scoring methods that enhance weights for terms appearing close together (clusters of terms), and can also be implemented efficiently within the current architecture. We have achieved good results with sentence level proximity measures based on a bonus score for the query terms that appear within the same sentence and within a certain distance of each other. The bonus is also proportional to the term weight itself. Experiments that used a bonus independent of term weight dramatically reduced performance (numbers not reported here), possibly due to noise introduced by clusters of unimportant terms. Similar experiments with paragraph level proximity yielded significantly poorer results as compared to sentence level proximity. Finally combining either document length or inverse paragraph scaling with sentence level proximity improved results.

V. CONCLUSIONS

We have successfully implemented an in-memory compressed inverted file text retrieval system on the CM5 Connection Machine system.

Queries that used both words and phrases composed of adjacent words did better that those that used words alone. While our experiments completed so far suggest that converting everything to lower case for adhoc queries seems somewhat better, it is not clear whether the minor differences couldn't be removed by further optimization of other parameters. Experiments that take document length and term position into account suggest that normalizing for document length and increasing the weight of terms appearing earlier in the documents lead to significant improvements for both routing and adhoc queries. We have also seen that proximity scores based on nearby terms in a sentence improve retrieval performance.

VI. FUTURE WORK

We would like to compare the effects of cosine normalization with what we have tried so far and also explore its interaction with techniques that use term position and proximity measures.

VII. REFERENCES

[1] Cleverdon, C. W., Mills, J. & Keen, E. M. (1966). *Factors Determining the Performance of Indexing Systems, Vol 1: Design, Vol. 2: Test Results.* Aslib Cranfield Research Project, Cranfield, England, 1966.

[2] Sparck Jones, K. and Webster, C.A. (1979). *Research in Relevance Weighting*, British Library Research and Development Report 5553, Computer Laboratory, University of Cambridge.

[3] Fox, E. (1983). Characteristics of Two New Experimental Collections in Computer and Information Science Containing Textual and Bibliographic Concepts. *Technical Report TR 83-561*, Cornell Univerisy: Computing Science Department.

[4] Salton, G. (1989). *Automatic Text Processing*. New York: Addison-Wesley.

[5] Salton, G & Buckley, C. (1988). Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, **24** (5), pp. 513-523.

[6] Harman, D (1993). The DARPA TIPSTER Project. *SIGIR Forum*, **26** (2), pp. 26-28.

[7] Harman, D (1993). Overview of the First TREC Conference. *Proceedings, SIGIR-93*, pp. 36-47. Pittsburgh.

[8] Hollaar, L. A. (1992). Special-Purpose Hardware for Information Retrieval. In *Information Retrieval: Data Structures and Algorithms.* Frakes, B. & Baeze-Yates, R. eds. Englewood Cliffs, New Jersey: Prentice Hall.

[9] Faloutsos, C. (1992). Signature Files. In *Information Retrieval: Data Structures and Algorithms.* Frakes, B. & Baeze-Yates, R. eds. Englewood Cliffs, New Jersey: Prentice Hall.

[10] Harman, D., Fox, E. Baeza-Yates, A. & Lee, W. (1992). Inverted Files. In *Information Retrieval: Data Structures and Algorithms.* Frakes, B. & Baeze-Yates, R. eds. Englewood Cliffs, New Jersey: Prentice Hall.

[11] Linoff, G. & Stanfill, C (1993). Compression of Indexes with Full Positional Information in Very Large Text Databases. *Proceedings, SIGIR-93*, pp. 88-95. Pittsburgh.

[12] Thinking Machines Corporation (1991). Connection Machine CM-5 Technical Summary. Cambridge, MA: Thinking Machines Corporation.

[13] Lo Verso, S., Isman, M., Nanopoulos, A., Nesheim, W., Milne, E. & Wheeler, R. (1993). SFS: A Parallel File System for the CM-5. *Proceedings, 1993 Summer USENIX Technical Conference*. pp. 291-306. Cincinnati, OH.

[14] Stanfill, C. (1992). Parallel Information Retrieval Algorithms. In *Information Retrieval: Data Structures and Algorithms.* Frakes, B. & Baeze-Yates, R. eds. Englewood Cliffs, New Jersey: Prentice Hall.

[15] Salton, G. & Buckley, C. (1992). SMART Trade-offs. *Proceedings, TREC-1*. Gaithersburgh, MD.

[16] Frakes, B.(1992). Stemming Algorithms. In *Information Retrieval: Data Structures and Algorithms.* Frakes, B. & Baeze-Yates, R. eds. Englewood Cliffs, New Jersey: Prentice Hall.

RECENT DEVELOPMENTS IN NATURAL LANGUAGE TEXT RETRIEVAL

Tomek Strzalkowski and Jose Perez Carballo
Courant Institute of Mathematical Sciences
New York University
715 Broadway, rm. 704
New York, NY 10003
tomek@cs.nyu.edu

ABSTRACT

This paper reports on some recent developments in our natural language text retrieval system. The system uses advanced natural language processing techniques to enhance the effectiveness of term-based document retrieval. The backbone of our system is a traditional statistical engine which builds inverted index files from pre-processed documents, and then searches and ranks the documents in response to user queries. Natural language processing is used to (1) preprocess the documents in order to extract content-carrying terms, (2) discover inter-term dependencies and build a conceptual hierarchy specific to the database domain, and (3) process user's natural language requests into effective search queries. For the present TREC-2 effort, the total of 550 MBytes of Wall Street Journal articles (ad-hoc queries database) and 300 MBytes of San Jose Mercury articles (routing data) have been processed. In terms of text quantity this represents approximately 130 million words of English. Unlike in TREC-1, we were able to create a single compound index for each database, and therefore avoid merging of results. While the general design of the system has not changed since TREC-1 conference, we nonetheless replaced several components and added a number of new features which are described in the present paper.

INTRODUCTION

A typical information retrieval (IR) task is to select documents from a database in response to a user's query, and rank these documents according to relevance. This has been usually accomplished using statistical methods (often coupled with manual encoding) that (a) select terms (words, phrases, and other units) from documents that are deemed to best represent their content, and (b) create an inverted index file (or files) that provide an easy access to documents containing these terms. A subsequent search process will attempt to match a preprocessed user query (or queries) against term-based representations of documents in each case determining a degree of relevance between the two which depends upon the number and types of matching terms. Although many sophisticated search and matching methods are available, the crucial problem remains to be that of an adequate representation of content for both the documents and the queries.

The simplest word-based representations of content are usually inadequate since single words are rarely specific enough for accurate discrimination, and their grouping is often accidental. A better method is to identify groups of words that create meaningful *phrases*, especially if these phrases denote important concepts in database domain. For example, *joint venture* is an important term in the Wall Street Journal (WSJ henceforth) database, while neither *joint* nor *venture* is important by itself. In the retrieval experiments with the training TREC database, we noticed that both *joint* and *venture* were dropped from the list of terms by the system because their idf (*inverted document frequency*) weights were too low. In large databases, such as TIPSTER, the use of phrasal terms is not just desirable, it becomes necessary.

An accurate syntactic analysis is an essential prerequisite for selection of phrasal terms. Various statistical methods, e.g., based on word co-occurrences and mutual information, as well as partial parsing techniques, are prone to high error rates (sometimes as high as 50%), turning out many unwanted associations. Therefore a good, fast parser is necessary, but it is by no means sufficient. While syntactic phrases are often better indicators of content than 'statistical phrases' -- where words are grouped solely on the basis of physical proximity (e.g., "college junior" is not the same as "junior college") -- the creation of compound terms makes term matching process more complex since in addition to the usual problems of synonymy and subsumption, one must deal with their structure (e.g., "college junior" is the same as "junior in college"). In order to deal with structure, the parser's output needs to be "normalized" or "regularized" so that complex terms with the same or closely related meanings would indeed receive matching representations. This goal has been achieved to a certain extent in the present work. As it will be discussed in more detail below, indexing terms were selected from among head-modifier pairs extracted from predicate-argument

representations of sentences.

Introduction of compound terms also complicates the task of discovery of various semantic relationships among them, including synonymy and subsumption. For example, the term *natural language* can be considered, in certain domains at least, to subsume any term denoting a specific human language, such as *English*. Therefore, a query containing the former may be expected to retrieve documents containing the latter. The same can be said about *language* and *English*, unless *language* is in fact a part of the compound term *programming language* in which case the association *language - Fortran* is appropriate. This is a problem because (a) it is a standard practice to include both simple and compound terms in document representation, and (b) term associations have thus far been computed primarily at word level (including fixed phrases) and therefore care must be taken when such associations are used in term matching. This may prove particularly troublesome for systems that attempt term clustering in order to create "meta-terms" to be used in document representation.

The system presented here computes term associations from text at word and fixed phrase level and then uses these associations in query expansion. A fairly primitive filter is employed to separate synonymy and subsumption relationships from others including antonymy and complementation, some of which are strongly domain-dependent. This process has led to an increased retrieval precision in experiments with both ad-hoc and routing queries for TREC-1 and TREC-2 experiments. However, the actual improvement levels can vary substantially between different databases, types of runs (ad-hoc vs. routing), as well as the degree of prior processing of the queries. We continue to study more advanced clustering methods along with the changes in interpretation of resulting associations, as signaled in the previous paragraph.

In the remainder of this paper we discuss particulars of the present system and some of the observations made while processing TREC-2 data. The above comments will provide the background for situating our present effort and the state-of-the-art with respect to where we should be in the future.

OVERALL DESIGN

Our information retrieval system consists of a traditional statistical backbone (NIST's PRISE system; Harman and Candela, 1989) augmented with various natural language processing components that assist the system in database processing (stemming, indexing, word and phrase clustering, selectional restrictions), and translate a user's information request into an effective query. This design is a careful compromise between purely statistical non-linguistic approaches and those requiring rather accomplished (and expensive) semantic analysis of data, often referred to as 'conceptual retrieval'.

In our system the database text is first processed with a fast syntactic parser. Subsequently certain types of phrases are extracted from the parse trees and used as compound indexing terms in addition to single-word terms. The extracted phrases are statistically analyzed as syntactic contexts in order to discover a variety of similarity links between smaller subphrases and words occurring in them. A further filtering process maps these similarity links onto semantic relations (generalization, specialization, synonymy, etc.) after which they are used to transform a user's request into a search query.

The user's natural language request is also parsed, and all indexing terms occurring in it are identified. Certain highly ambiguous, usually single-word terms may be dropped, provided that they also occur as elements in some compound terms. For example, "natural" is deleted from a query already containing "natural language" because "natural" occurs in many unrelated contexts: "natural number", "natural logarithm", "natural approach", etc. At the same time, other terms may be added, namely those which are linked to some query term through admissible similarity relations. For example, "unlawful activity" is added to a query (TREC topic 055) containing the compound term "illegal activity" via a synonymy link between "illegal" and "unlawful".

One of the striking observations made during the course of TREC-2 was to note that removing low-quality terms from the queries is at least as important (and often more so) as adding synonyms and specializations. In some instances (e.g., routing runs) low-quality terms had to be removed (or inhibited) *before* similar terms could be added to the query or else the effect of query expansion was all but drowned out by the increased noise.[1]

After the final query is constructed, the database search follows, and a ranked list of documents is returned. It should be noted that all the processing steps, those performed by the backbone system, and those performed by the natural language processing components, are fully automated, and no human intervention or manual encoding is required.

FAST PARSING WITH TTP PARSER

TTP (Tagged Text Parser) is based on the Linguistic String Grammar developed by Sager (1981). The parser currently encompasses some 400 grammar productions, but it is by no means complete. The parser's output is a regularized parse tree representation of each

[1] We would like to thank Donna Harman for turning our attention to the importance of term weighting schemes, including term deletion.

sentence, that is, a representation that reflects the sentence's logical predicate-argument structure. For example, logical subject and logical object are identified in both passive and active sentences, and noun phrases are organized around their head elements. The parser is equipped with a powerful skip-and-fit recovery mechanism that allows it to operate effectively in the face of ill-formed input or under a severe time pressure. In the runs with approximately 130 million words of TREC's Wall Street Journal and San Jose Mercury texts,[2] the parser's speed averaged between 0.3 and 0.5 seconds per sentence, or up to 70 words per second, on a Sun's SparcStation2. In addition, TTP has been shown to produce parse structures which are no worse than those generated by full-scale linguistic parsers when compared to hand-coded Treebank parse trees.

TTP is a full grammar parser, and initially, it attempts to generate a complete analysis for each sentence. However, unlike an ordinary parser, it has a built-in timer which regulates the amount of time allowed for parsing any one sentence. If a parse is not returned before the allotted time elapses, the parser enters the skip-and-fit mode in which it will try to "fit" the parse. While in the skip-and-fit mode, the parser will attempt to forcibly reduce incomplete constituents, possibly skipping portions of input in order to restart processing at a next unattempted constituent. In other words, the parser will favor reduction to backtracking while in the skip-and-fit mode. The result of this strategy is an approximate parse, partially fitted using top-down predictions. The fragments skipped in the first pass are not thrown out, instead they are analyzed by a simple phrasal parser that looks for noun phrases and relative clauses and then attaches the recovered material to the main parse structure. Full details of TTP parser have been described in the TREC-1 report (Strzalkowski, 1993a), as well as in other works (Strzalkowski, 1992; Strzalkowski & Scheyen, 1993).

As may be expected, the skip-and-fit strategy will only be effective if the input skipping can be performed with a degree of determinism. This means that most of the lexical level ambiguity must be removed from the input text, prior to parsing. We achieve this using a stochastic parts of speech tagger to preprocess the text (see TREC-1 report for details). For TREC-2 a number of problems have been corrected in the tagger, including improper tokenization of input and handling of abbreviations.

[2] Approximately 0.85 GBytes of text, over 6 million sentences.

WORD SUFFIX TRIMMER

Word stemming has been an effective way of improving document recall since it reduces words to their common morphological root, thus allowing more successful matches. On the other hand, stemming tends to decrease retrieval precision, if care is not taken to prevent situations where otherwise unrelated words are reduced to the same stem. In our system we replaced a traditional morphological stemmer with a conservative dictionary-assisted suffix trimmer. [3] The suffix trimmer performs essentially two tasks: (1) it reduces inflected word forms to their root forms as specified in the dictionary, and (2) it converts nominalized verb forms (e.g., "implementation", "storage") to the root forms of corresponding verbs (i.e., "implement", "store"). This is accomplished by removing a standard suffix, e.g., "stor+age", replacing it with a standard root ending ("+e"), and checking the newly created word against the dictionary, i.e., we check whether the new root ("store") is indeed a legal word.

HEAD-MODIFIER STRUCTURES

Syntactic phrases extracted from TTP parse trees are head-modifier pairs. The head in such a pair is a central element of a phrase (main verb, main noun, etc.), while the modifier is one of the adjunct arguments of the head. In the TREC experiments reported here we extracted head-modifier word and fixed-phrase pairs only. While TREC databases are large enough to warrant generation of larger compounds, we were in no position to verify their effectiveness in indexing. This was largely because of the tight schedule, but also because of rapidly escalating complexity of the indexing process: even with 2-word phrases, compound terms accounted for nearly 88% of all index entries, in other words, including 2-word phrases increased the index size approximately 8 times.

Let us consider a specific example from the WSJ database:

> The former Soviet president has been a local hero ever since a Russian tank invaded Wisconsin.

The tagged sentence is given below, followed by the regularized parse structure generated by TTP, given in Figure 1.

> The/*dt* former/*jj* Soviet/*jj* president/*nn* has/*vbz* been/*vbn* a/*dt* local/*jj* hero/*nn* ever/*rb* since/*in* a/*dt* Russian/*jj* tank/*nn* invaded/*vbd* Wisconsin/*np* ./*per*

[3] Dealing with prefixes is a more complicated matter, since they may have quite strong effect upon the meaning of the resulting term, e.g., *un-* usually introduces explicit negation.

```
[assert
 [[perf [HAVE]]
  [[verb [BE]]
   [subject
    [np
     [n PRESIDENT]
     [t_pos THE]
     [adj [FORMER]]
     [adj [SOVIET]]]]
   [object
    [np
     [n HERO]
     [t_pos A]
     [adj [LOCAL]]]]
   [adv EVER]
   [sub_ord
    [SINCE
     [[verb [INVADE]]
      [subject
       [np
        [n TANK]
        [t_pos A]
        [adj [RUSSIAN]]]]
      [object
       [np
        [name [WISCONSIN]]]]]]]]]]
```

Figure 1. Predicate-argument parse structure.

It should be noted that the parser's output is a predicate-argument structure centered around main elements of various phrases. In Figure 1, BE is the main predicate (modified by HAVE) with 2 arguments (*subject, object*) and 2 adjuncts (*adv, sub_ord*). INVADE is the predicate in the subordinate clause with 2 arguments (*subject, object*). The subject of BE is a noun phrase with PRESIDENT as the head element, two modifiers (FORMER, SOVIET) and a determiner (THE). From this structure, we extract head-modifier pairs that become candidates for compound terms. The following types of pairs are considered: (1) a head noun and its left adjective or noun adjunct, (2) a head noun and the head of its right adjunct, (3) the main verb of a clause and the head of its object phrase, and (4) the head of the subject phrase and the main verb. These types of pairs account for most of the syntactic variants for relating two words (or simple phrases) into pairs carrying compatible semantic content. For example, the pair *retrieve+information* will be extracted from any of the following fragments: *information retrieval system; retrieval of information from databases*; and *information that can be retrieved by a user-controlled interactive search process*. In the example at hand, the following head-modifier pairs are extracted (pairs containing low-content elements, such as BE and FORMER, or names, such as WISCONSIN, will be later discarded):

PRESIDENT+BE, PRESIDENT+FORMER, PRESIDENT+SOVIET,
BE+HERO, HERO+LOCAL,
TANK+INVADE, TANK+RUSSIAN, INVADE+WISCONSIN

We may note that the three-word phrase *former Soviet president* has been broken into two pairs *former president* and *Soviet president*, both of which denote things that are potentially quite different from what the original phrase refers to, and this fact may have potentially negative effect on retrieval precision. This is one place where a longer phrase appears more appropriate. The representation of this sentence may therefore contain the following terms (along with their inverted document frequency weights):

PRESIDENT	2.623519
SOVIET	5.416102
PRESIDENT+SOVIET	11.556747
PRESIDENT+FORMER	14.594883
HERO	7.896426
HERO+LOCAL	14.314775
INVADE	8.435012
TANK	6.848128
TANK+INVADE	17.402237
TANK+RUSSIAN	16.030809
RUSSIAN	7.383342
WISCONSIN	7.785689

While generating compound terms we took care to identify 'negative' terms, that is, those whose denotations have been explicitly excluded by negation. Even though matching of negative terms was not used in retrieval (nor did we use negative weights), we could easily prevent matching a negative term in a query against its positive counterpart in the database by removing known negative terms from queries. As an example consider the following fragment from topic 067:

> It should NOT be about economically-motivated civil disturbances and NOT be about a civil disturbance directed against a second country.

The corresponding compound terms are:

NOT disturb+civil
NOT country+second
NOT direct+disturb

The particular way of interpreting syntactic contexts was dictated, to some degree at least, by statistical considerations. Our initial experiments were performed on a relatively small collection (CACM-3204), and therefore we combined pairs obtained from different syntactic relations (e.g., verb-object, subject-verb, noun-adjunct, etc.) in order to increase frequencies of some associations. This became largely unnecessary in a large collection such as TIPSTER, but we had no means to test alternative options, and thus decided to stay with the original. It should not be difficult to see that this was a compromise solution, since many important distinctions were

potentially lost, and strong associations could be produced where there weren't any. A way to improve things is to consider different syntactic relations independently, perhaps as independent sources of evidence that could lend support (or not) to certain term similarity predictions. We have started investigating this option during TREC-2, however, it has not been sufficiently tested yet.

One difficulty in obtaining head-modifier pairs of highest accuracy is the notorious ambiguity of nominal compounds. For example, the phrase *natural language processing* should generate *language+natural* and *processing+language*, while *dynamic information processing* is expected to yield *processing+dynamic* and *processing+information*. A still another case is *executive vice president* where the association *president+executive* may be stretching things a bit too far. Since our parser has no knowledge about the text domain, and uses no semantic preferences, it does not attempt to guess any internal associations within such phrases. Instead, this task is passed to the pair extractor module which processes ambiguous parse structures in two phases. In phase one, all and only unambiguous head-modifier pairs are extracted, and the frequencies of their occurrences are recorded. In phase two, frequency information about pairs generated in the first pass is used to form associations from ambiguous structures. For example, if *language+natural* has occurred unambiguously a number of times in contexts such as *parser for natural language*, while *processing+natural* has occurred significantly fewer times or perhaps none at all, then we will prefer the former association as valid. In TREC-2 phrase disambiguation was not used, instead we decided to avoid ambiguous phrases altogether. While our *disambig* program worked generally satisfactorily, we could resolve only a small fraction of cases (about 7%) and thus it's impact on the overall system's performance was limited. However, query-level disambiguation may be more important.

TERM CORRELATIONS FROM TEXT

Head-modifier pairs form compound terms used in database indexing. They also serve as occurrence contexts for smaller terms, including single-word terms. If two terms tend to be modified with a number of common modifiers and otherwise appear in few distinct contexts, we assign them a similarity coefficient, a real number between 0 and 1. The similarity is determined by comparing distribution characteristics for both terms within the corpus: how much information content do they carry, do their information contribution over contexts vary greatly, are the common contexts in which these terms occur specific enough? In general we will credit high-content terms appearing in identical contexts, especially if these contexts are not too commonplace.[4]

For TREC-2 runs we used a similarity formula which, unlike the similarity formula used in TREC-1, produces clusters of related words and phrases, but will not generate uniform term similarity ranking across clusters. This new formula, however, appeared better suited to handle the diverse subject matter of the WSJ database. We used a (revised) variant of weighted Tanimoto's measure described in (Grefenstette, 1992):

$$SIM(x_1,x_2) = \frac{\sum_{att} MIN(W([x,att]),W([y,att])}{\sum_{att} MAX(W([x,att]),W([y,att])}$$

with

$$W([x,y]) = GEW(x)*log(f_{x,y})$$

$$GEW(x) = 1 + \sum_{y} \left(\frac{\frac{f_{x,y}}{n_y} * \log\left(\frac{f_{x,y}}{n_y}\right)}{log(N)} \right)$$

In the above, $f_{x,y}$ stands for absolute frequency of pair $[x,y]$, n_y is the frequency of term y, and N is the number of single-word terms. Sample clusters obtained from approx. 250 MByte (42 million words) subset of WSJ (years 1990-1992) are given in Table 1.

In order to generate better similarities, we require that words x_1 and x_2 appear in at least M *distinct* common contexts, where a common context is a couple of pairs $[x_1,y]$ and $[x_2,y]$, or $[y,x_1]$ and $[y,x_2]$ such that they each occurred at least three times. Thus, *banana* and *Baltic* will not be considered for similarity relation on the basis of their occurrences in the common context of *republic*, no matter how frequent, unless there is another such common context comparably frequent (there wasn't any in TREC's WSJ database). For smaller or narrow domain databases M=2 is usually sufficient. For large databases covering a rather diverse subject matter, like WSJ or SJMN (San Jose Mercury News), we used $M \geq 5$.[5] This, however, turned out not to be sufficient. We would still generate fairly strong similarity links between terms such as *aerospace* and *pharmaceutical* where 6 and more common contexts were found. In the example at hand the following common contexts were located, all occurring at the head (left) position of a pair (at right are their global entropy weights and frequencies with *aerospace* and

[4] It would not be appropriate to predict similarity between *language* and *logarithm* on the basis of their co-occurrence with *natural*.

[5] For example *banana* and *Dominican* were found to have two common contexts: *republic* and *plant*, although this second occurred in apparently different senses in *Dominican plant* and *banana plant*.

pharmaceutical, respectively):[6]

firm	GEW=0.58	fx1y=9	fx2y=22
industry	GEW=0.51	fx1y=84	fx2y=56
sector	GEW=0.61	fx1y=5	fx2y=9
concern	GEW=0.50	fx1y=130	fx2y=115
analyst	GEW=0.62	fx1y=23	fx2y=8
division	GEW=0.53	fx1y=36	fx2y=28
giant	GEW=0.62	fx1y=15	fx2y=12

Note that while some of these weights are quite low (less than 0.6 -- GEW takes values between 0 and 1), thus indicating a low importance context, the frequencies with which these contexts occurred with both terms were high and balanced on both sides (e.g., *concern*), thus adding to the strength of association. We are now considering additional thresholds to bar low importance contexts from being used in similarity calculation.

It may be worth pointing out that the similarities are calculated using term co-occurrences in syntactic rather than in document-size contexts, the latter being the usual practice in non-linguistic clustering (e.g., Sparck Jones and Barber, 1971; Crouch, 1988; Lewis and Croft, 1990). Although the two methods of term clustering may be considered mutually complementary in certain situations, we believe that more and stronger associations can be obtained through syntactic-context clustering, given sufficient amount of data and a reasonably accurate syntactic parser.[7]

QUERY EXPANSION

Similarity relations are used to expand user queries with new terms, in an attempt to make the final search query more comprehensive (adding synonyms) and/or more pointed (adding specializations).[8] It follows that not all similarity relations will be equally useful in query expansion, for instance, complementary and antonymous relations like the one between *Australian* and *Canadian*, *accept* and *reject*, or even generalizations like from *aerospace* to *industry* may actually harm system's performance, since we may end up retrieving many irrelevant documents. On the other hand, database search is likely to miss relevant documents if we overlook the fact that *vice director* can also be *deputy director*, or that *takeover* can also be *merge*, *buy-out*, or *acquisition*. We noted that an average set of similarities generated from a text corpus contains about as many "good" relations (synonymy, specialization) as "bad" relations (antonymy, complementation, generalization), as seen from the query expansion viewpoint. Therefore any attempt to separate these two classes and to increase the proportion of "good" relations should result in improved retrieval. This has indeed been confirmed in our experiments where a relatively crude filter has visibly increased retrieval precision.

In order to create an appropriate filter, we devised a global term specificity measure (GTS) which is calculated for each term across all contexts in which it occurs. The general philosophy here is that a more specific word/phrase would have a more limited use, i.e., a more specific term would appear in fewer *distinct* contexts. In this respect, GTS is similar to the standard *inverted document frequency (idf)* measure except that term frequency is measured over syntactic units rather than document size units.[9] Terms with higher GTS values are generally considered more specific, but the specificity comparison is only meaningful for terms which are already known to be similar. The new function is calculated according to the following formula:

$$GTS(w) = \begin{cases} IC_L(w) * IC_R(w) & \text{if } \textit{both exist} \\ IC_R(w) & \text{if } \textit{only } IC_R(w) \textit{ exists} \\ IC_L(w) & \textit{otherwise} \end{cases}$$

where (with n_w, $d_w > 0$):

$$IC_L(w) = IC([w,\_]) = \frac{n_w}{d_w(n_w+d_w-1)}$$

$$IC_R(w) = IC([\_,w]) = \frac{n_w}{d_w(n_w+d_w-1)}$$

For any two terms w_1 and w_2, and a constant $\delta > 1$, if $GTS(w_2) \geq \delta * GTS(w_1)$ then w_2 is considered more specific than w_1. In addition, if $SIM_{norm}(w_1,w_2) = \sigma > \theta$, where θ is an empirically established threshold, then w_2 can be added to the query containing term w_1 with weight σ.[10] For example, the following were obtained

[6] Other common contexts, such as *company* or *market*, have already been rejected because they were paired with too many different words (a high dispersion ratio, see note 12).

[7] Non-syntactic contexts cross sentence boundaries with no fuss, which is helpful with short, succinct documents (such as CACM abstracts), but less so with longer texts; see also (Grishman et al., 1986).

[8] Query expansion (in the sense considered here, though not quite in the same way) has been used in information retrieval research before (e.g., Sparck Jones and Tait, 1984; Harman, 1988), usually with mixed results. An alternative is to use term clusters to create new terms, "meta-terms", and use them to index the database instead (e.g., Crouch, 1988; Lewis and Croft, 1990). We found that the query expansion approach gives the system more flexibility, for instance, by making room for hypertext-style topic exploration via user feedback.

[9] We believe that measuring term specificity over document-size contexts (e.g., Sparck Jones, 1972) may not be appropriate in this case. In particular, syntax-based contexts allow for processing texts without any internal document structure.

[10] For TREC-2 we used $\sigma = 0.2$; δ varied between 10 and 100.

from the WSJ training database:

$GTS(takeover) = 0.00145576$
$GTS(merge) = 0.00094518$
$GTS(buy\text{-}out) = 0.00272580$
$GTS(acquire) = 0.00057906$

with

$SIM(takeover, merge) = 0.190444$
$SIM(takeover, buy\text{-}out) = 0.157410$
$SIM(takeover, acquire) = 0.139497$
$SIM(merge, buy\text{-}out) = 0.133800$
$SIM(merge, acquire) = 0.263772$
$SIM(buy\text{-}out, acquire) = 0.109106$

Therefore both *takeover* and *buy-out* can be used to specialize *merge* or *acquire*. With this filter, the relationships between *takeover* and *buy-out* and between *merge* and *acquire* are either both discarded or accepted as synonymous. At this time we are unable to tell synonymous or near synonymous relationships from those which are primarily complementary, e.g., *man* and *woman*.

In TREC-1 the impact of query expansion through term similarities on the system's overall performance was generally disappointing. For TREC-2 we have made a number of changes to the term correlation model, but again time limitations prevented us from properly testing all options. Among the most important changes are:

(1) Exclusion of pairs obtained from SUBJECT--VERB relations: we determined that these contexts are generally of little use as neither subject nor verb subcategorizes well for the other. Moreover we observed that the presence of these pairs was the source of many unwanted term associations.[11]

(2) Automatic pruning of low-content terms from the queries: terms with low idf weights, terms with low information contribution weights that are elements of compound terms, are removed from queries before database search. As we tuned various cutoff thresholds we noted that a significant increase in both recall and precision could be obtained.[12]

word	cluster
takeover	*merge, buy-out, acquire, bid*
benefit	*compensate, aid, expense*
capital	*cash, fund, money*
staff	*personnel, employee, force*
attract	*lure, draw, woo*
sensitive	*crucial, difficult, critical*
speculate	*rumor, uncertainty, tension*
president	*director, executive, chairman*
vice	*deputy*
outlook	*forecast, prospect, trend*
law	*rule, policy, legislate, bill*
earnings	*profit, revenue, income*
portfolio	*asset, invest, loan*
inflate	*growth, demand, earnings*
industry	*business, company, market*
growth	*increase, rise, gain*
firm	*bank, concern, group, unit*
environ	*climate, condition, situation*
debt	*loan, secure, bond*
lawyer	*attorney*
counsel	*attorney, administrator, secretary*
compute	*machine, software, equipment*
competitor	*rival, competition, buyer*
alliance	*partnership, venture, consortium*
big	*large, major, huge, significant*
fight	*battle, attack, war, challenge*
base	*facile, source, reserve, support*
shareholder	*creditor, customer, client investor, stockholder*

Table 1. Selected clusters obtained from syntactic contexts, derived from approx. 40 million words of WSJ text, with weighted Tanimoto formula.

[11] Subject-Verb pairs were retained as compound terms, however.

[12] The Information Contribution measure indicates the strength of word pairings, and is defined as $IC(x,[x,y]) = \frac{f_{x,y}}{n_x + d_x - 1}$ where $f_{x,y}$ is the absolute frequency of pair [x,y] in the corpus, n_x is the frequency of term x at the head position, and d_x is a dispersion parameter understood as the number of distinct syntactic contexts in which term x is found.

CREATING AN INDEX

Most of the problems we encountered when adapting NIST's PRISE system for use in TREC-2 had to do with the size of the data that had to be indexed.

We had to deal with the restrictions imposed by the resources we had (e.g., only 96 MBytes of virtual memory). The rest of this paragraph signals some of the changes we made to the NIST system in order to deal with our restrictions. The original system would request twice the previously requested amount of memory each time it needed more. As a result of this the system would reach the limit of virtual memory after only a relatively small portion of the total number of documents had been indexed. In our version, the memory requested by the system grows linearly. The increments are estimated in such a way that the system never requests too much memory.

The indexing process became too fragile when the limits of the environment were approached. When a large portion of the virtual memory and of the disk space was being used by the indexing process, crashes became very likely. Unfortunately, it turned out that the process was very difficult to restart after some crashes (e.g., in the *rebuild* phase), thus leading to time consuming repeats.

Indexing also takes too long at present. Given the size of the data to be indexed the whole process takes at least 250 hours if everything goes well, which happens seldom. Given TREC-2's deadlines we could not afford to perform too many experiments: we barely had time to index the corpus once.

Most of the previous problems could be solved by distributing the indexing process to several different machines and performing the indexing in parallel.

We believe that it is possible to create several small indexes instead of a single very large one. If certain rules are followed when creating the distributed index, it should be possible to merge the results of querying the set of small indexes and to obtain a performance (recall and precision) comparable to the results obtained using a single index. The test setup we built in order to perform the experiments required for TREC-2 should allow us to test these hypotheses. The advantages of a distributed index are clear:

(1) The indexing process would be faster.

(2) Each one of the distributed indexing processes would be smaller and less fragile.

(3) Even if one of the distributed processes crashes restarting it would be less expensive.

(4) A distributed system would be much easier to update, i.e., adding a new document would not require to reindex the whole corpus.

(5) A distributed system would be more likely to be useful in order to study the kinds of problems and solutions that are likely to be encountered in a real world situation.

SUMMARY OF RESULTS

We have processed the total of 850 MBytes of text during TREC-2. The first 550 MBytes were articles from the Wall Street Journal which were previously processed for TREC-1; we had to repeat most of the processing to correct early tokenization errors introduced by the tagger. The entire process (tagging, parsing, phrase extraction) took just over 4 weeks on 2 Sun's SparcStations (1 and 2). Building a comprehensive index for the WSJ database took up another 2 weeks. This time we were able to create a single index thanks to the improved indexing software received from NIST. The final index size was 204 MBytes, and included 2,274,775 unique terms (of which about 310,000 were single word terms, and the remaining 1,865,000 were syntactic word pairs) occurring in 173,219 documents, or more than 13 (unique) terms per document. Note that this gives potentially much better coverage of document content than single word terms alone with less than 2 unique terms per document. We say 'potentially' since the process of deriving phrase-level terms from text is still only partially understood, including the complex problem of 'normalization' of representation.

The remaining 300 MBytes were articles from the San Jose Mercury News, which were contained in TIPSTER disk-3. Processing of this part, and creating an index for routing purposes took about 3 weeks. While natural language processing required 2 weeks to complete (at approximately the same speed as WSJ database), we were able to cut indexing time in half by using a faster in-memory version of the NIST system. This new version reduces the time required by the first phase of indexing from days to hours, however the second phase remains slow (days) and fragile (we had to redo it 3 times). The final size of the SJMN index was 101 MBytes, with 1,535,971 unique terms occurring in 86,270 documents (nearly 18 unique terms per document).[13]

Two types of retrieval have been done: (1) new topics 101-150 were run in the ad-hoc mode against WSJ database, and (2) topics 51-100, previously used in TREC-1, were run in the routing mode against SJMN database. In each category several runs were attempted

[13] It has to be noted that the ratios at which new terms are generated are nearly identical in both databases; at 86,319 documents (or about half way through WSJ database) 1,335,622 unique terms had been recorded.

with a different combination of fields from the topics used to create search queries. A typical topic is shown below:

<top>
<head> Tipster Topic Description
<num> Number: 107
<dom> Domain: International Economics
<title> Topic: Japanese Regulation of Insider Trading
<desc> Description:
Document will inform on Japan's regulation of insider trading.
<narr> Narrative:
A relevant document will provide data on Japanese laws, regulations, and/or practices which help the foreigner understand how Japan controls, or does not control, stock market practices which could be labeled as insider trading.
<con> Concept(s):
1. insider trading
2. Japan
3. Ministry of Finance, Securities and Exchange Council, Osaka Securities Exchange, Tokyo Stock Exchange
4. Securities and Exchange Law, Article 58, law, legislation, guidelines, self-regulation
5. Nikko Securities, Yamaichi Securities, Nomura Securities, Daiwa Securities, Big Four brokerage firms
<fac> Factor(s):
<nat> Nationality: Japan
</fac>
</top>

This topic actually consists of two different statements of the same query: the natural language specification consisting of <desc> and <narr> fields, and an expert-selected list of key terms which are often far more informative than the narrative part (in some cases these terms were selected via feedback in actual retrieval attempts). The table below shows the search query obtained from fields <desc> and <narr> of topic 107, after an expansion with similar terms and deleting low-content terms.

Query 107

standard+trade	idf=16.81	weight=0.38
standard+trade	idf=16.81	weight=0.38
regulate+japanese	idf=15.40	weight=1.00
standard+japanese	idf=14.08	weight=0.38
regulate+trade	idf=12.84	weight=1.00
regulate+trade	idf=12.84	weight=1.00
controls	idf=9.97	weight=1.00
labele	idf=9.20	weight=1.00
trade+inside	idf=8.62	weight=1.00
trade+inside	idf=8.62	weight=1.00
inside	idf=6.49	weight=1.00
inside	idf=6.49	weight=1.00
inside	idf=6.49	weight=1.00
regulate	idf=5.66	weight=1.00
regulate	idf=5.66	weight=1.00
regulate	idf=5.66	weight=1.00
practice	idf=5.46	weight=1.00
practice	idf=5.46	weight=1.00
data	idf=4.91	weight=1.00
data	idf=4.91	weight=0.51
data	idf=4.91	weight=0.26
japanese	idf=4.84	weight=1.00
japanese	idf=4.84	weight=1.00
standard	idf=4.81	weight=0.38
standard	idf=4.81	weight=0.38
standard	idf=4.81	weight=0.38
inform	idf=4.71	weight=1.00
inform	idf=4.71	weight=0.26
inform	idf=4.71	weight=0.51
protect	idf=4.69	weight=0.41

Note that many 'function' words have been removed from the query, e.g., *provide*, *understand*, as well as other 'common words' such as *document* and *relevant* (this is in addition to our regular list of 'stopwords'). Some still remain, however, e.g., *data* and *inform*, because these could not be uniformly considered as 'common' across all queries.

Results obtained for queries using text fields only and those based primarily on keyword fields are reported separately. The purpose of this distinction was to demonstrate that (or whether) an intensive natural language processing can make an imprecise and frequently convoluted narrative into a better query that an expert would create.[14]

The ad-hoc category runs were done as follows (these are the official TREC-2 results):

(1) *nyuir1*: An automatic run of topics 101-150 against the WSJ database with the following fields used: <title>, <desc>, and <narr> only. Both syntactic phrases and term similarities were included.

(2) *nyuir2*: An automatic run of topics 101-150 against the WSJ database with the following fields used: <title>, <desc>, <con> and <fac> only. Both syntactic phrases and term similarities were included.

[14] Some results on the impact of different fields in TREC topics on the final recall/precision results were reported by Broglio and Croft (1993) at the ARPA HLT workshop, although text-only runs were not included. One of the most striking observations they have made is that the narrative field is entirely disposable, and moreover that its inclusion in the query actually hurts the system's performance. Croft (personal communication, 1992) has suggested that excluding all expert-made fields (i.e., <con> and <fac>) would make the queries quite ineffective. Broglio (personal communication, 1993) confirms this showing that text-only retrieval (i.e., with <desc> and <narr>) shows an average precision at more than 30% below that of <con>-based retrieval.

(3) *nyuir3*: A run of manually pruned topics 101-150 against the WSJ database with the following fields used: <title>, <desc>, <con> and <fac> only. Both syntactic phrases and term similarities were included. Manual intervention involved removing some terms from queries before database search.

Summary statistics for these runs are shown in Table 2. In addition, the 'base' column reports the system's performance on text fields with no language preprocessing, and no phrase terms or similarities used. We must note, however, that in all cases the topics have been processed with our suffix-trimmer, which means some NLP has been done already (tagging + lexicon), and therefore what we do not see here a performance of 'pure' statistical system.

In the routing category only automatic runs were done (again, these are the official TREC-2 results):

(1) *nyuir1*: An automatic run of topics 51-100 against the SJMN database with the following fields used: <title>, <desc>, and <narr> only. Both syntactic phrases and term similarities were included.

(2) *nyuir2*: An automatic run of topics 51-100 against the SJMN database with the following fields used: <title>, <desc>, <con> and <fac> only. Both syntactic phrases and term similarities were included.

A (simulated) routing mode run means that queries (i.e., terms and their weights) were derived with respect to a (different) training database (here WSJ), and were subsequently run against the new database (here SJMN). In particular, this means that the terms and their relative importance (reflected primarily through idf weights) were those of WSJ database rather than SJMN database.

Routing runs are summarized in Table 3. Again a column 'base' is added to show the system's performance without NLP module. We may note that the routing results are generally well below the ad-hoc results, both because the base system performance is inferior and because query processing has a different effect on the final statistics. The last column is a post-TREC run.[15]

Run Name	base ad-hoc	nyuir1 ad-hoc	nyuir2 ad-hoc	nyuir3 ad-hoc
Queries	50	50	50	50
Tot number of docs over all queries				
Ret	49887	49884	49876	49877
Rel	3929	3929	3929	3929
RelRet	2740	2983	3274	3281
Recall	(interp) Precision Averages			
0.00	0.7038	0.7013	0.7528	0.7528
0.10	0.4531	0.4874	0.5567	0.5574
0.20	0.3708	0.4326	0.4721	0.4724
0.30	0.3028	0.3531	0.4060	0.4076
0.40	0.2550	0.3076	0.3617	0.3621
0.50	0.2059	0.2637	0.3135	0.3142
0.60	0.1641	0.2175	0.2703	0.2711
0.70	0.1180	0.1617	0.2231	0.2237
0.80	0.0766	0.1176	0.1667	0.1697
0.90	0.0417	0.0684	0.0915	0.0916
1.00	0.0085	0.0102	0.0154	0.0160
Average precision over all rel docs				
Avg	0.2224	0.2649	0.3111	0.3118
Precision at				
5 docs	0.4640	0.4920	0.5360	0.5360
10 docs	0.4140	0.4420	0.4880	0.4880
15 docs	0.3867	0.4240	0.4693	0.4707
20 docs	0.3670	0.4050	0.4390	0.4410
30 docs	0.3253	0.3640	0.4067	0.4080
100 docs	0.2304	0.2720	0.3094	0.3094
200 docs	0.1626	0.1886	0.2139	0.2140
500 docs	0.0911	0.1026	0.1137	0.1140
1000 docs	0.0548	0.0597	0.0655	0.0656
R-Precision (after RelRet)				
Exact	0.2605	0.3003	0.3320	0.3321

Table 2. Automatic ad-hoc run statistics for queries 101-150 against WSJ database: (1) *base* - statistical terms only with <desc> and <narr> fields; (2) *nyuir1* - using syntactic phrases and similarities with <desc> and <narr> fields only; (3) *nyuir2* - same as 2 but with <desc>, <con>, and <fac> fields only; and (4) *nyuir3* - same as 3 but queries manually pruned before search.

[15] It should be noted that in category B runs, three topics (63, 65, and 88) had no relevant documents in SJMN database. Unfortunately, the evaluation program counts those as if there were relevant documents but none had been found, thus underestimating the system's performance by 5 to 8%. Excluding these three topics from consideration we obtain, in the last column, the average precision of 0.2624 and the R-precision of 0.3000.

Run Name	base routing	nyuir1 routing	nyuir2 routing	nyuir2a routing
Queries	50	50	50	50
Tot number of docs over all queries				
Ret	50000	50000	50000	50000
Rel	2064	2064	2064	2064
RelRet	1349	1390	1610	1623
Recall	(interp) Precision Averages			
0.00	0.5276	0.5400	0.6435	0.6458
0.10	0.3685	0.3937	0.4610	0.5021
0.20	0.3054	0.3423	0.3705	0.4151
0.30	0.2373	0.2572	0.3031	0.3185
0.40	0.2039	0.2263	0.2637	0.2720
0.50	0.1824	0.2032	0.2282	0.2379
0.60	0.1596	0.1674	0.1934	0.1899
0.70	0.1167	0.1295	0.1542	0.1571
0.80	0.0854	0.0905	0.1002	0.1163
0.90	0.0368	0.0442	0.0456	0.0434
1.00	0.0228	0.0284	0.0186	0.0158
Average precision over all rel docs				
Avg	0.1884	0.2038	0.2337	0.2466
Precision at				
5 docs	0.3160	0.3360	0.4280	0.4440
10 docs	0.3100	0.3240	0.4000	0.4180
15 docs	0.2813	0.2933	0.3613	0.3800
20 docs	0.2670	0.2790	0.3260	0.3530
30 docs	0.2240	0.2404	0.2760	0.2993
100 docs	0.1306	0.1412	0.1708	0.1698
200 docs	0.0865	0.0939	0.1078	0.1107
500 docs	0.0464	0.0489	0.0575	0.0570
1000 docs	0.0270	0.0278	0.0322	0.0325
R-Precision (after Rel)				
Exact	0.2196	0.2267	0.2513	0.2820

Table 3. Automatic routing run statistics for queries 51-100 against SJMN database: (1) *base* - statistical terms only with <desc> and <narr> fields; (2) *nyuir1* - using syntactic phrases and similarities with <desc> and <narr> fields only; (3) *nyuir2* - same as 2 but with <desc>, <con>, and <fac> fields only; and (4) *nyuir2a* - run *nyuir2* repeated with new weighting for phrases.

TERM WEIGHTING ISSUES

Finding a proper term weighting scheme is critical in term-based retrieval since the rank of a document is determined by the weights of the terms it shares with the query. One popular term weighting scheme, known as tf.idf, weights terms proportionately to their inverted document frequency scores and to their in-document frequencies (tf). The in-document frequency factor is usually normalized by the document length, that is, it is more significant for a term to occur 5 times in a short 20-word document, than to occur 10 times in a 1000-word article.[16]

In our official TREC runs we used the normalized tf.idf weights for all terms alike: single 'ordinary-word' terms, proper names, as well as phrasal terms consisting of 2 or more words. Whenever phrases were included in the term set of a document, the length of this document was increased accordingly. This had the effect of decreasing tf factors for 'regular' single word terms.

A standard tf.idf weighting scheme (and we suspect any other uniform scheme based on frequencies) is inappropriate for mixed term sets (ordinary concepts, proper names, phrases) because:

(1) It favors terms that occur fairly frequently in a document, which supports only general-type queries (e.g., "all you know about 'star wars'"). Such queries are not typical in TREC.

(2) It attaches low weights to infrequent, highly specific terms, such as names and phrases, whose only occurrences in a document often decide of relevance. Note that such terms cannot be reliably distinguished using their distribution in the database as the sole factor, and therefore syntactic and lexical information is required.

(3) It does not address the problem of inter-term dependencies arising when phrasal terms and their component single-word terms are all included in a document representation, i.e., *launch+satellite* and *satellite* are not independent, and it is unclear whether they should be counted as two terms.

In our post-TREC-2 experiments we considered (1) and (2) only. We changed the weighting scheme so that the phrases (but not the names which we did not distinguish in TREC-2) were more heavily weighted by their idf scores while the in-document frequency scores were replaced by logarithms multiplied by sufficiently large constants. In addition, the top N highest-idf matching terms (simple or compound) were counted more toward the document score than the remaining terms. This 'hot-spot' retrieval option is discussed in the next section.

[16] This is not always true, for example when all occurrences of a term are concentrated in a single section or a paragraph rather than spread around the article. See the following section for more discussion.

The table below illustrates the problem of weighting phrasal terms Using topic 101 and a relevant document (WSJ870226-0091).

Topic 101 matches WSJ870226-0091
duplicate terms not shown

TERM	*TF.IDF*	*NEW WEIGHT*
sdi	1750	1750
eris	3175	3175
star	1072	1072
wars	1670	1670
laser	1456	1456
weapon	1639	1639
missile	872	872
space+base	2641	2105
interceptor	2075	2075
exoatmospheric	1879	3480
system+defense	2846	2219
reentry+vehicle	1879	3480
initiative+defense	1646	2032
system+interceptor	2526	3118
DOC RANK	30	10

Changing the weighting scheme for compound terms, along with other minor improvements (such as expanding the stopword list for topics, or correcting a few parsing bugs) has lead to the overall increase of precision of nearly 20% over our official TREC-2 ad-hoc results. Table 4 summarizes these new runs for queries 101-150 against WSJ database. Similar improvements have been obtained for queries 51-100.

The results of the routing runs against SJMN database are somewhat more troubling. Applying the new weighting scheme we did see the average precision increase by some 5 to 12% (see column 4 in Table 3), but the results remain far below those for the ad-hoc runs. Direct runs of queries 51-100 against SJMN database produce results that are about the same as in the routing runs (which may indicate that our routing scheme works fine), however the same queries run against WSJ database have retrieval precision some 25% above SJMN runs. This may indicate some problems with SJMN database or the relevance judgements for it.

HOT SPOT' RETRIEVAL

Another difficulty with frequency-based term weighting arises when a long document needs to be retrieved on the basis of a few short relevant passages. If the bulk of the document is not directly relevant to the query, then there is a strong possibility that the document will score low in the final ranking, despite some strongly relevant material in it. This problem can be dealt with by subdividing long documents at paragraph breaks, or into approximately equal length fragments and indexing the database with respect to these (e.g., Kwok 1993). While

Run	nyuir1	nyuir1a	nyuir2	nyuir2a
Name	ad-hoc	ad-hoc	ad-hoc	ad-hoc
Queries	50	50	50	50
Tot number of docs over all queries				
Ret	49884	50000	49876	50000
Rel	3929	3929	3929	3929
RelRet	2983	3108	3274	3401
Recall	(interp) Precision Averages			
0.00	0.7013	0.7201	0.7528	0.8063
0.10	0.4874	0.5239	0.5567	0.6198
0.20	0.4326	0.4751	0.4721	0.5566
0.30	0.3531	0.4122	0.4060	0.4786
0.40	0.3076	0.3541	0.3617	0.4257
0.50	0.2637	0.3126	0.3135	0.3828
0.60	0.2175	0.2752	0.2703	0.3380
0.70	0.1617	0.2142	0.2231	0.2817
0.80	0.1176	0.1605	0.1667	0.2164
0.90	0.0684	0.1014	0.0915	0.1471
1.00	0.0102	0.0194	0.0154	0.0474
Average precision over all rel docs				
Avg	0.2649	0.3070	0.3111	0.3759
Precision at				
5 docs	0.4920	0.5200	0.5360	0.6040
10 docs	0.4420	0.4900	0.4880	0.5580
15 docs	0.4240	0.4653	0.4693	0.5253
20 docs	0.4050	0.4420	0.4390	0.4980
30 docs	0.3640	0.3993	0.4067	0.4607
100 docs	0.2720	0.2914	0.3094	0.3346
200 docs	0.1886	0.2064	0.2139	0.2325
500 docs	0.1026	0.1103	0.1137	0.1229
1000 docs	0.0597	0.0622	0.0655	0.0680
R-Precision (after Rel)				
Exact	0.3003	0.3332	0.3320	0.3950

Table 4. Automatic ad-hoc run statistics for queries 101-150 against WSJ database: (1) *nyuir1* - TREC-2 official run with <desc> and <narr> fields only; (2) *nyuir1a* - revised term weighting run; (3) *nyuir2* - official TREC-2 run with <desc>, <con>, and <fac> fields only; and (4) *nyuir2a* - revised weighting run.

such approaches are effective, they also tend to be costly because of increased index size and more complicated

access methods.

Efficiency considerations has led us to investigate an alternative approach to the *hot spot* retrieval which would not require re-indexing of the existing database or any changes in document access. In our approach, the maximum number of terms on which a query is permitted to match a document is limited to N highest weight terms, where N can be the same for all queries of may vary from one query to another. Note that this is not the same as simply taking the N top terms from each query. Rather, for each document for which there are M matching terms with the query, only min(M,N) of them, namely those which have highest weights, will be considered when computing the document score. Moreover, only the global importance weights for terms are considered (such as idf), while local in-document frequency (eg., tf) is suppressed by either taking a log or replacing it with a constant. The effect of this 'hot spot' retrieval is shown below in the ranking of relevant documents within the top 1000 retrieved documents for topic 65:

Full tf.idf retrieval

DOCUMENT ID	RANK	SCORE
WSJ870304-0091	4	12228
WSJ891017-0156	7	9771
WSJ920226-0034	14	8921
WSJ870429-0078	26	7570
WSJ870205-0078	33	6972
WSJ880712-0033	34	6834
WSJ920116-0002	37	6580
WSJ910328-0013	74	4872
WSJ910830-0140	80	4701
WSJ890804-0138	102	4134
WSJ911212-0022	104	4065
WSJ870825-0026	113	3922
WSJ880712-0023	135	3654
WSJ871202-0145	153	3519

Hot-spot idf-dominated with N=20

DOCUMENT ID	RANK	SCORE
WSJ920226-0034	1	11955
WSJ870304-0091	3	11565
WSJ870429-0078	5	9997
WSJ920116-0002	7	9997
WSJ910830-0140	11	8792
WSJ870205-0078	20	8402
WSJ910328-0013	29	8402
WSJ880712-0033	71	6834
WSJ880712-0023	72	6834
WSJ891017-0156	87	6834
WSJ890804-0138	92	6834
WSJ911212-0022	111	6834
WSJ871202-0145	124	6834

The final ranking is obtained by merging the two rankings by score. While some of the recall may be sacrificed ('hot spot' retrieval has, understandably, lower recall than full query retrieval, and this becomes the lower bound on recall for the combined ranking) the combined ranking precision has been consistently better than in either of the original rankings: an average improvement is 10-12% above the tf.idf run precision (which is often stronger of the two).

CONCLUSIONS

We presented in some detail our natural language information retrieval system consisting of an advanced NLP module and a 'pure' statistical core engine. While many problems remain to be resolved, including the question of adequacy of term-based representation of document content, we attempted to demonstrate that the architecture described here is nonetheless viable. In particular, we demonstrated that natural language processing can now be done on a fairly large scale and that its speed and robustness can match those of traditional statistical programs such as key-word indexing or statistical phrase extraction. We suggest, with some caution until more experiments are run, that natural language processing can be very effective in creating appropriate search queries out of user's initial specifications which can be frequently imprecise or vague.

On the other hand, we must be aware of the limits of NLP technologies at our disposal. While part-of-speech tagging, lexicon-based stemming, and parsing can be done on large amounts of text (hundreds of millions of words and more), other, more advanced processing involving conceptual structuring, logical forms, etc., is still beyond reach, computationally. It may be assumed that these super-advanced techniques will prove even more effective, since they address the problem of representation-level limits; however the experimental evidence is sparse and necessarily limited to rather small scale tests (e.g., Mauldin, 1991).

ACKNOWLEDGEMENTS

We would like to thank Donna Harman of NIST for making her PRISE system available to us. We would also like to thank Ralph Weischedel and Heidi Fox of BBN for providing and assisting in the use of the part of speech tagger. This paper is based upon work supported by the Advanced Research Project Agency under Contract N00014-90-J-1851 from the Office of Naval Research, under Contract N00600-88-D-3717 from PRC

Inc., and the National Science Foundation under Grant IRI-93-02615. We also acknowledge support from the Canadian Institute for Robotics and Intelligent Systems (IRIS).

REFERENCES

Broglio, John and W. Bruce Croft. 1993. "Query Processing for Retrieval from Large Text Bases." Proceedings of ARPA HLT Workshop, March 21-24, Plainsboro, NJ.

Church, Kenneth Ward and Hanks, Patrick. 1990. "Word association norms, mutual information, and lexicography." *Computational Linguistics*, 16(1), MIT Press, pp. 22-29.

Crouch, Carolyn J. 1988. "A cluster-based approach to thesaurus construction." Proceedings of ACM SIGIR-88, pp. 309-320.

Grefenstette, Gregory. 1992. "Use of Syntactic Context To Produce Term Association Lists for Text Retrieval." Proceedings of SIGIR-92, Copenhagen, Denmark. pp. 89-97.

Grishman, Ralph, Lynette Hirschman, and Ngo T. Nhan. 1986. "Discovery procedures for sublanguage selectional patterns: initial experiments". *Computational Linguistics*, 12(3), pp. 205-215.

Grishman, Ralph and Tomek Strzalkowski. 1991. "Information Retrieval and Natural Language Processing." Position paper at the workshop on Future Directions in Natural Language Processing in Information Retrieval, Chicago.

Harman, Donna. 1988. "Towards interactive query expansion." Proceedings of ACM SIGIR-88, pp. 321-331.

Harman, Donna and Gerald Candela. 1989. "Retrieving Records from a Gigabyte of text on a Minicomputer Using Statistical Ranking." *Journal of the American Society for Information Science*, 41(8), pp. 581-589.

Hindle, Donald. 1990. "Noun classification from predicate-argument structures." Proc. 28 Meeting of the ACL, Pittsburgh, PA, pp. 268-275.

Kwok, K.L., L. Papadopoulos and Kathy Y.Y. Kwan. 1993. "Retrieval Experiments with a Large Collection using PIRCS." Proceedings of TREC-1 conference, NIST special publication 500-207, pp. 153-172.

Lewis, David D. and W. Bruce Croft. 1990. "Term Clustering of Syntactic Phrases". Proceedings of ACM SIGIR-90, pp. 385-405.

Mauldin, Michael. 1991. "Retrieval Performance in Ferret: A Conceptual Information Retrieval System." Proceedings of ACM SIGIR-91, pp. 347-355.

Meteer, Marie, Richard Schwartz, and Ralph Weischedel. 1991. "Studies in Part of Speech Labeling." Proceedings of the 4th DARPA Speech and Natural Language Workshop, Morgan-Kaufman, San Mateo, CA. pp. 331-336.

Sager, Naomi. 1981. *Natural Language Information Processing.* Addison-Wesley.

Sparck Jones, Karen. 1972. "Statistical interpretation of term specificity and its application in retrieval." *Journal of Documentation*, 28(1), pp. 11-20.

Sparck Jones, K. and E. O. Barber. 1971. "What makes automatic keyword classification effective?" *Journal of the American Society for Information Science*, May-June, pp. 166-175.

Sparck Jones, K. and J. I. Tait. 1984. "Automatic search term variant generation." *Journal of Documentation*, 40(1), pp. 50-66.

Strzalkowski, Tomek and Barbara Vauthey. 1991. "Fast Text Processing for Information Retrieval." Proceedings of the 4th DARPA Speech and Natural Language Workshop, Morgan-Kaufman, pp. 346-351.

Strzalkowski, Tomek and Barbara Vauthey. 1992. "Information Retrieval Using Robust Natural Language Processing." Proc. of the 30th ACL Meeting, Newark, DE, June-July. pp. 104-111.

Strzalkowski, Tomek. 1992. "TTP: A Fast and Robust Parser for Natural Language." Proceedings of the 14th International Conference on Computational Linguistics (COLING), Nantes, France, July 1992. pp. 198-204.

Strzalkowski, Tomek. 1993. "Natural Language Processing in Large-Scale Text Retrieval Tasks." Proceedings of the First Text REtrieval Conference (TREC-1), NIST Special Publication 500-207, pp. 173-187.

Strzalkowski, Tomek. 1993. "Robust Text Processing in Automated Information Retrieval." Proc. of ACL-sponsored workshop on Very Large Corpora. Ohio State Univ. Columbus, June 22.

Strzalkowski, Tomek, and Peter Scheyen. 1993. "An Evaluation of TTP Parser: a preliminary report." Proceedings of International Workshop on Parsing Technologies (IWPT-93), Tilburg, Netherlands and Durbuy, Belgium, August 10-13.

Design and Evaluation of the CLARIT–TREC-2 System

David A. Evans[1,2] and Robert G. Lefferts[2]

[1]Laboratory for Computational Linguistics
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890

and [2]CLARIT Corporation
Suite 200A, 319 South Craig St.
Pittsburgh, Pennsylvania 15213-3726

1 Introduction

The CLARIT team used the opportunity of the TREC-2 evaluations to explore several facets of the CLARIT system. In particular, given the performance of the CLARIT system on TREC-1 tasks (Evans *et al.* 1993), we focused our attention on evaluating

1. *fully-automatic processing* of topics and potentially-relevant documents and
2. *topic/query augmentation* using CLARIT thesaurus-discovery techniques.

All of the results we report in this paper follow from straightforward applications of base-level CLARIT processing, utilizing essentially the same CLARIT components that were employed in the CLARIT–TREC-1 system. The general improvements we observe in CLARIT–TREC-2 processing are attributable to modifications (especially simplifications) in processing steps and in the settings of system variables.

In the following sections, we describe the CLARIT–TREC-2 system, report our official processing results, and offer a brief analysis of performance. In addition, we report on several subsequent experiments we have conducted on the TREC-2 collection that test the parameters of the CLARIT–TREC-2 system and identify sources of immediate improvements in processing.

2 CLARIT–TREC-2 System Description and Processing Method

The CLARIT–TREC-2 system reflects a re-organization of the tools and techniques employed in the CLARIT–TREC-1 system. One of our principal goals was to streamline CLARIT processing and to establish a baseline method that is amenable to parameterization and analysis. As a consequence, the flow of data in the CLARIT–TREC-2 system is simple, straightforward, and efficient; furthermore, all CLARIT processing is fully automatic.

2.1 Changes from TREC-1

The essential differences between the CLARIT–TREC-1 and TREC-2 systems are in the preparation and evaluation of queries (TREC-2 "topics") and the automation of steps designed to identify and process potentially relevant documents for use in query augmentation. The following summaries highlight these points.

- **One-Pass Querying.** The CLARIT–TREC-1 system employed a two-step process to retrieve documents—a first pass for partitioning ("evoking") and a second pass for final ranking ("discrimination"). This has been eliminated in the CLARIT–TREC-2 system. Querying takes place in one step over the entire collection using vector-space-retrieval methods.
- **Automatic Query Creation.** The CLARIT–TREC-1 system was categorized as a "manual" system, though the required manual intervention was minimal. In particular, users were expected to assign an *importance coefficient* (with possible values "1", "2", or "3") to the CLARIT-parsed terms in a topic statement and possibly also to add terms to or delete terms from the CLARIT-generated list. In the CLARIT–TREC-2 system, the importance coefficient is assigned automatically by simple heuristics (described below). While users are still free to modify coefficients or terms, such intervention is not required. All "CLARTA" results reported in this paper reflect processing in which queries were fully automatically prepared by the CLARIT system, without review or modification.
- **Automatic Retrieval Refinement.** When processing ad-hoc queries, the CLARIT–TREC-1 system required that the user evaluate a few of the top-ranked retrieved documents. User-nominated documents were processed to identify terms for use in supplementing the source query. In the CLARIT–TREC-2 system, user evaluations are not required. Initial querying is accurate enough to support the automatic processing of the highest-scoring retrieved documents without 'inspection'.

- **Sub-Document Processing.** The CLARIT–TREC-1 system treated all documents as whole texts; retrieval 'scores' were calculated over full documents. The CLARIT–TREC-2 system treats all documents as collections of one or more sub-documents, operationalized as variable-sized units of approximately paragraph length. Such units are used as the basis for all statistical calculations and for measuring 'similarity' to a query. A full document is assigned the score (*e.g.*, for ranking) of the highest-scoring sub-document it contains.

2.2 Processing Method

Figure 1 offers a schematic overview of processing in the CLARIT–TREC-2 system. All topics were parsed for noun phrases. These, in turn, were either manually ("CLARTM") or automatically ("CLARTA") assigned weights (values "1", "2", or "3") for 'importance'. The terms for each topic were automatically supplemented with terms from a (pseudo-)thesaurus, automatically extracted from available known-relevant documents (in the case of routing topics) or from the top-ranked sub-documents returned in a first-pass querying of the TREC-2 collection (in the case of ad-hoc topics). All instances of retrieval took place over the applicable full set of documents, which had undergone an initial round of CLARIT processing (parsing).

The CLARIT–TREC-2 system incorporates a vector-space retrieval system that uses several CLARIT-specific techniques to improve retrieval results. The principal techniques involve the use of (1) natural-language processing to identify and normalize indexing terms, (2) fully automatic query augmentation based on CLARIT thesaurus discovery, and (3) simple text-analysis heuristics to approximate the effect of more sophisticated discourse analysis of texts. These techniques are described in greater detail in the following sections.

2.2.1 Natural-Language Processing

CLARIT natural-language processing (NLP) encompasses an inflectional morphological analyzer for word recognition and normalization and a deterministic rule-based parser for phrase identification. For TREC-2 processing, only simplex noun phrases (NPs) were used. Simplex NPs are phrasal constitutents that include the modifiers and head noun(s) of an NP but not the post-head prepositional phrases, relative clauses, or verb constructions. The CLARIT parser can provide a more complex linguistic analysis of texts, but such additional detail was not used in TREC-2 experiments.

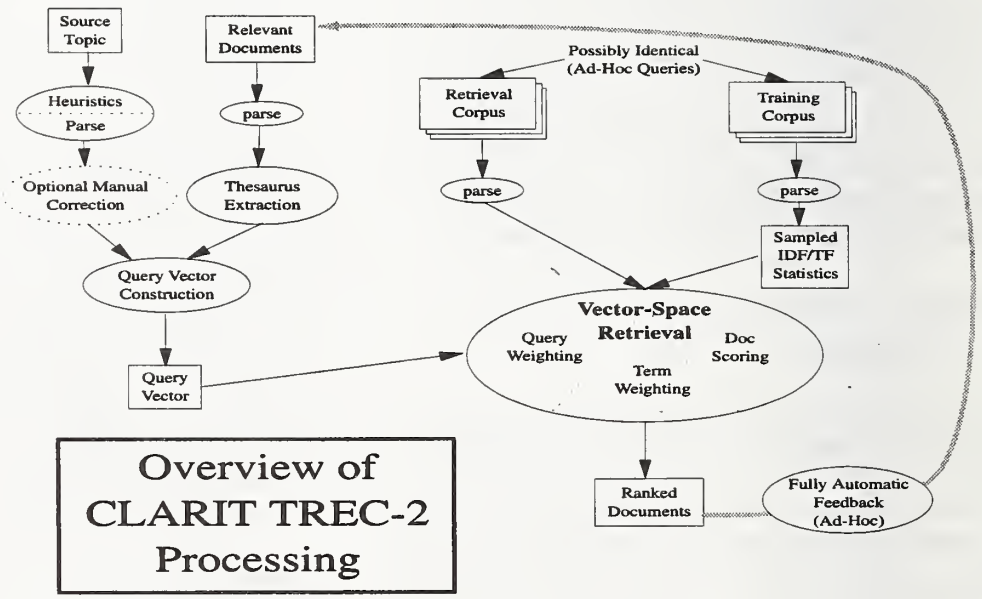


Figure 1: Overview of CLARIT–TREC-2 Processing

2.2.2 Query Augmentation

CLARIT thesaurus discovery is a process that can identify a set of core, representative terminology in a collection of documents. If the documents are relatively homogeneous topically, the resulting 'first-order' thesaurus can be regarded as a broad 'signature' of the topic of the collection. Typically, the procedure is most reliable when the collection is large—2 megabytes or more of text. In the CLARIT–TREC-2 system, however, thesaurus discovery was used on the relatively small collections of relevant documents for each topic. For each routing topic, the document collection used to establish a first-order thesaurus consisted of the training set of relevant documents for each topic; for each ad-hoc topic, a set of the top-ranked sub-documents from among the documents automatically retrieved via an initial pass of querying over the corpus. Terms in the discovered thesaurus were used to supplement the terms in the source query (topic). Thus, in the case of ad-hoc topics, the procedure represents an approach to query augmentation based on fully automatic feedback.

2.2.3 Text Analysis Heuristics

Texts can have complex and interesting discourse structure, which often provides clues about the 'topic(s)' and the 'important' information in a document. In general, it is very difficult to exploit text-structure information reliably in retrieval tasks over large collections of heterogeneous documents. Nevertheless, the CLARIT–TREC-2 system applied two simple processing techniques to topics and corpus documents to attempt to capture information encoded in rhetorical structure.

First, all training and corpus documents were divided into paragraph-sized units called "sub-documents". This procedure is sensitive both to the 'normal' demarcation of paragraphs (successive blank lines) and also to the total length of the text (measured in numbers of sentences). After documents are partioned into sub-documents, the sub-document is taken as the basic unit for subsequent processing, *viz.*, in collecting statistics and in scoring retrieval.

Second, terms extracted from a topic description were assigned importance coefficients based on their locations in the topic text. Terms found in the first paragraph are given a weight of 3; terms in the second paragraph, 2; and all other terms, 1.

Of course, both techniques exploit possibly idiosyncratic characterisitics of the TREC-2 processing task. The use of scoring over sub-documents is clearly sensitive to the TREC definition of document relevancy, *viz.*, that a document is relevant regardless of length if it contains a single relevant sentence. Furthermore, automatic term importance weighting is possible only because of the formal discourse structure of TREC topics; it would not neccessarily apply to other presentations of topics or queries and certainly would not apply to free text in general.

We observe, however, that there is no one set of techniques that will perform optimally in every information-retrieval (IR) situation. We emphasize, instead, that one important measure of a system's utility and adaptability is its ability to take advantage of exploitable features in a given IR task.

2.3 System Performance Notes

All CLARIT–TREC-2 processing took place on DEC 3000/400 (ALPHA/AXP) workstations running DEC OSF/1. One system had 128 megabytes of RAM, one 64 megabytes, and two 32 megabytes. Realized performance was considerably slower than would be expected given the clockrate (133.33 MHz) of the DEC 3000/400's CPU. In fact, because of suboptimal compilers for the 64-bit architecture, performance was perhaps two times slower than the maximum possible. One pass over the entire TREC-2 collection for 50 topics (processed simultaneously) required approximately four hours, running on the four machines in parallel. The processing of a single topic is proportionally faster, requiring approximately 20 minutes on a single machine.

3 Results

The CLARIT team processed all the topics in both the "routing" and "ad-hoc" categories and worked with the full set of data. Two sets of results were submitted for each category, corresponding to the "manual" ("CLARTM") and fully-"automatic" ("CLARTA") processing approaches taken with the topics.

3.1 General Summary of Official Results

Table 1 gives the official CLARIT–TREC-2 system routing results as reported by NIST. A graph of the precision–recall curves for the two sets of results is given in Figure 2. The total number of documents retrieved under the routing task was 6,785 (CLARTM) and 6,811 (CLARTA), representing, respectively, 64.69% and 64.93% of the total known relevants (10,489).

Table 2 gives the official CLARIT–TREC-2 system ad-hoc query results as reported by NIST. A graph of the precision–recall curves for the two sets of results is given in Figure 3. The total number of documents retrieved under the ad-hoc query task was 8,229 (CLARTM) and 8,109 (CLARTA), representing, respectively, 76.30% and 75.19% of the total known relevants (10,785).

The graph in Figure 4 shows the average precision score for each process at N documents, for selected values of N. It should be noted that the maximum possible precision score at 500 and 1,000 documents is less than 100%. In particular, the average number of relevants per routing topic is 209.78; this corresponds to a maximum precision of 41.96% at 500 documents and 20.98% at 1,000 documents. The average number of relevants per ad-hoc query topic is 215.70; this corresponds to a maximum precision of 43.14% at 500 documents and 21.57% at 1,000 documents.

Tables 3 and 4 provide another view of total performance. The numbers in each cell give the number of times the CLARIT–TREC-2 system produced results above, equal to, or below the median for all TREC-participant systems. Numbers in brackets give the instances of 'extreme' performance—best and worst—among all systems. For the routing topics, for example, CLARIT retrieval results at 1,000 documents were better than the median 36 times in both "manual" and "automatic" modes; CLARIT scored the maximum 10 and 11 times, respectively. For the ad-hoc query topics, CLARIT retrieval results at 1,000 documents were better than the median 44 times in "manual" mode and 42 times in "automatic" mode; CLARIT scored the maximum 4 and 9 times, respectively.

3.2 CLARIT Automatic vs. Manual Modes of Processing

In both tasks (routing and ad-hoc querying), CLARIT–TREC-2 automatic processing results are virtually identical to manual results. This confirms our hypothesis that the principal contribution to performance derives from (1) the base-level CLARIT process (using linguistic phrases as information units) and (2) the effect of query augmentation via thesaural terms. On this latter point, we note that, on average, the final query vector for a topic will contain many more terms that derive from thesaurus extraction than terms that derive from the source topic. In general, then, when reliable information is available (as in sample known relevants or highly-likely relevants returned in a first-pass retrieval), the CLARIT process will succeed in finding good supplemental terminology for a topic and the overall effects of manual intervention will be minimized.[1]

Figures 5 and 6 illustrate the relative absence of a positive effect for manual intervention in the selection and weighting of query terms. There are approximately as many instances of decreased performance as there are instances of increased performance. Most topics show very little percentage difference in numbers of documents returned;[2] this is especially underscored in the results for routing topics at 1,000 documents.

4 Analysis

4.1 CLARIT Precision

As in TREC-1, CLARIT precision–recall curves demonstrate very high precision at low percentages of recall. The first few documents returned by the system are extremely likely to be relevant for the given topic. This fact of CLARIT processing was successfully exploited in the TREC-2 processing method: query augmentation was possible because there was, in general, a good concentration of topic-relevant information among the sub-documents of the first-pass returned documents.

As shown in Figure 4, precision remains quite stable for all methods across the first 30 documents retrieved and is relatively high across the full retrieved set of 1,000 documents.

5 Query Augmentation Experiments

A distinguishing feature of the CLARIT–TREC-2 system is the use of fully-automatic query augmentation. As noted above, the selection of terms for query augmentation depends on (1) the selection of a source set of known- or nominated-relevant documents and (2) the application of the CLARIT thesaurus-discovery procedure. Since the size (and quality) of the source set of documents can vary and since CLARIT thesaurus-discovery processing can be adjusted to nominate relatively greater or fewer numbers of terms, the 'query-augmentation' facet of the CLARIT process is a natural source of potential variation in system performance.

---

[1] Of course, there may be some forms of manual intervention—not utilized in the CLARIT 'manual' process—that would have effects dramatically better than the CLARIT automatic process. We know of no such process that can be applied efficiently to arbitrary topics and databases, however.

[2] Indeed, even in the absolute number of documents returned for each topic—not shown in the figures—there is very little difference.

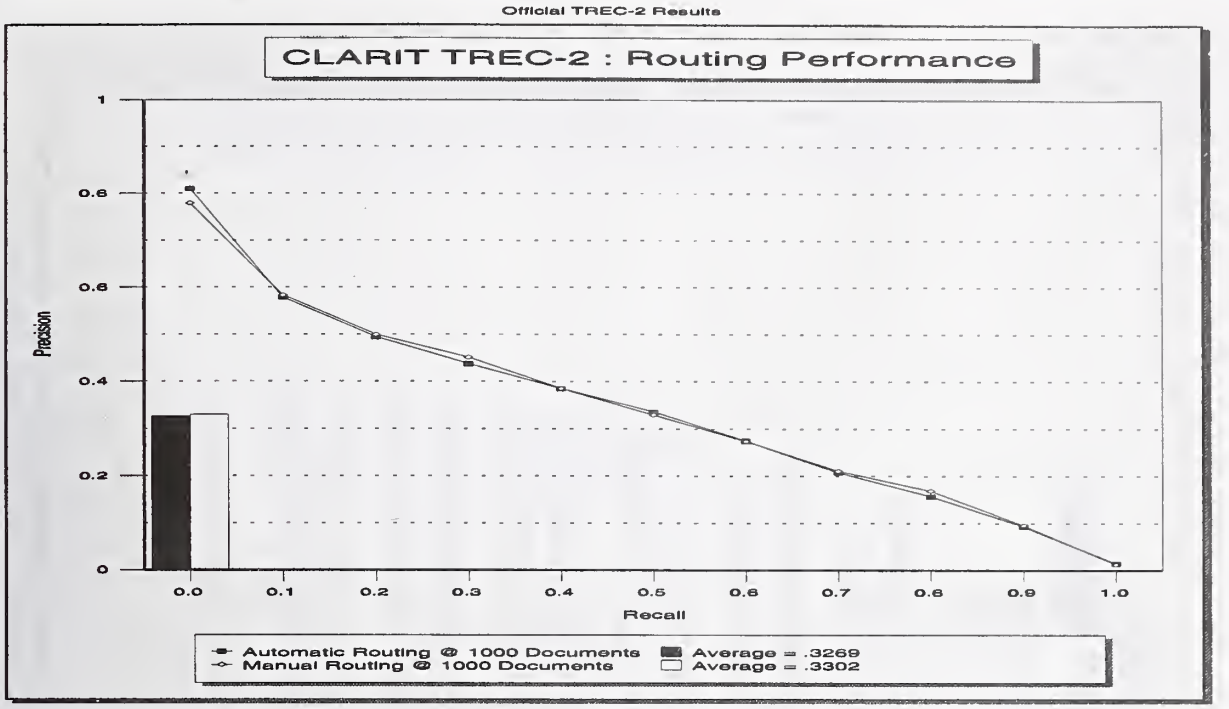


Figure 2: P–R Curve for Routing Over 1000 Docs—CLARTM and CLARTA

CLARTM Routing Queries, 51–100	
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10489
Rel_ret:	6785
Interpolated Recall–Precision Averages:	
at 0.00	0.7791
at 0.10	0.5825
at 0.20	0.4987
at 0.30	0.4505
at 0.40	0.3848
at 0.50	0.3295
at 0.60	0.2739
at 0.70	0.2109
at 0.80	0.1683
at 0.90	0.0951
at 1.00	0.0126
Average Precision (non-interpolated) over all rel docs:	0.3302
Precision:	
At 5 docs:	0.6120
At 10 docs:	0.5940
At 15 docs:	0.5800
At 20 docs:	0.5700
At 30 docs:	0.5527
At 100 docs:	0.4396
At 200 docs:	0.3436
At 500 docs:	0.2176
At 1000 docs:	0.1357
R-Precision (precision after R retrieved)	
Exact	0.3642

CLARTA Routing Queries, 51–100	
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10489
Rel_ret:	6811
Interpolated Recall–Precision Averages:	
at 0.00	0.8101
at 0.10	0.5781
at 0.20	0.4941
at 0.30	0.4368
at 0.40	0.3851
at 0.50	0.3358
at 0.60	0.2749
at 0.70	0.2074
at 0.80	0.1561
at 0.90	0.0933
at 1.00	0.0141
Average Precision (non-interpolated) over all rel docs:	0.3269
Precision:	
At 5 docs:	0.6200
At 10 docs:	0.6060
At 15 docs:	0.5893
At 20 docs:	0.5730
At 30 docs:	0.5460
At 100 docs:	0.4346
At 200 docs:	0.3416
At 500 docs:	0.2184
At 1000 docs:	0.1362
R-Precision (precision after R retrieved)	
Exact	0.3646

Table 1: Offical Results, Routing Topics 51–100

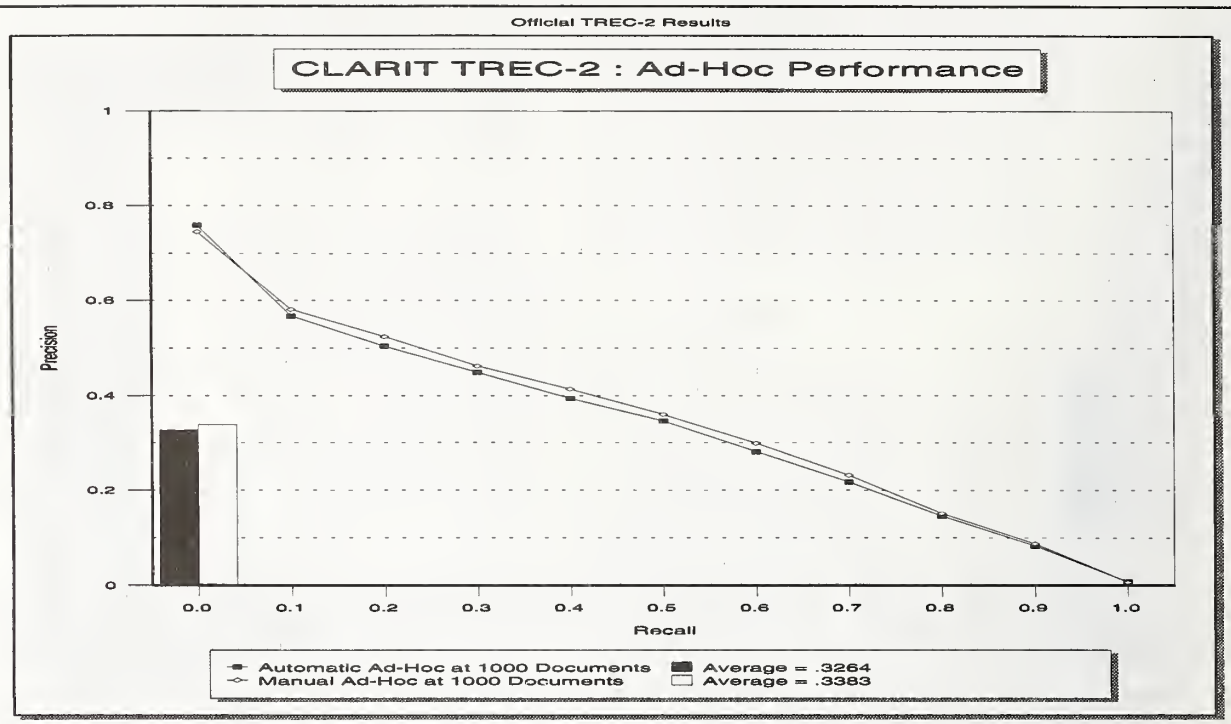


Figure 3: P–R Curve for Ad-Hoc Queries Over 1000 Docs—CLARTM and CLARTA

CLARTM Ad-Hoc Queries, 101–150	
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	8229
Interpolated Recall–Precision Averages:	
at 0.00	0.7455
at 0.10	0.5811
at 0.20	0.5240
at 0.30	0.4622
at 0.40	0.4135
at 0.50	0.3593
at 0.60	0.2983
at 0.70	0.2312
at 0.80	0.1516
at 0.90	0.0874
at 1.00	0.0062
Average Precision (non-interpolated) over all rel docs:	0.3383
Precision:	
At 5 docs:	0.5840
At 10 docs:	0.5740
At 15 docs:	0.5640
At 20 docs:	0.5590
At 30 docs:	0.5433
At 100 docs:	0.4846
At 200 docs:	0.3975
At 500 docs:	0.2601
At 1000 docs:	0.1646
R-Precision (precision after R retrieved) Exact	0.3741

CLARTA Ad-Hoc Queries, 101–150	
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	8109
Interpolated Recall–Precision Averages:	
at 0.00	0.7587
at 0.10	0.5676
at 0.20	0.5038
at 0.30	0.4489
at 0.40	0.3938
at 0.50	0.3455
at 0.60	0.2806
at 0.70	0.2172
at 0.80	0.1463
at 0.90	0.0827
at 1.00	0.0070
Average Precision (non-interpolated) over all rel docs:	0.3264
Precision:	
At 5 docs:	0.5800
At 10 docs:	0.5680
At 15 docs:	0.5547
At 20 docs:	0.5490
At 30 docs:	0.5253
At 100 docs:	0.4644
At 200 docs:	0.3874
At 500 docs:	0.2542
At 1000 docs:	0.1622
R-Precision (precision after R retrieved) Exact	0.3645

Table 2: Offical Results, Ad-Hoc Topics 101–150

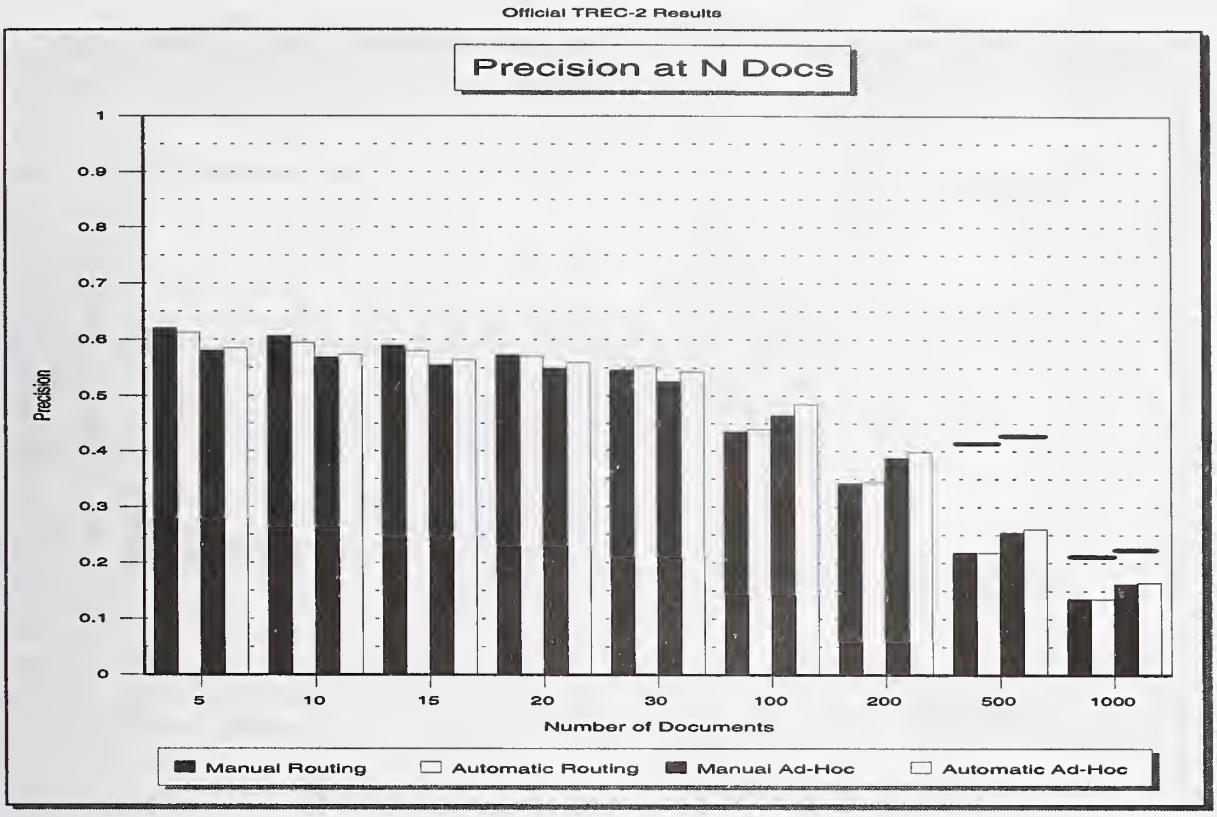


Figure 4: Comparative Precision at N Documents

	>Median		=Median		<Median	
	CLARTM	CLARTA	CLARTM	CLARTA	CLARTM	CLARTA
Precision	37 [1]	34 [0]	0	4	13 [0]	12 [0]
Rels in Top 100	32 [2]	29 [2]	6	8	12 [1]	13 [1]
Rels in Top 1000	36 [10]	36 [11]	5	6	9 [0]	8 [0]

Table 3: Summary of Results for Routing (51–100)

	>Median		=Median		<Median	
	CLARTM	CLARTA	CLARTM	CLARTA	CLARTM	CLARTA
Precision	38 [0]	34 [3]	4	7	8 [0]	9 [0]
Rels in Top 100	39 [1]	31 [0]	3	2	9 [0]	17 [0]
Rels in Top 1000	44 [4]	42 [9]	2	2	4 [0]	6 [0]

Table 4: Summary of Results for Ad-Hoc Queries (101–150)

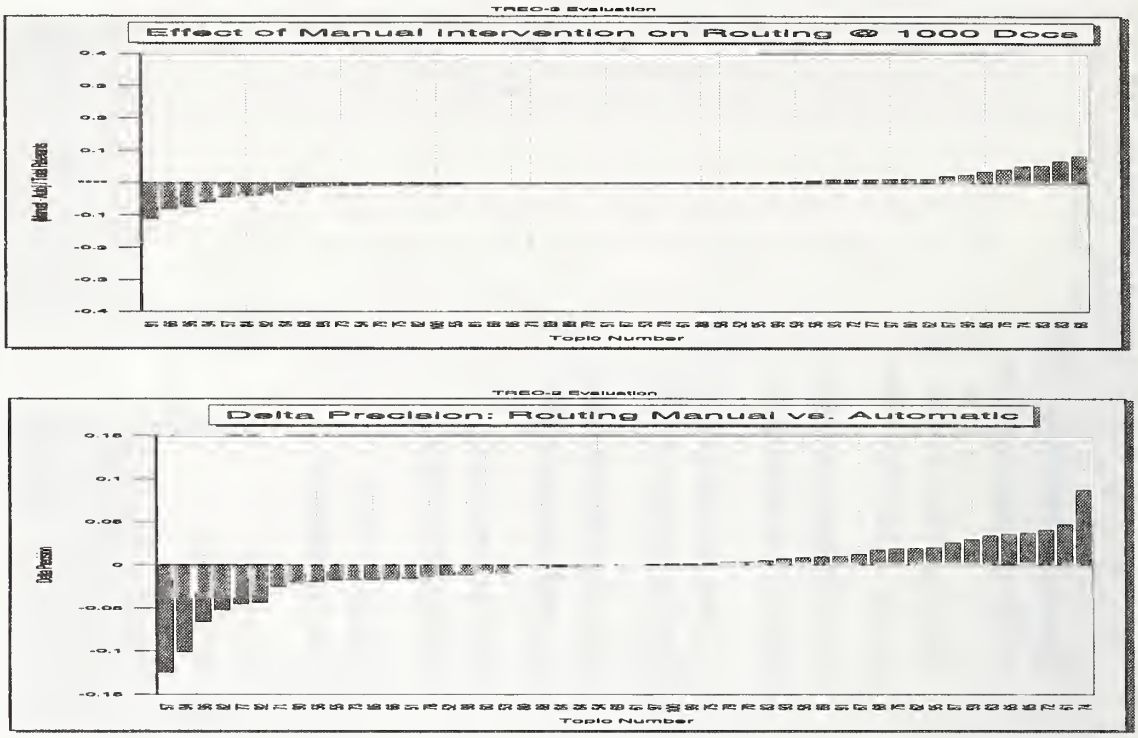


Figure 5: Manual vs. Automatic Preparation of Routing Queries

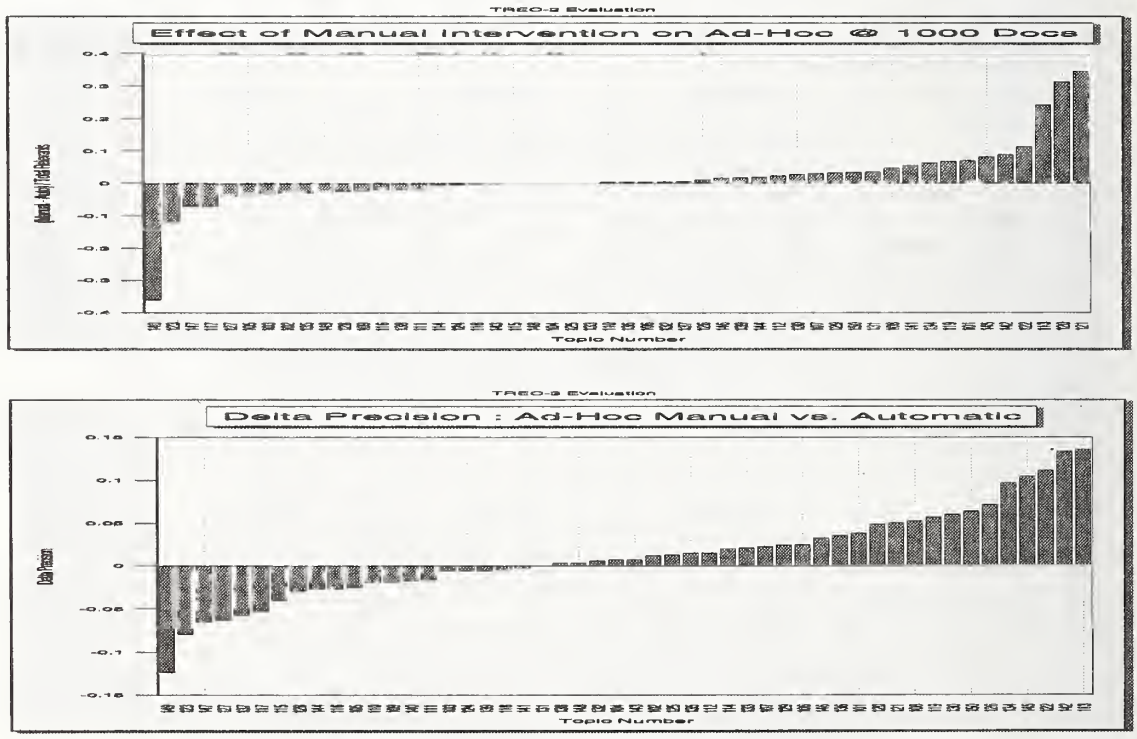


Figure 6: Manual vs. Automatic Preparation of Ad-Hoc Queries

Using TREC-2 processing results as a baseline, we have begun exploring several of the parameters of query augmentation in the CLARIT–TREC-2 system. The first set of experiments focuses on three parameters:

- The use of query augmentation compared to a procedure involving no query augmentation;
- The size of relevant document samples as source collections for thesaurus extraction; and
- The 'threshold' chosen for the thesaurus discovery process.

5.1 Query Augmentation vs. No Augmentation

In order to verify that CLARIT query augmentation techniques do have a positive effect on retrieval results, we have compared the official TREC-2 results against experimental runs that do not take advantage of any augmentation. For the routing queries, we re-ran the routing task using only the raw queries as vectors (consisting of terms from the topic statement only). The effects for manual and automatic modes are shown in Figure 7. In the case of the ad-hoc queries, the unaugmented results are simply the results of the initial round of querying, before automatic feedback has taken place. The effects for both modes are shown in Figure 8.

Query augmentation using known-relevant documents, as in the routing experiment, has a dramatic effect on system performance. The manual routing task showed a 69% improvement with augmentation, while automatic routing improved by 76%. Figure 9 shows the effect of query augmentation for individual routing topics, calculated as the difference in average precision between augmented and unaugmented queries. Note that very few queries do not improve with augmentation; most show great improvement.

Query augmentation is not as effective when used for automatic feedback in processing ad-hoc queries, but it still results in substantial improvements. For manual ad-hoc queries, we found a 21% improvement when using query augmentation; for automatic ad-hoc queries, a 22% improvement. Obviously, the use of known-relevant documents in the case of routing topics has an impact on query augmentation, but it is not the only important factor. The query-by-query effect of augmentation for ad-hoc queries is shown in Figure 10. While the effect is not as great as with routing topics, most queries show improvement with augmentation. By comparing the query-by-query results for automatic and manual modes, one can see that manual intervention in query formulation improves the results of augmentation in specific cases, such as query 121, but the positive effect is difficult to predict. Even without user review, we can have confidence in the ability of query augmentation to improve query results, given reasonably accurate initial query formulation.

We believe that the techniques we used in CLARIT–TREC-2 automatic feedback can be refined to give better documents as input to query augmentation. First, due to engineering and time constraints in TREC-2 processing, we did not select the best sub-documents from the entire corpus. Instead, we selected the best sub-documents from a pool of the best relevant documents, which might not always correspond to the optimal set of sub-documents for query augmentation. Second, the TREC-2 process used an absolute number of sub-documents—the top N—in query augmentation, regardless of the 'similarity' scores of those sub-documents to the source query. While it may not be possible to determine 'absolute' relevance on a query-by-query basis, minimum thresholds might be applied to exclude clearly irrelevant sub-documents, should such be within the N otherwise to be used in augmentation.

5.2 Source Sample Size

In the case of the submitted results of CLARIT–TREC-2 processing, the thesaurus for each routing query was extracted from the set of all its known-relevant training documents. However, we can imagine that many such documents contain some—perhaps a great deal of—information that is not relevant to the topic at hand. (This is especially expected with long documents.) Therefore, we have experimented with alternative approaches to sampling text from known-relevant documents. In particular, we have used a ranking of sub-documents (paragraphs) to nominate candidate 'good' relevant texts and have used variable numbers of sub-documents as source text in thesaurus extraction.

In practice, to select source text for the thesaurus-discovery process for a topic, we run the raw topic vector as a query over the collection of relevant documents, which are partitioned into sub-documents, and return only the top N relevant sub-documents. (In cases where the topic has fewer than N sub-documents in the collection of relevants, all sub-documents are selected.) Our hypothesis is that thesauri extracted from relevant sub-documents will contain greater numbers of 'true-positive' terms related to the topic. We have run the experiment for N = 100, 200, 300, 500, and 1000 as the cutoff for selected sub-documents. Figure 11 gives a sample of the results for routing topics, including the best case of 500 sub-documents.

While the differences for different N are not great, certain trends are evident. First, it is clear that using only the more similar sub-documents from a collection of relevants gives better results than using all relevant full documents. (This represents an important im-

provement over the method employed in the CLARIT–TREC-2 system.) Second, larger sample sizes seem to perform better than smaller ones. In the experiments that we ran, this effect peaks at 500 sub-documents, but such results probably interact with the average number of relevant sub-documents available. As noted previously, we would like ideally to use a variable number of sub-documents for each query, but such an approach requires an accurate measure of 'absolute' relevance.

5.3 Thesaurus Size

CLARIT thesaurus discovery techniques allow for the selection (sampling) of greater or fewer numbers of terms from a collection. In practice, the size of the sample is determined by a real-number threshold between 0.00 and 1.00. A larger threshold results in the inclusion of more general terminology from the document collection; a smaller threshold results in selection of terms that are more specific to the collection. (Intuitively, such variation correlates with the 'breadth' or 'narrowness' of the thesaurus.) The set of terms selected at a larger threshold will always properly include all of the terms that would be selected from the same collection at a smaller threshold. In CLARIT–TREC-2 processing, the threshold was set at 0.50. Note, however, that in the document-sample-size experiments reported above, we used a threshold of 0.75.

Using sub-document samples of relevant documents with $N = 300$ (reflecting the 'second-best' performance obtained in the experiments on sample size), we have explored the effects of using different thesaurus extraction thresholds—at 0.50, 0.75, 0.85, and 0.95 levels. Results for routing topics are given in Figure 12.

As with sample-size variation, differences in performance between increments are not dramatic, but trends do appear. First, the 0.50 thesaurus is clearly inferior and actually performs very much like the baseline CLARIT–TREC-2 system. Such a result indicates that much of the variation observed between baseline CLARIT–TREC-2 processing and our current 'best' technique is due to changes in thesaurus thresholds, rather than changes in the document sample size. Second, while all of the 0.75, 0.85, and 0.95 thesauri have the similar precision at low recall levels, the 0.75 thesaurus performs slightly better in the average case. From this we might hypothesize that the 0.75 value is close to the optimal threshold for thesaurus discovery in the context of query augmentation.

Figure 13 gives the results for the automatic routing task for the optimal number of sub-documents and thesaurus threshold compared to TREC-2 reported results, compared to the unaugmented baseline. Here we can see that a simple refinement to parameter setting yields a 5.5% overall improvement in average precision and 9.1%, 8.8%, and 7.6% improvement in precision at 10%, 20%, and 30% recall levels, respectively. In addition, there is a 6.3% improvement in total relevant documents (7,241 vs. 6,811). Furthermore, our experiments suggest mechanisms, such as measures of absolute relevance, that might result in further significant gains.

6 Conclusion

The CLARIT–TREC-2 system has successfully demonstated the ability to operate as a fully-automatic IR system. Since the performance differences between CLARIT manual and automatic processing modes are negligible, one can use CLARIT in fully-automatic mode and expect high precision and very good recall on retrieval tasks.

The TREC-2 results also demonstrate the efficacy of the CLARIT technique of automatic query augmentation. It is generally difficult for a user to predict whether the addition of terms to a query will have a positive or negative effect on performance. CLARIT query augmentation, using CLARIT thesaurus-discovery techniques, however, shows positive effects. Because the technique is fully automatic, it can be applied either at the time of query formulation (if exemplary relevant texts are known) or at the time of 'first-pass' retrieval. In either case, final results will be improved.

In several experiments, we have already identified two simple adjustments to CLARIT parameters that will improve performance beyond the CLARIT–TREC-2 system baseline. The system is not yet optimized; we expect to make other straightforward improvements.

Many text processing functions currently available in the CLARIT system or near completion were not used on TREC-2 documents. In future evaluations, we plan to utilize some of the more sophisticated functionality in the system. For example, we have been developing grammars for recognizing complex tokens such as proper names, dates, times, monetary values, *etc.*, but did not use token recognition modules in CLARIT–TREC processing. We believe that such token recognition will improve the results for queries involving specific persons or time intervals. Finally, we have also been experimenting with generating sub-corpus-derived equivalence classes for words and terms. We expect to use equivalence classes selectively to supplement thesaurus terms in query augmentation.

In sum, we believe that CLARIT–TREC-2 processing results demonstrate the power of CLARIT tools to solve IR tasks. The CLARIT–TREC-2 system represents only one of many possible configurations of CLARIT modules. In subsequent work, we plan on exploring other configurations.

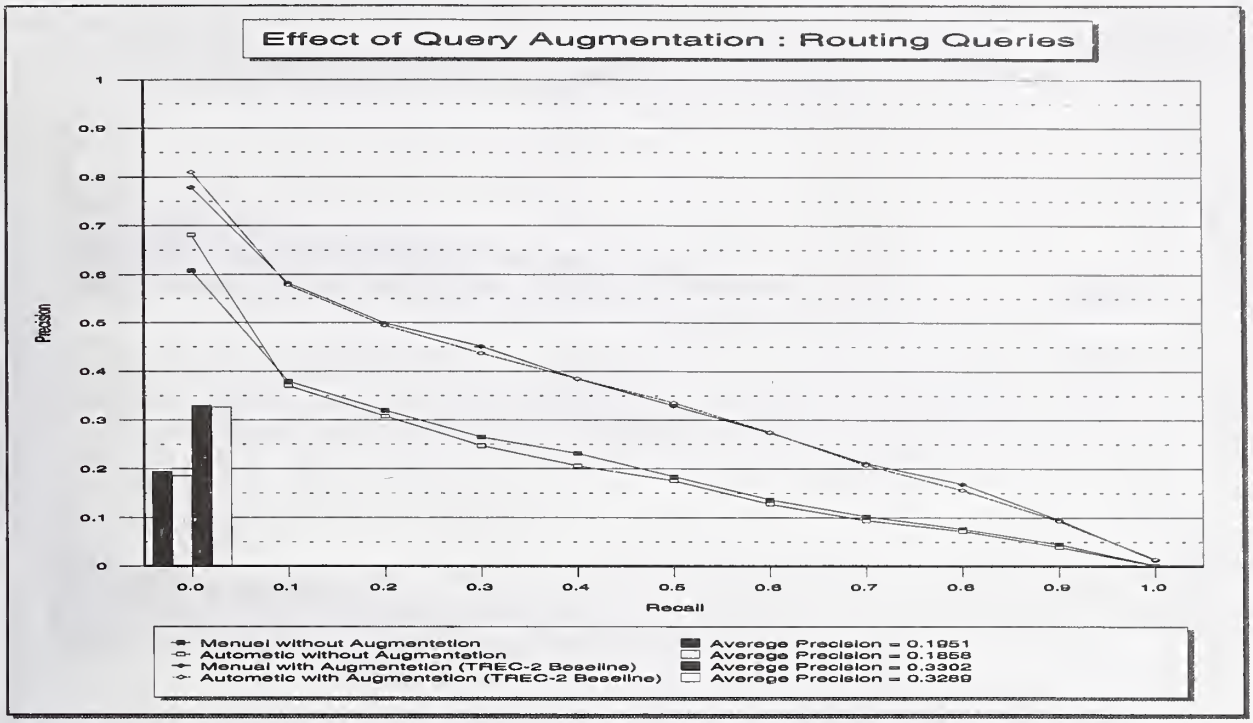


Figure 7: Effect of Query Augmentation on Routing Queries

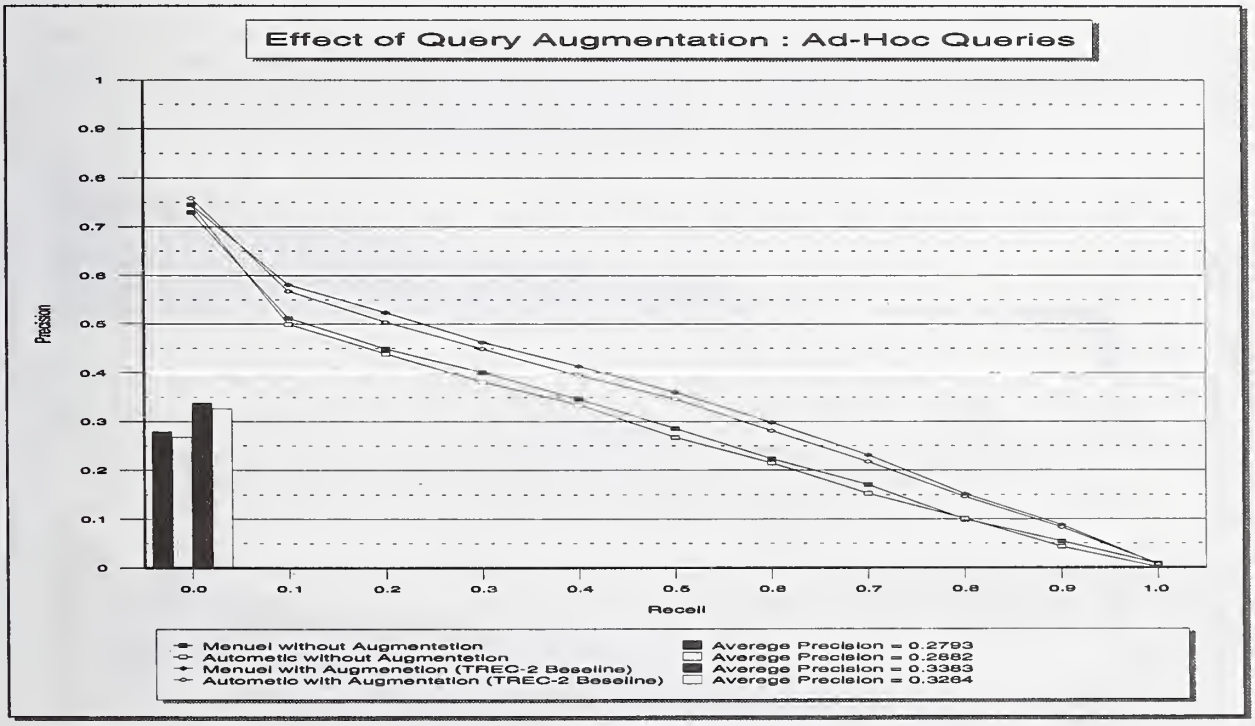


Figure 8: Effect of Query Augmentation on Ad-Hoc Queries

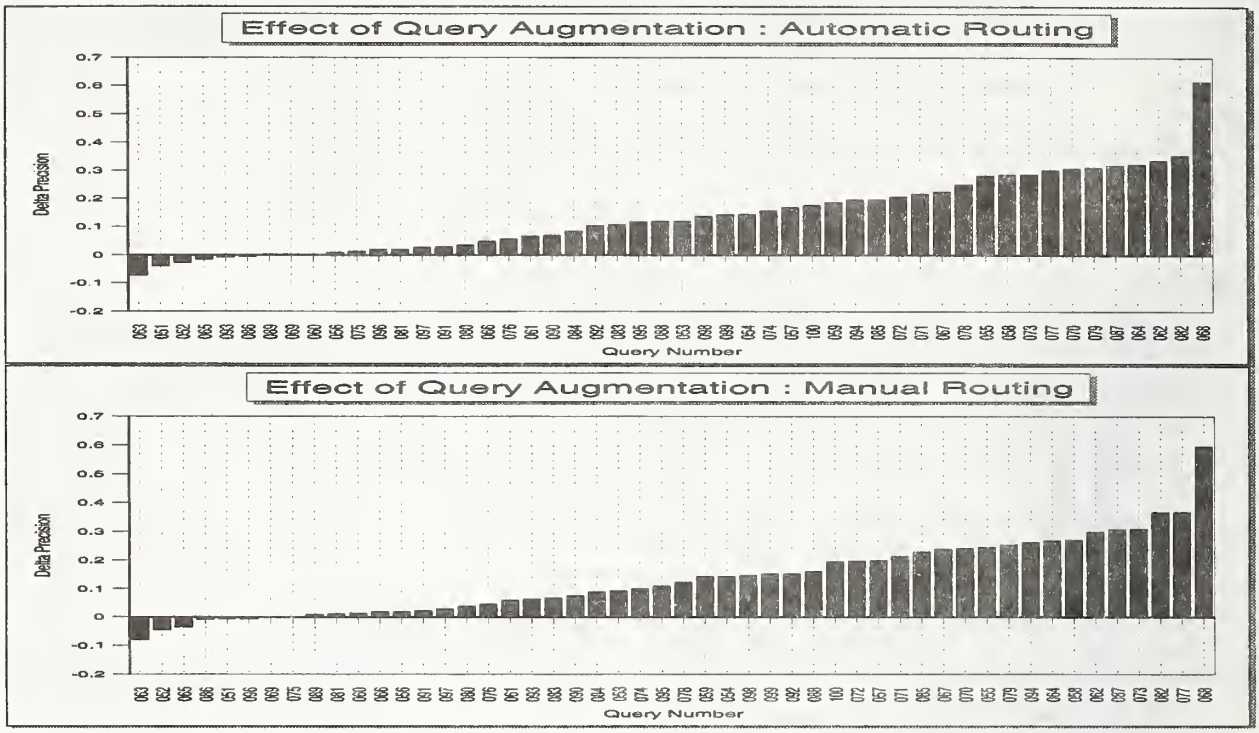


Figure 9: Query-by-Query Analysis of Effect of Query Augmentation on Routing Queries

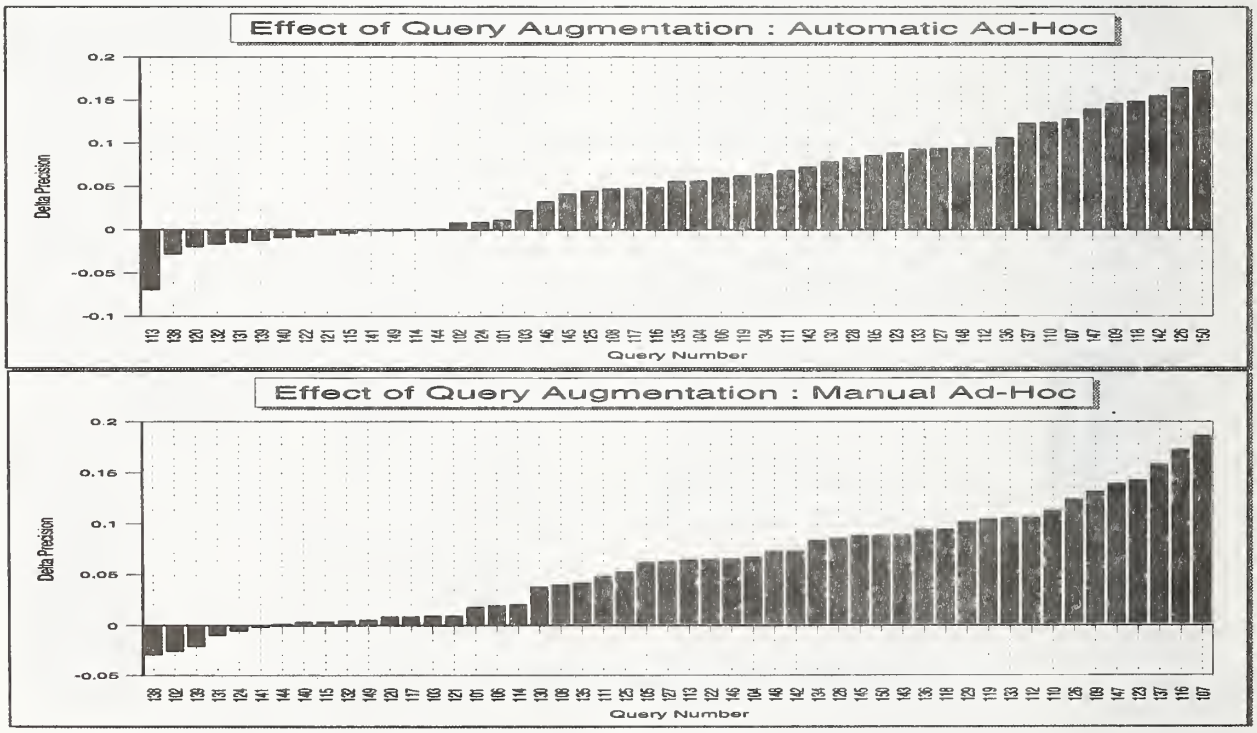


Figure 10: Query-by-Query Analysis of Effect of Query Augmentation on Ad-Hoc Queries

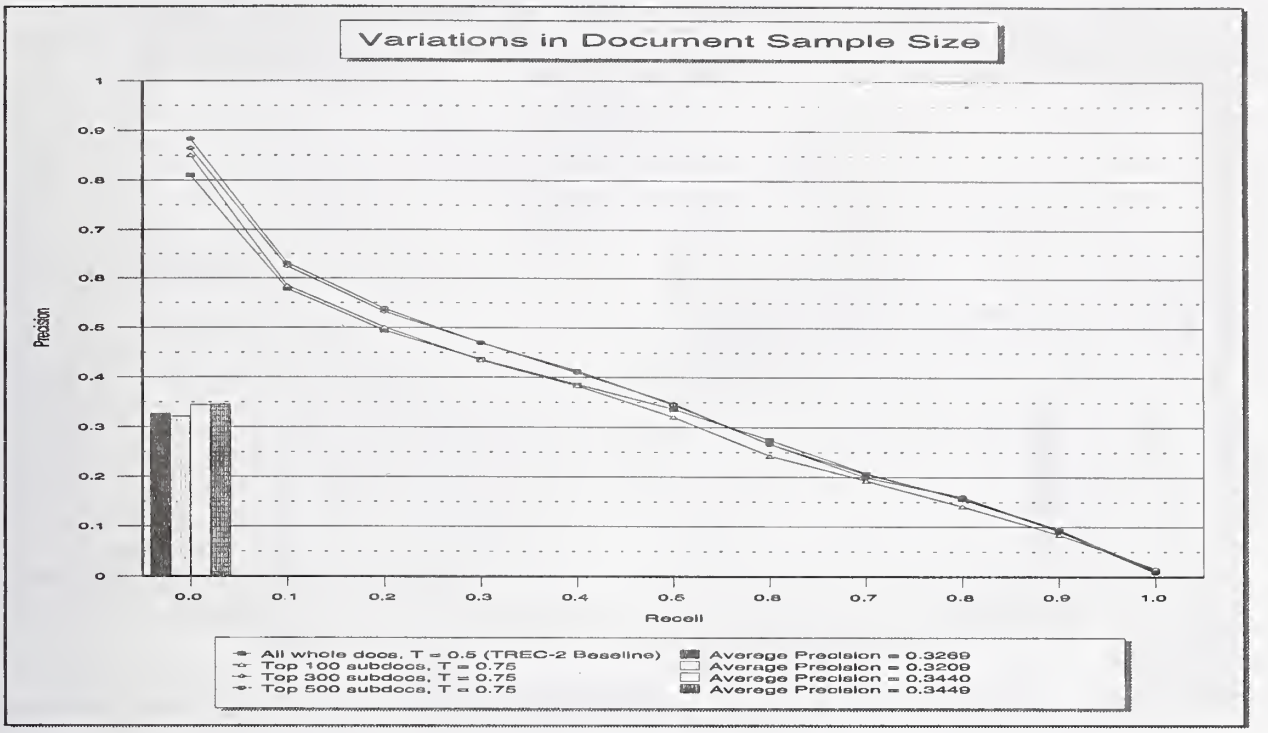


Figure 11: Effect of Sample Size (Routing Topics)

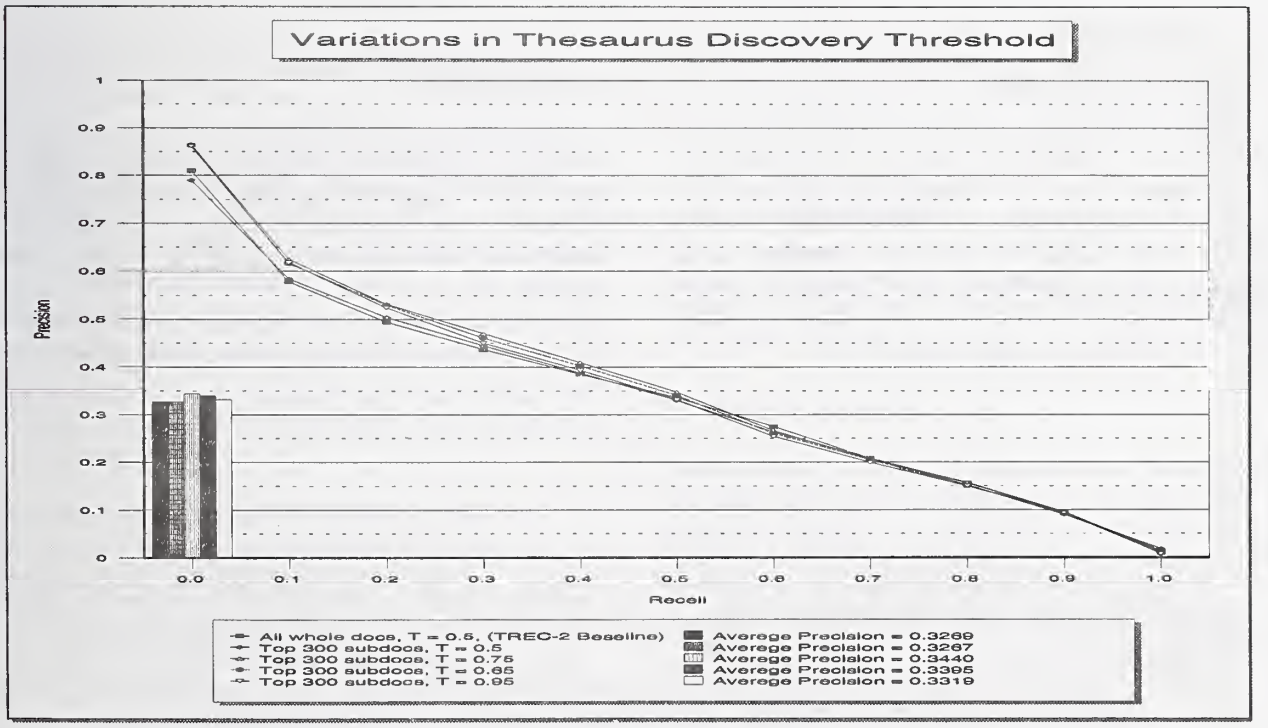


Figure 12: Effect of Thesaurus Size (Routing Topics)

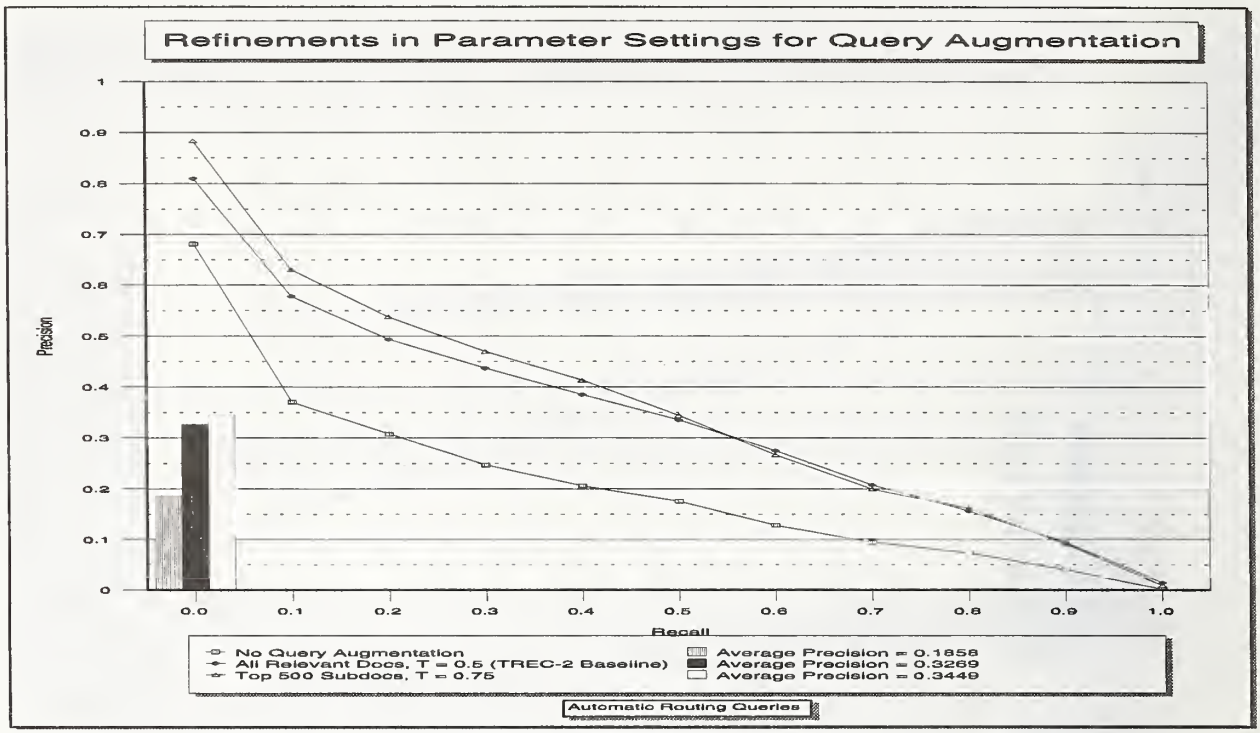


Figure 13: Optimized Sample Size and Thesaurus Threshold Results for Automatic Routing

7 Acknowledgements

CLARIT team participation in the TREC activities was made possible, in part, by grants from ARPA/NIST and CLARIT Corporation. We received valuable assistance from CLARIT-Project-team members Greg Grefenstette, Steve Handerson, and William R. Hersh and from CLARIT Corporation staff Armar A. Archbold, Michael McInerny, and Nataša Milić.

8 References

[Evans *et al.* 1993] Evans, David A., Lefferts, Robert G., Grefenstette, Gregory, Handerson, Steven K., Hersh, William R., and Archbold, Armar A., "CLARIT TREC Design, Experiments, and Results". In Donna Harman (Editor), *The First Text REtrieval Conference (TREC-1)*. NIST Special Publication 500-207. Washington, DC: U.S. Government Printing Office, 1993, 251–286; 494–501.

Bayesian Inference with Node Aggregation for Information Retrieval

Brendan Del Favero
Robert Fung
Institute for Decision Systems Research
350 Cambridge Avenue, Suite 380
Palo Alto, CA 94306
idsr@netcom.com

1 Introduction

Information retrieval can be viewed as an evidential reasoning problem. Given a representation of a document (e.g., the presence or absence of selected words and phrases), and a representation of an information need (e.g., topics of interest), the problem of information retrieval is to infer the degree to which the document matches the information need. Since probability theory is the classical choice for automating evidential reasoning, probabilistic approaches to information retrieval are natural and have had a long history, starting in the 1960's (Maron & Kuhns, 1960).

In this paper we describe research that adapts and applies Bayesian networks, a new technology for probabilistic representation and inference, to information retrieval. The technology has substantial advantages over older technologies including an intuitive representation and a set of efficient inference algorithms. We discuss the Bayesian network technology and probabilistic information retrieval in Section 2 of this paper.

Our research is directed at developing a probabilistic information retrieval architecture that:

- is oriented towards assisting users that have stable information needs in routing (i.e., sorting through) large amounts of time-sensitive material,
- gives users an intuitive language with which to specify their information needs,
- requires modest computational resources (i.e., memory and CPU speed), and
- can integrate relevance feedback and training data with users' judgements to incrementally improve retrieval performance.

Towards these goals, we have developed a system that allows a user to specify: multiple topics of interest (i.e., information needs), qualitative and quantitative relationships between the topics, document features that relate to the topics, and quantitative relationships between these features and the topics. The system runs on a Macintosh II computer and can use training data to estimate any of the quantitative values in the system. We discuss the particular methods we developed and used in our system in Section 3.

We participated in the exploratory group (Category B) of the 1993 Text Retrieval Conference (TREC-2), sponsored by the National Institute of Standards and Technology (NIST). As a participant in the exploratory group, we were tasked with working with a subset of the TREC-2 training and test data. Our training data consisted of *Wall Street Journal* (*WSJ*) articles and our test data consisted of *San Jose Mercury News* (*SJMN*) articles. We chose a subset of 10 topics out of the 50 TREC-2 routing topics to best illustrate the methods and concepts we developed. The choice of the 10 topics was reported to the TREC coordinators prior to our training runs and, of course, prior to our receipt of the test data. We generated routing queries for each of the 10 chosen topics, trained against the *WSJ* training set to improve our queries, and tested these queries against the *SJMN* articles in the test data set.

Our system was developed entirely within the duration of the TREC-2 project (January 93 to June 93) including the document handling, feature extraction, inference, and reporting capabilities. Our TREC-2 effort consisted of the two authors. We describe the experimental set-up in Section 4 and the result of our test run in Section 5.

We are very encouraged by the test results and have many ideas for future research, which we discuss in Section 6.

2 Background

In this section we describe the Bayesian network technology and outline the previous efforts in probabilistic information retrieval.

2.1 Bayesian Networks

While probability theory provides a suitable theoretical foundation for evidential reasoning, a technology based on probability theory that is computationally tractable and that includes an effective methodology for acquiring the needed probabilistic information has been lacking. Recent developments in Bayesian networks have provided these features. As the name suggests, the technology is based on a network representation of probabilistic information (Howard & Matheson, 1981; Pearl, 1988).

A Bayesian network represents beliefs and knowledge about a particular class of situations. The use of Bayesian

networks is similar to expert system technologies. Given a Bayesian network (i.e., a knowledge base) for a class of situations and evidence (i.e., facts) about a particular situation of that class, conclusions can be drawn about that situation. The technology has been used in a wide variety of situations, including medical diagnosis, military situation assessment, and machine vision.

A Bayesian network is a directed acyclic graph where each node represents a random variable (i.e., a set of mutually exclusive and collectively exhaustive propositions). Each set of arcs into a node represents a probabilistic dependence between the node and its predecessors (the nodes at the other end of arcs). The primary technical innovation of Bayesian networks is the representation, through the network's structure, of conditional independence relations between the variables in a network. This innovation not only provides an intuitive representation to acquire probabilistic information but also renders inference tractable for large numbers of real-world situations.

Inferences can be drawn from a Bayesian network with a wide variety of algorithms. There are exact algorithms (Lauritzen & Spiegelhalter, 1988; Shachter, 1986; Shachter, D'Ambrosio, & Del Favero, 1990) as well as approximate algorithms (Fung & Chang, 1989) Inference algorithms compute a probability distribution of some combination of variables in the network given all the evidence represented in the network.

Since the introduction of the Bayesian network technology, several efforts have been made to apply it to information retrieval (Fung, Crawford, Appelbaum, & Tong, 1990; Turtle & Croft, 1990). The results are promising. However, several recent innovations have been made in the Bayesian network technology that have not yet been applied to information retrieval. The innovations involve the representation of conditional independence relations that are finer than those currently represented at the network level. One of these innovations is Similarity Networks (Heckerman, 1991) developed by David Heckerman at Stanford University. Another innovation for representing the relationship between variables, the "co-occurrence diagram," was developed on this project.

2.3 Probabilistic Information Retrieval

Most probabilistic information retrieval techniques can be illustrated graphically through the use of Bayesian networks. In a simple formulation, there are n topics of interest ($t_1, \ldots, t_n$) and m identifiable document features ($f_1, \ldots, f_m$). The information retrieval problem is to compute the posterior probability $p(t_j | f_1, \ldots, f_m)$ for each topic, given a quantification (i.e., the joint probability $p(t_1, \ldots, t_n)$) of the frequency that the topics appear in the corpus and a quantification (i.e., the conditional probability $p(f_1, \ldots, f_m | t_1, \ldots, t_n)$) of the relationship of the "presence" of a topic in a document and the "presence" of features.

Figure 2.1 is a Bayesian network representing a retrieval model with one topic of interest and three features. The node t represents the events "the document is relevant to topic t" and "the document is not relevant to topic t." The nodes f_i represent the events "the feature f_i is present in the document" and "the feature f_i is not present (is absent) in the document." The prior probabilities of the events t and not-t are stored in the t node. The conditional probabilities $p(f_i | t)$ are stored in each of the feature nodes.

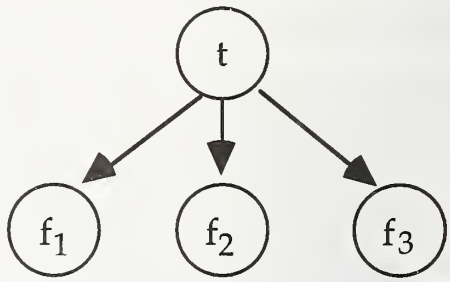


Figure 2.1: The two-level Bayesian network model of information retrieval

Because of the lack of arcs between the feature nodes, this diagram embodies the assumption, used in many probabilistic systems, that the features are conditionally independent of each other given the topic.

Using the Bayesian network form of Bayes' rule called arc reversal (Shachter, 1986), the posterior probability of the event t can be computed. The inversion formulas are straightforward and computationally feasible (they are described in the Appendix). Figure 2.2 shows the network in Figure 2.1 with the arcs reversed. It represents a model in which knowing whether one or more of the features are present provides information (i.e., the posterior distribution on node t) about whether the topic is relevant to the document.

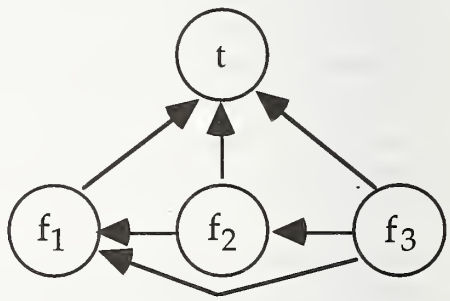


Figure 2.2: The Information retrieval model after Bayesian inversion

To address multiple topics within this framework, a model similar to the one represented by Figure 2.1 would be generated for each topic, and inference would be carried out on each topic separately. However, these multiple models

fail to represent possible relationships between the topics. As a consequence, the acquisition of consistent feature-topic probabilities and the interpretation and comparison of multiple topic probabilities are problematic. For example, it would be impossible using this framework to compute the probability that a document was relevant to two selected topics or to compute the probability that a document was relevant to at least one of two selected topics.

Bayesian networks can be used to explicitly represent the relationship between topics. For example, consider Figure 2.3 in which the two topics t_1 and t_2 are related.

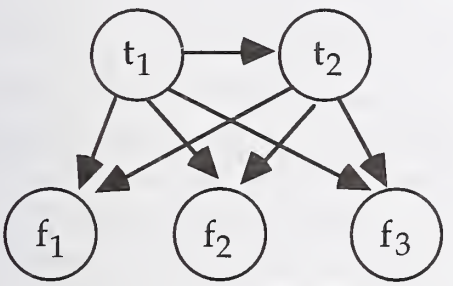


Figure 2.3: Information Retrieval Model with Two Related Topics

The Bayesian inference problem with multiple topics becomes more complicated than before. To compute the posterior probability of each topic, all the other topics must be removed through marginalization as well as reversing the topic-feature arcs as before. In addition, any joint distribution between the topics can be computed. To perform these computations, a general inference algorithm is needed.

However, there is a way to simply a multiple topic network to look like a single topic network (Figure 2.1). The topics can be combined into a compound topic (node s in Figure 2.4) (Chang & Fung, 1989) whose range is all possible present-or-absent combinations of t_1 and t_2. An advantage of this representation is that the same simple computational formulas can be used as before.

A disadvantage of this representation is that the intermediate query, s, must contain (in the worst case) 2^n states, representing all possible present-or-absent combinations of its n parent topics. However, this worst case is rarely seen, because the relationships between topics are typically sparse. Building this intermediate query is one of the innovations of our research and is described in Section 3.2.

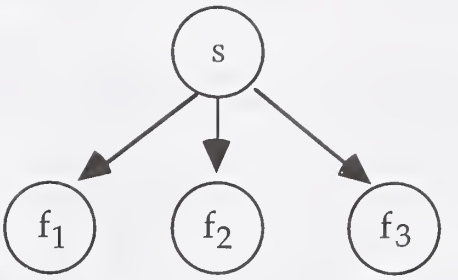


Figure 2.4: Multiple-topic query represented as a single compound-topic query

3 Retrieval Architecture

This section describes the information retrieval architecture we have developed.

The inputs to the system are the descriptions of topics of interest and a set of documents to test. The output of the system is list of documents ranked by their degree of relevance to each topic. The system computes the degree of relevance as the probability that a document is relevant to a topic, for every document-topic pair. To help in this task, the system is given a training set of documents to which relevance judgements have been attached.

There are several components to our retrieval system. The feature extraction component, described in Section 3.1, translates a document from its raw text form to a simpler internal representation that the system can use. The query construction component, described in Section 3.2, translates the description of a set of topics into an internal representation. The document scoring component, described in Section 3.3, uses Bayesian inference to calculate a measure of a document's relevance to a topic, given the internal representations of both document and topic.

3.1 Feature Selection and Extraction

Any observable characteristic of the text of a document that may be clearly defined (e.g., as either present or absent) may be regarded as a feature. Our system looks for two types of features in the text of a document: single words and pairs of adjacent single words. If a feature appears at least once in a document, it is counted as "present." Otherwise, it is counted as "absent."

The internal representation of a document is therefore a binary vector, each element indicating the presence or absence of the corresponding feature in the document.

The system removes many common suffixes from a word using Paice's stemming rules (Paice, 1977). This means, for example, that if either of the words "walking" or

"walks" is present in a document, the system considers the root word "walk" to be present in the document.

The system must be given a list of features for which to look. This target list was constructed in three steps. First, a list of candidate features was generated from the descriptions of the 50 TREC-2 routing topics. The text of the descriptions was broken up into individual words, from which the suffixes (if any) were removed. Duplicate words and common words (stop words) were removed. A number of two-word features (such as phrases and proper names) were identified by hand. This procedure created a list of about 1400 candidate features.

Second, the system extracted the relative frequency information for each of these features from the training documents. Then, for each topic (the 10 plus an additional topic representing "none of the above,") the system sorted the candidate features in descending order according to the F4 formula of Robertson and Sparck Jones (Robertson & Sparck Jones, 1976), which we used is a measure of the ability of a feature to characterize a topic.

Third, the top 30 features for each topic (and the top 60 features for "none of the above") were combined into a single list. After removing duplicates, this yielded the final list of 229 features.

We tried numerous feature selection strategies and settled on this one as the most satisfactory.

3.2 Query Representation and Construction

In preparation for the inference step, the 10 single-topic queries are combined into one multiple-topic query. As mentioned in Section 2.3, this aggregation can in the worst case require 2^{10} different states in the multiple-topic query. In this section, we describe how to reduce the number of states required by considering the relationships between the topics.

Once the states of the query have been structured, we must assign numerical values to the Bayesian model. We describe how to generate estimates of the prior probabilities of each of these states in Section 3.2.3. We describe in Section 3.2.4 how to define, for each feature and state, the conditional probability that the feature appears in a document given that the document is relevant to that state.

3.2.1 Pairwise Relationships Between Topics

There are six possible pairwise relationships between any two topics t_1 and t_2:

1. t_1 and t_2 are *mutually exclusive*: if there is no document relevant to both topics
2. t_1 is a *subset* of t_2 if all documents relevant to t_1 are also relevant to t_2
3. t_2 is a *subset* of t_1 if all documents relevant to t_2 are also relevant to t_1
4. t_1 and t_2 are *equivalent* if t_1 and t_2 are subsets of each other (they satisfy Relations 2 and 3)
5. t_1 and t_2 are *dependent* if knowing that a document is relevant to t_1 gives you some information about whether the document is relevant to t_2
6. t_1 and t_2 are *independent* if knowing that a document is relevant to t_1 gives you no information on whether the document is relevant to t_2

Each of the Relations 1-5 is a type of dependence between topics. In a belief network, topics satisfying Relations 1-5 would be connected by an arc. To ensure that two topics satisfy only one of the relations, Relation 5 (dependence) is defined as any type of dependence that is different from those of Relations 1-4.

Relation 6, independence, is represented in a belief network by the absence of an arc between the topics.

The distinction between dependence (Relation 5) and independence (Relation 6) is useful in calculating the probabilities of combinations of the topics, as described in Section 3.2.3.

Relations 1-4 can be identified by testing whether the defining conditions are satisfied in the training set. If the topics satisfy none of the first four relations, then a chi-square test can distinguish between Relations 5 and 6. If there are too few documents with relevance judgements for both topics to make reliable conclusions (our cutoff was 13 documents), the system makes an assessment, then prompts the user to verify it manually.

To fully explore the pairwise relationships between topics, we selected ten of the fifty TREC-2 routing topics to use in our models and tests. The topics are listed in Table 3.1.

Topic Number	Topic Description
57	Financial Health of MCI
61	Israeli Role in the Iran-Contra Affair
74	Conflicting Policies of the US Government
85	Official Corruption in any government
88	Crude Oil Price Trends
89	Downstream investing by OPEC members
90	Data on Proven Reserves of Oil and Gas
97	Fiber Optic Applications
98	Fiber Optic Equipment Manufacturers
99	Iran-Contra Affair

Table 3.1: List of the 10 topics we selected

The ten topics were selected to provide an interesting mixture of relationships among the topics. Before looking at the relevance information in the training set, we assessed these relationships manually, as shown in Table 3.2. The table is symmetrical about the main diagonal.

	57	61	74	85	88	89	90	97	98	99
57	e		i	i				d	d	
61		e	d	s						sd
74	i	d	e	d	i	i		i	i	sd
85	i	s	d	e	i	i		i	i	sd
88			i	i	e	d	d			i
89			i	i	d	e	d			i
90					d	d	e			
97	d		i	i				e	d	
98	d		i	i				d	e	
99		sd	sd	sd	i	i				e

d = dependent
s = subset
sd = subset or dependent
i = independent
e = equivalent
empty = mutually exclusive

Table 3.2: Pairwise relationships between 10 TREC topics, assessed manually before looking at the training data

Table 3.3 shows the topic relationships that were generated automatically from the relevance judgements on the training documents. Manual verification of the system's assessment was required in about half of the cases (marked with an asterisk). The table is symmetrical about the main diagonal.

	57	61	74	85	88	89	90	97	98	99
57	e	*	d					d	d	*
61	*	e	d	s	*		*	*	*	d
74	d	d	e	d				*		d
85		s	d	e	*		*	*	d*	d
88		*		*	e	i	i	*	*	*
89					i	e	i	*		*
90		*		*	i	i	e	*		*
97	d	*	*	*	*	*	*	e	d	*
98	d	*		d*	*			d	e	*
99	*	d	d	d	*	*	*	*	*	e

* = manual intervention required

Table 3.3: Pairwise relationships determined from the data

The system's assessments match the manual assessments quite well. The system found even more mutually exclusive pairs than we had intuitively thought. One surprise is between topics 61 ("Israeli role in the Iran-Contra Affair") and topic 91 ("Iran Contra Affair"). One might assume that 61 is a subset 99, but there are indeed documents in the training set that are relevant to 61 but not to 99. Thus, the relation between 61 and 99 is dependence rather subset.

The information in Table 3.3 can be expressed as a directed graph, as shown in Figure 3.1. Each topic is a node. Two topics are connected by an arc if there is at least one document containing them both. There is no arc between mutually exclusive topics. The arcs marked "i" connect independent topics, the unmarked arcs connect topics that are dependent, and the directed arc points to a subset from its superset.

Figure 3.1 is not a belief network. It may be considered a "co-occurrence diagram," since topics that are relevant together (that co-occur) in the collection are connected.

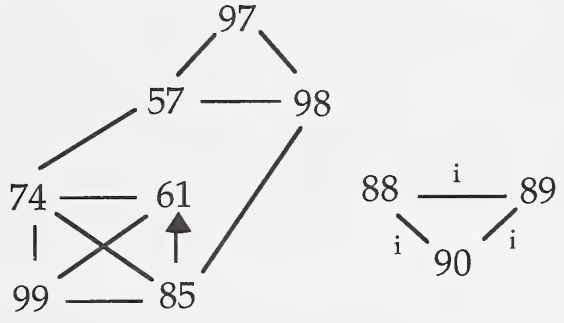


Figure 3.1: Relationships diagram among 10 TREC topics

3.2.2 Generating states from the topics

The states of the multiple-topic query can be generated automatically by enumerating all possible "relevant or not relevant" combinations of topics, subject to two rules:

1. Two mutually exclusive topics cannot both be relevant within the same state.

2. A topic that is a subset of another cannot be relevant within a state unless its superset topic is also relevant.

By construction, there is always one state, U^-, to which a document is relevant if it is relevant to none of the topics.

For the ten topics, these rules reduce the number of states drastically from the theoretical maximum of $1024 = 2^{10}$ states to the actual number of 28 states.

A state is identified by listing the topics to which a document is relevant (or not relevant) if it is relevant to that state. For instance, documents relevant to the state (61 -74 85 -99) are relevant to topics 61 and 85 and are not relevant to topics 74 and 99. The list of states appears in the first column of Table 3.4.

3.2.3 Generating State Priors

In theory, the prior probabilities of the states can be calculated from their relative frequencies in the training set. However, since there are very few documents in the TREC-2 training set that were evaluated for three or more topics, a different way to estimate the priors was required.

One estimate is provided by factoring the states prior into products of the priors of smaller compound topics. This can be accomplished without manual intervention. For instance, for the independent topics 88, 89, and 90, only three numbers are needed (the prior probabilities of the three topics in the training set) to compute the probabilities of all seven states containing the three topics. For the state (61 -74 85 -99), two numbers are needed: the prior probability of the compound topic (-74 85 -99) and the probability that topic 61 is relevant given that topic 85 is relevant. The priors obtained by this method are shown in the second column of Table 3.4. They are expressed as inverse frequencies: the number given is the number of weeks (on average) between articles relevant to a state, assuming 1000 documents per week.

State	Average Weeks between Articles, Assessed Automatically	Average Weeks between Articles, Assessed Manually
-88 -89 90	< 1	3
-88 89 -90	< 1	3
-88 89 90	20	7
88 -89 -90	< 1	2
88 -89 90	20	4
88 89 -90	40	4
88 89 90	4000	9
57 97 98	10	8
57 97 -98	< 1	5
57 -97 98	< 1	6
57 -97 -98	< 1	3
-57 97 98	1	6
-57 97 -98	< 1	4
-57 -97 98	< 1	3
61 74 85 99	33	20
-61 74 85 99	14	3
61 74 85 -99	< 1	50
-61 74 85 -99	< 1	3
61 -74 85 99	< 1	20
-61 -74 85 99	< 1	5
61 -74 85 -99	5	100
-61 -74 85 -99	2	2
74 -85 99	< 1	3
74 -85 -99	< 1	2
-74 -85 99	< 1	3
57 74	< 1	20
85 98	1	50
- U		

Table 3.4: The states and their prior probabilities expressed as frequencies (assuming 1000 articles per week) generated by two assessment methods

However, these priors proved unsatisfactory and required manual override, for three reasons. First, the training set is a very biased sample of the *WSJ*. The training documents were selected by the retrieval systems of TREC-1 to be intentionally relevant to at least one of the topics. Thus, the prior derived from the training set tends to be a gross overestimate of the true prior.

Second, the time period of the training set (1987 to 1992) is different from that of the test set (only 1991). The frequency of the topics relative to each other changes over time. For instance, there are many fewer articles on the Iran-Contra affair (relative to the other topics) in 1991 than there were in 1987-1990. We adjusted the priors to match the relative frequencies expected in the test set.

Third, the priors are widely divergent from the intuition of any user who reads a newspaper. The numbers suggest, for instance, that about 1 out of every 100 articles in the *WSJ* (and, by analogy, in the *SJMN*) are relevant to topic 57, "MCI." Any reader of the *WSJ* or the *SJMN* knows that this estimate is much too high – the relative frequency is probably closer to 1 in 500 or 1 in 1000 than to 1 in 100.

Thus, the prior probabilities were assessed manually. It is assumed that the user has some knowledge of the test domain (in this case, articles in the *SJMN*) and can with some thought assess the relative frequencies of various states as a part of specifying the query for the particular routing request. For each state, we ask the user what is the average number of weeks between the publication of articles relevant to that state. This number is presented in the third column of Table 3.4. It can be converted to a prior probability by combininbg it with the assumption that there are 1000 documets per week.

The prior probability of $\bar{U}$, $p(\bar{U})$, is calculated as one minus the sum of the priors of all the other states, to ensure that the probability of all states together is one.

3.2.4 Feature Conditional Probabilities

The inference algorithm requires, for each feature and for each state, the conditional probability of the feature given the state. These probabilities cannot be obtained directly from the relative frequencies obtained from the training set, because there are few documents that are relevant to more than one topic at a time.

We approximate these probabilities by using a structure called a noisy-or gate.

The noisy-or gate combines the effects of two or more factors, each of which may contribute to the presence of a feature. It is a model of disjunctive interaction, as described in (Pearl, 1988). It has been used in medical decision research to calculate the probability of a particular symptom being present, given diseases that cause the symptom (Heckerman, 1989).

In the context of information retrieval, a feature may be present due to any of the topics that are relevant in the state. For each state-feature pair, we build a noisy-or model. The contributing factors are the topics that are relevant within the state. The effect is the feature's presence or absence in the document.

For example, consider a feature f and the state (57 -97 98). The feature may be present due to topics 57 or 98. It cannot be present due to topic 97 because that topic is not relevant within the state. Let E1 be the event that the feature is present due to topic 57, and let E2 be the event that the feature is present due to topic 98. Table 3.5 lists all the possible cases of the two uncertain events. Figure 3.4 shows the belief network structure of the noisy-or model. The node with the double wall is a deterministic logical or gate.

Feature Present due to 57 (E1)	Feature Present due to 98 (E2)	Feature Present at all (E1 OR E2)
Yes	Yes	Yes
Yes	No	Yes
No	Yes	Yes
No	No	No

Table 3.5: Possible cases for a noisy-or node

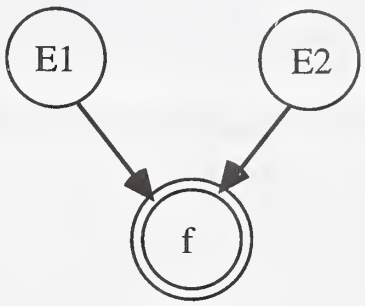


Figure 3.4: Belief Network corresponding to a Noisy-Or Gate Model

The only case in Table 3.5 in which the feature is absent is the fourth case. Thus, the conditional probability that the feature is absent, given this state, is the probability of that fourth case. The probability that the feature is present is one minus the probability that the feature is absent.

3.3 Document Scoring

The Bayesian inversion described in Section 2.1 yields, for each document, the posterior probability that the document is relevant to each *state*. We calculate the posterior probability that the document is relevant to each *topic* by summing the posterior probabilities of all of the states in which the topic appears.

For example, the posterior probability for topic 57 is the sum of the posterior probabilities of the five states in which topic 57 is relevant (refer to Table 3.4). The states are (57 97 98), (57 97 -98), (57 -97 98), (57 -97 -98 -74) and (57 74).

The final list of documents for each topic contains the top 1000 documents, ranked in descending order according to the posterior probability that they are relevant to that topic.

4 Experimental Environment

4.1 Hardware and Software Environment

The system was developed by one programmer over the period of six months. It was written in C using the Symantec Think C 5.0 compiler under Macintosh System 7.0.1. The experiments were run on two machines, depending on availability: a Macintosh IIfx and a Centris 650. Both machines had 8 Mb of memory and a 500 Mb hard disk.

4.2 Summary of System Data Structures

Table 4.1 is a list of the data structures used by the system. Each is listed with the sections of this paper in which it is described and with an indicator of whether it was provided by NIST (N), assessed manually by us (M), or generated automatically by the system (A).

Data Structure, (Sections Source)
Descriptions of Routing Topics (3.1, N)
Training Documents (3.1, N)
Candidate Feature List (3.1, A/M)
Final Feature List (3.1, 4.3.2, A)
Relevance Judgements on the Training Documents (3.2.1, N)
Topic Relationships (3.2.1, A/M)
State Prior Probabilities (3.2.2, 4.3.2, M)
Feature Conditional Probabilities (3.2.4, A)
Test Documents (3.3, N)
Document Posterior Probabilities (3.3, A)
Final list of retrieved documents (3.3, A)

Table 4.1: Data Structures used by the System

The candidate feature list could be defined entirely automatically by sufficiently intelligent procedures such as phrase and proper name identification. The state prior probabilities are the only data items that must be assessed manually.

The data files needed by the system, other than those provided by NIST, occupy about 100 Kb on disk.

4.3 Training Phase

4.3.1 Building the Data Structures

We are a Category B participant in TREC-2. As such, we used a subset (just the *WSJ* articles) of the full training collection. Of these we used only those (roughly) 29,000 articles for which there is some relevance judgment available. In addition, because we considered only ten of the TREC-2 routing topics, we used only the 5941 documents that have a relevance judgment on at least one of the ten topics.

The inputs to a training run are the candidate feature list, the list of topic relationships, the training documents, and the training relevance judgements. The outputs of a training run are the final list of features and the feature conditional probabilities.

The time to complete a training run is about 1.5 hours.

4.3.2 Defining the Query

The design of the TREC-2 experiment required that before we receive the test data, we submit the definitions of our system and of our particular query to NIST. The query definition includes the final feature list, the list of topic relationships, and the state prior probabilities. All of the other data is determined automatically, as shown in Table 4.1.

We ran about two dozen sample experiments that applied the system to the training documents to gauge its performance with different query definitions. In these tests, we varied only two of the data inputs, the state prior probabilities and the final feature list. The list of topic relationships remained constant throughout these tests.

It took much longer than expected to finish programming the system. Thus, all of the experiments to define the query were completed in the two weeks immediately preceding the final submission of our query definition to NIST.

4.4 Test Phase

As a Category B participant in TREC-2, we ran our routing experiment on just the *SJMN* articles.

The inputs to the test runs were the final list of features, the feature conditional probabilities, the list of topic relationships, and the test documents. The output of the test run is the final list of retrieved documents.

A test run takes about 5.5 hours. This includes decompressing the 150 MB of test data, one file at a time, when it is needed.

5 Results

The TREC-2 designation for our system is "idsra2." Figure 5.1 shows the precision-recall curves for the ten topics we considered, excluding topic 88, for which no TREC-2 participant found any relevant documents.

Table 5.1 shows several measures of performance for our results, along with the best, median, and worst values for those measures among all TREC-2 participants. We again exclude topic 88.

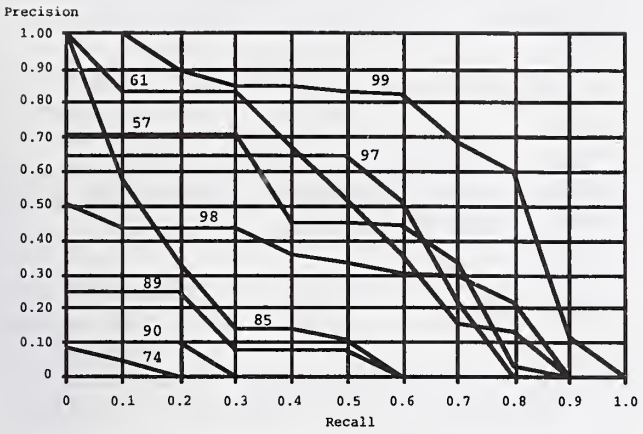


Figure 5.1: Precision vs. Recall for system idsra2

Topic	Relevant Retrieved at 100			
Number	idsra2	Best	Median	Worst
57	17	18	17	0
61	19	19	9	0
74	6	16	6	0
85	33	54	33	1
89	2	3	2	0
90	1	1	0	0
97	25	28	18	1
98	24	24	17	0
99	60	60	52	0

Table 5.1a: Relevant documents in the top 100 retrieved, for idsra2 and for all systems

Topic	Relevant Retrieved at 1000			
Number	idsra2	Best	Median	Worst
57	18	19	18	0
61	24	25	24	0
74	11	31	11	1
85	88	115	88	2
89	2	4	2	0
90	1	3	0	0
97	27	32	27	1
98	26	29	26	0
99	70	70	66	0

Table 5.1b: Relevant documents in the top 1000 retrieved, for idsra2 and for all systems

Topic	Average Precision			
Number	idsra2	Best	Median	Worst
57	0.387	0.460	0.374	0.000
61	0.464	0.464	0.083	0.000
74	0.008	0.074	0.008	0.000
85	0.174	0.353	0.174	0.000
89	0.081	0.259	0.077	0.000
90	0.025	0.025	0.000	0.000
97	0.383	0.383	0.202	0.002
98	0.282	0.427	0.334	0.000
99	0.700	0.700	0.509	0.000

Table 5.1c: Average precision (as defined for TREC-2), for idsra2 and for all systems

6 Conclusions and Future Directions

We believe that we have made significant progress to developing an information retrieval architecture that:

- is oriented towards assisting users with stable information needs in routing large amounts of time-sensitive material
- gives users an intuitive language with which to specify their information needs
- requires modest computational resources, and
- can integrate relevance feedback and training data with users' judgements to incrementally improve retrieval performance.

We are encouraged by the test results. We have not had very much time to analyze the results but we intend to try to understand why we did very well on some topics and not so well on others. Very preliminary analysis suggests that the features for the topics in which we did well (e.g., 61 and 99) were much more informative than the ones on which we did very poorly (e.g., 74).

We have many ideas for future research. These ideas fall into three basic categories: probabilistic representation, user interface, and inference methods.

The most important improvements we would like to make are in the category of probabilistic representation of the topic and the document. One research goal is to develop a way to intuitively represent relationships between features. Also, we would like to explore more sophisticated feature extractors that recognize phrases, synonyms, and features derived from natural language processing. We believe that achievement of these goals could lead to significant improvements in performance.

In conjunction with these representational improvements, we would like to design a complete user interface that would allow users to make the needed probabilistic judgements easily and intuitively. We would also like to develop an explanation facility that could describe why one document was preferred over another.

There are new exact and inexact algorithms that could handle these representational modifications. We would like to experiment with these algorithms to see if they are suitable. We would also like to implement a relevance feedback mechanism based on the Bayesian concept of equivalent sample size.

Finally, while the information retrieval problem has been viewed primarily (and rightly so) as an evidential reasoning problem, we take the position that a decision-theoretic perspective is more accurate since information retrieval is a decision process. We believe that this perspective can provide additional insight and eventually improve upon the probabilistic approaches that have been developed.

Acknowledgment

We would like the thank Dr. Richard Tong for his assistance and guidance.

Appendix Bayesian Inference

We calculate the posterior probability that a document is relevant to a state using Bayes' rule and an assumption of conditional independence between the features.

The Bayesian inversion formula is this:

$$p(s \mid \text{document}) = p(s \mid F) = \frac{p(F \mid s)\, p(s)}{p(F)} ,$$

where s is the state, and F is a binary feature vector with

$F_i = 1$ if feature i is present (the event f_i^+)

$F_i = 0$ if feature i is absent (the event f_i^-).

The set of all features that are present is denoted F^+. All other features are absent and are in the set denoted F^-.

The first term in the numerator is expanded under the assumption that the features are conditionally independent given the state.

$$p(F \mid s) = \prod_{i \text{ in } F^+} p(f_i^+ \mid s) \prod_{i \text{ in } F^-} p(f_i^- \mid s)$$

The second term in the numerator is the state prior, p(s), which can be estimated as described in Section 3.2.

The denominator is a normalization constant obtained by summing over the values for the numerator for all the states.

$$p(F) = \sum_{\text{all } s} p(F \mid s)\, p(s)$$

p(F) is the probability that one would observe the particular set of features F.

References

Chang, K. C., & Fung, R. M. (1989). *Node Aggregation for Distributed Inference in Bayesian Networks*. In *IJCAI 89*. Detroit: Morgan Kaufmann.

Fung, R. M., & Chang, K. C. (1989). Weighing and Integrating Evidence for Stochastic Simulation in Bayesian Networks. In R. D. S. M. Henrion L.N. Kanal and J.F. Lemmer (Eds.), *Uncertainty and Artificial Intelligence 5* Elsevier Science Publishers B.V. (North-Holland).

Fung, R. M., Crawford, S. L., Appelbaum, L., & Tong, R. M. (1990). A Probabilistic Concept-based Architecture for Information Retrieval. In *Proceedings of the ACM International Conference on Information Retrieval* Brussels, Belgium.

Heckerman, D. E. (1989). A tractable algorithm for diagnosing multiple diseases. In *Proceedings of the Fifth Workshop on Uncertainty in Artificial Intelligence*, (pp. 162-173). Detroit, MI.

Heckerman, D. E. (1991). *Probabilistic Similarity Networks*. The MIT Press.

Howard, R. A., & Matheson, J. E. (1981). Influence diagrams. In R. A. Howard & J. E. Matheson (Eds.), *Readings on the Principles and Applications of Decision Analysis* (pp. 721-762). Menlo Park, Ca.: Strategic Decisions Group.

Lauritzen, S. L., & Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *JRSS, 50*, 157-224.

Maron, M. E., & Kuhns, J. L. (1960). On relevance, probabilistic indexing and information retrieval. *Journal of the ACM, 7*, 216-244.

Paice, C. D. (1977). *Information Retrieval and the Computer*. London.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. San Mateo, Ca.: Morgan Kaufman.

Robertson, S. E., & Sparck Jones, K. (1976). Relevance Weighting of Search Terms. *Journal of the American Society for Information Science*(May-June), 129-146.

Shachter, R. D. (1986). Evaluating influence diagrams. In*Operations Research* (pp. 871-882).

Shachter, R. D., D'Ambrosio, B., & Del Favero, B. (1990). Symbolic Probabilistic Inference in Belief Networks. In *AAAI-90*, (pp. 126-131). Boston: Morgan Kaufmann.

Turtle, H., & Croft, W. B. (1990). Inference networks for document retrieval. In J. L. Vidick (Ed.), *ACM SIG IR*, (pp. 1-24). Brussels, Belgium.

Effective and Efficient Retrieval from Large and Dynamic Document Collections

Daniel Knaus, Peter Schäuble
(knaus|schauble)@inf.ethz.ch
Swiss Federal Institute of Technology (ETH)
CH-8092 Zürich, Switzerland

Abstract

A new retrieval method together with a new access structure is presented that is aimed at a high update efficiency, a high retrieval efficiency and a high retrieval effectiveness. The access structure consists of signatures and non-inverted descriptions. This access structure can be updated efficiently because the description of a single document is stored in a compact form. The signatures are used to compute approximate retrieval status values first, and the non-inverted descriptions are then used to determine the final list of documents ranked by the exact retrieval status values. Our basic approach based on the standard $tf * idf$ weighting scheme has been improved in in both retrieval effectiveness and retrieval efficiency. On an average, the time for retrieving the top ranked document is clearly below two seconds while the document collection can be updated in 10 msec. (inserting, deleting, or modifying a document description).

1 Introduction

Information retrieval systems are more and more used to retrieve information from *large and dynamic* data collections, e.g. in office automation or in newscasting companies. Since in such environments, the data collections may have to be updated many times every second, the update efficiency is an important evaluation criterion that should be taken into account in addition to the traditional evaluation criteria (retrieval effectiveness, retrieval efficiency, etc.) [10, pp. 161], [12, pp. 144].

Before introducing our approach, we show that the update efficiency is conflicting with the retrieval effectiveness and with the retrieval efficiency. In the case of static data collections, the retrieval effectiveness criterion is usually met by means of *weighted retrieval* including relevance feedback [4], [9] whereas the retrieval efficiency criterion is met by precomputing the document descriptions and by organizing these descriptions as an inverted file [5]. In the case of dynamic data collections, however, a transaction updating the inverted file may block other retrieval and update transactions for an unacceptable long time because adding the description of a new document to an inverted file is time consuming, particularly if the document is long. A possible remedy is to use signatures instead of an inverted file [2]. Signatures can be updated efficiently; however, they do not allow document feature weighting and hence, the retrieval effectiveness of the signature based method is inferior to a fully weighted retrieval method [8]. Thus, it is difficult to achieve simultaneously high retrieval effectiveness, high retrieval efficiency, and high update efficiency.

In our approach, we focus on the retrieval from large and dynamic data collections where we face the problem of achieving simultaneously high retrieval effectiveness, high retrieval efficiency, and high update efficiency. Addressing this problem is justified by the need for appropriate retrieval capabilities in dynamic environments such as in office automation or in newscasting companies. Within the SPIDER project [11] we have developed a retrieval method and an access structure which supports fully weighted retrieval and which achieves short response times even when the collection is updated several times every second. In Section 2, we describe our basic approach, in Section 3, we focus on refinements of this basic approach, in Section 4, we present the results, and in Section 5, we draw some conclusions.

2 SPIDER's Query Evaluation Algorithm

In this section, we present a retrieval method together with a new access structure facilitating both fast weighted retrieval and efficient updates of the access structure. A preliminary version was presented in [11]. Our access structure consists of *signatures and non-inverted descriptions* of the documents. The signatures are used to compute approximate retrieval status values $RSV^0(q, d_j)$ first and the non-inverted document descriptions are then used to determine the exact retrieval status values $RSV(q, d_j)$. The trick is that the approx-

imate retrieval status values provide fairly tight *upper bounds* for the exact retrieval status values. We will see how we can take advantage of these upper bounds such that only a few exact retrieval status values have to be computed.

The retrieval method determining the approximate retrieval status values and the retrieval method determining the exact retrieval status values are given by the functions RSV^0 and RSV respectively. Let

$$\Phi \quad := \quad \{\varphi_0, \ldots, \varphi_{m-1}\}$$

be the indexing vocabulary (e.g. a set of terms) and let

$$D \quad := \quad \{d_0, \ldots, d_{n-1}\}$$

be the current document collection. We assume that the signatures $\sigma(d_j)$ consist of w bits where the bit at position p has the value $\sigma(d_j)[p]$.

$$\sigma(d_j) \quad = \quad (\sigma(d_j)[0], \ldots, \sigma(d_j)[w-1])$$

Every indexing feature φ_i is assigned a bit position $p = h(\varphi_i)$ by means of a hash function $h(\varphi)$.

$$h : \Phi \rightarrow \{0, \ldots, w-1\}$$

The function h specifies a signature $\sigma(d_j)$ for every document d_j by setting the bit at position p iff d_j contains a feature φ_i which is hashed to this position.

$$\sigma(d_j)[p] \quad := \quad \begin{cases} 1 & \text{if } \exists \varphi_i \in d_j : h(\varphi_i) = p \\ 0 & \text{otherwise} \end{cases}$$

We now define the approximate retrieval status value $RSV^0(q, d_j)$ and the exact retrieval status value $RSV(q, d_j)$ as follows.

$$RSV^0(q, d_j) := \frac{1}{|\vec{d_j}|} * \sum_{\varphi_i \in q : \sigma(d_j)[h(\varphi_i)]=1} a^0_{ij} * b_i \qquad (1)$$

$$RSV(q, d_j) := \frac{1}{|\vec{d_j}|} * \sum_{\varphi_i \in q : \varphi_i \in d_j} a_{ij} * b_i \qquad (2)$$

where a^0_{ij} denote the approximate weights of document features, a_{ij} denote the exact document weights, and b_i denote the query weights. The size $|\vec{d_j}|$ of the description vector $\vec{d_j}$ is defined as usual (see Table 1).

Our basic approach consists of a $tf * idf$ weighting scheme, i.e. the query features are "ntn" weighted whereas the document features are "ntc" weighted (see Table 1).

The query weights b_i depend on the feature frequencies $ff(\varphi_i, q)$ and on the normalized inverse document frequencies $nidf(\varphi_i)$. The exact document weights a_{ij} depend on the feature frequencies $ff(\varphi_i, d_j)$ and on the $nidf(\varphi_i)$. In addition, they (a_{ij}) are cosine normalized, which is expressed by the division by $|\vec{d_j}|$ in (1) and (2). The $nidf(\varphi_i)$'s are determined by the document frequencies $df(\varphi_i)$. The normalizations of the feature frequency component and the normalizations of the document frequency component do not affect the retrieval status values because they cancel out when using the cosine measure.

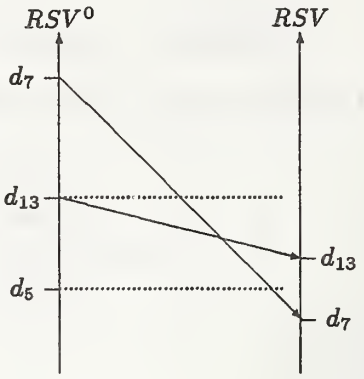


Figure 1: Computation of the exact retrieval status values in decreasing order of the approximate retrieval status values.

Because of these normalizations, the approximate document weights a^0_{ij} are always upper bounds of the exact document weights a_{ij}, but they do not depend on the feature frequencies $ff(\varphi_i, d_j)$.

$$0 \leq a_{ij} \leq a^0_{ij}$$
$$0 \leq b_i$$

It is easy to show that $RSV(q, d_j) = 0$ if $\sigma(q) \wedge \sigma(d_j) = 0$. Furthermore, from the fact that $\varphi_i \in d_j$ implies $\sigma(d_j)[h(\varphi_i)] = 1$ and from $a_{ij} \leq a^0_{ij}$ follows

$$RSV(q, d_j) \quad \leq \quad RSV^0(q, d_j) \qquad (3)$$

In what follows, we describe the evaluation algorithm by means of an example. Let q be the user's query. In a first step, bitwise AND-operations are performed with the query signature $\sigma(q)$ and the signature $\sigma(d_j)$ of every document $d_j \in D$. If $\sigma(q) \wedge \sigma(d_j)$ is non-zero then the approximate retrieval status value $RSV^0(q, d_j)$ is computed. The first step is finished by ordering the documents in decreasing order of the approximate retrieval status values. Remember that $\sigma(q) \wedge \sigma(d_j) = 0$ implies that $RSV(q, d_j) = 0$ and hence, d_j can be ignored in the sequel.

In the second step, the exact retrieval status values are computed in the order that was previously determined by means of the approximate retrieval status values. As shown in the following example, only as many

exact retrieval status values are computed as necessary. Assume that d_7 is the first (i.e. d_7 has the highest RSV^0), d_{13} is the second, and d_5 is the third document as shown in Figure 1. In this case, $RSV(q, d_7)$ is computed first and next, $RSV(q, d_{13})$ is computed. According to Figure 1, $RSV(q, d_{13})$ is the highest exact retrieval status value of those two exact retrieval status values that are known at this moment. Furthermore, all unknown exact retrieval status values are less than or equal to $RSV^0(q, d_5)$ because of (3). Since $RSV(q, d_{13}) > RSV^0(q, d_5)$, we can conclude that d_{13} has the highest exact retrieval status value of *all* documents. In this example, only two exact retrieval status values had to be computed to determine the top ranked document. In practice, more exact retrieval status values have to be computed.

The evaluation algorithm is based on an *access structure* that specifies the following functions.

$$\sigma : d_j \mapsto \sigma(d_j) \tag{4}$$

$$\pi : d_j \mapsto \{(\varphi_i, ff(\varphi_i, d_j)) : \varphi_i \in d_j\} \tag{5}$$

$$nidf : \varphi_i \mapsto nidf(\varphi_i) \tag{6}$$

$$size : d_j \mapsto |\vec{d_j}| \tag{7}$$

The functions σ and π determine the signatures and the non-inverted document descriptions respectively. The functions *nidf* and *size* provide scaling and normalization factors. The function values $\sigma(d_j)$ and $\pi(d_j)$ are determined completely by the document d_j itself. The function values $nidf(\varphi_i)$ and $size(d_j)$, however, depend on the entire document collection. Fortunately, the variation of $nidf(\varphi_i)$ is small if the data collection is sufficiently large. Thus, $nidf(\varphi_i)$ has to be recomputed only if the domain of the data collection is shifting and $size(d_j)$ has to be recomputed if either *nidf* was recalculated or d_j was modified. In the next section, we will see how this basic query evaluation scheme can be refined.

3 Refinements

The basic evaluation algorithm described in Section 2 achieves an excellent update efficiency because the description of a single document is stored in a compact form which can be updated efficiently. However, the retrieval effectiveness as well as the retrieval efficiency need further refinements in the case of the TREC experiment where extremely large queries are matched against a large document collection containing some large documents.

To *improve the retrieval effectiveness*, we adapted the basic evaluation algorithm to the best global weighting scheme achieved by the SMART system at TREC-1 [1]. The query features are "ltn" weighted, i.e. the query weights depend on the logarithm of the feature frequencies and on the inverse document frequencies, and they are not cosine normalized. The "lnc" weighted document features depend on the logarithm of the feature frequencies. They are cosine normalized, but they do not have a document frequency component. The detailed definitions of the feature weights are given in Table 1.

In order to *achieve a good retrieval efficiency* for the TREC collection we have reduced the indexing vocabulary in such a way that on one hand the retrieval efficiency is improved and on the other hand, the retrieval effectiveness is not deteriorated very much. Our indexing vocabulary was not only reduced by applying a stop word list and a word reduction algorithm, but also by eliminating further indexing terms. From another project [3] and from the experiences of the SMART system at TREC-1 [1] we knew that the removal of the indexing features having a very high document frequency does not deteriorate the retrieval effectiveness very much. The reasons why this restriction has a small effect on the retrieval effectiveness is the following. The elimination of the high document frequency features changes the retrieval status values very little because these features have a small inverse document frequency and hence, they contribute little to the retrieval status values. Such an elimination of the high document frequency features is similar to the application of a collection dependent stop word list in addition to a general stop word list. Thus, this restriction has only a small effect on the retrieval status values.

In what follows, we focus on the retrieval efficiency and why it is improved by this restriction. With a reduced indexing vocabulary,

1. fewer documents have a positive approximate retrieval status value,

2. fewer features φ_i contribute an error $(a^0_{ij} - a_{ij}) * b_i$ to the approximate retrieval status values, and

3. fewer false matches (caused by collisions) occur when computing the approximate retrieval status values.

Both scanning the signatures and sorting the approximate retrieval status values become faster when fewer approximate retrieval status values are positive. Furthermore, the approximate retrieval status values become tighter upper bounds when fewer false matches occur and when the sum of the errors $(a^0_{ij} - a_{ij}) * b_i$ is reduced. Having tighter upper bounds means that fewer exact retrieval status values have to be computed until the top ranked document is determined. The retrieval effectiveness and the retrieval efficiency achieved by means of the reduced indexing vocabulary are presented in the next section.

4 Experiments

In this section, we present the evaluation of the method described above and we compare it to methods to other weighting schemes. We focus on the efficiency of modifying documents and on the correlation between the retrieval efficiency and the retrieval effectiveness. We will also see what is the influence of the vocabulary restriction on the retrieval effectiveness and on the retrieval efficiency. For the final evaluations we concentrated on the adhoc queries.

Before discussing the results, we define what we mean by a *partition*, a *run* and an *experiment*. The document collection has been split up into several *partitions*, each consisting of at most 100,000 documents. Thus, the large collections DOE1 (Department of Energy, Disk 1) and ZIFF3 (Ziff-Davids Publishing, Disk 3) were divided into three and two partitions respectively. A *run* consists of the evaluation of 50 queries (either the set of routing queries or the set of ad hoc queries) against all documents of one partition. For each query, the 1000 top ranked documents have been retrieved. An *experiment* consists of several runs and the merging of the lists of ranked documents for each query. For TREC-2, the two sets of experiments "Topics 51-100 versus Disk 3" and "Topics 101-150 versus Disks 1 and 2" have been evaluated.

All efficiency evaluations are based on CPU time rather than on real time in order to eliminate side effects from other jobs running on the same machine. In these experiments, we used a SUN SPARCserver MP690 with 128 MBytes RAM.

We derived the document descriptions directly from the CD's. The indexing process included the elimination of stop words (van Rijsbergen's stop list [12, pp. 18]) and Porter's word reduction algorithm [6]. The normalized inverse document frequencies have been derived from the documents of disks 1 and 2 only. Uncompressing and indexing a single document needs around 100 msec on an average depending on the length of the document. The computation of the inverse document frequencies from the descriptions took about 1.5 hours of CPU time.

The average time for inserting a document description into the access structure is on a scale of 10 msec - again depending on the number of features φ_i per document. Inserting a document description into an inverted file would need more time because the postings had to be inserted into the different lists associated to each feature.

The restriction of the vocabulary was accomplished by omitting features occurring in more than 15% of all documents (from the disks 1 and 2), i.e. in more than 111'337 documents. We have chosen 15% of the collection although also a stronger limit of 10% should not affect the retrieval effectiveness [1]. In our experiments we compare the 15% limit ("df15") to a non restricted vocabulary ("all").

We now have three parameters which can be combined to specify eight different retrieval methods. Each method can be identified by a string built from the labels for the document feature weighting, the query feature weighting and the vocabulary restriction:

$$\langle doc\_feat\_weight \rangle.\langle query\_feat\_weight \rangle.\langle vocab \rangle$$

In what follows, we present the results of the following nine methods:

M0 ntc.ntn.all

M1 lnc.ltn.all
M2 lnc.ltn.df15
M3 lnc.ntn.all
M4 lnc.ntn.df15

M5 ltc.ltn.all
M6 ltc.ltn.df15
M7 ltc.ntn.all
M8 ltc.ntn.df15

First, we compare the retrieval effectiveness of our method (M1) described in Section 2 to the standard $tf * idf$ method (M0) by means of the precision-recall graph in Figure 2. As expected, the method M1 is more effective than M0 and achieves a retrieval effectiveness among the best methods presented at TREC-2. In order to find out the reason for this difference in the retrieval effectiveness we must have a closer look at the influences of each parameter (document and query feature weighting).

In Figure 3, the 11-pts average precisions of each method (M0 to M8) are plotted on the left axis, and they are connected to the median response times (for the top ranked document) plotted on the right axis. The most obvious conclusion from this graph is the following: the higher the precision, the slower the response, and vice versa. The method M0 performs clearly worse than the methods M1 to M8 in respect to both retrieval effectiveness and retrieval efficiency.

We concentrate on the response times of the top ranked document because the response times of all further ranked documents are of secondary interest, since a user is supposed to read the top ranked document before looking at the other documents and the retrieval system can retrieve further documents while the user is reading the top ranked document.

We can also see from Figure 3, what are the *influences of the different parameters on the average precision.* Regarding the weighting of the document features, the "lnc" weighting achieves a 4-10% higher precision than the "ltc" weighting. The "ntc" looses 5% of precision compared to "ltc". In the case of query feature weighting, again the logarithmic "ltn" weighting is more

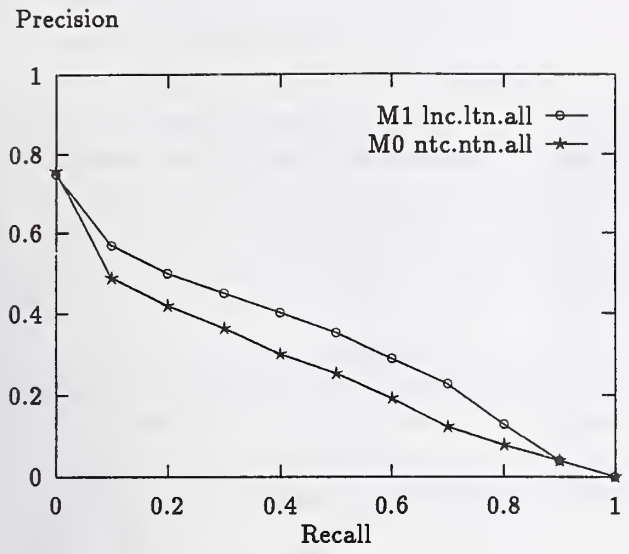


Figure 2: Precision recall graphs of the most effective method (M1) and of the least effective method (M0).

effective than the "ntn" weighting (10-15%). Restricting the vocabulary results in a 2-7% lower precision. We can summarize the experiences as follows:

- The $nidf(\varphi_i)$'s in the document feature weights have a bad influence on the retrieval effectiveness. It could be that for long documents the estimation of the lengths $|\vec{d_j}|$ is inappropriate when the $nidf(\varphi_i)$'s are taken into account.

- Logarithmic feature weighting is more appropriate for the long TREC documents and queries than linear feature weighting. Logarithmic feature weighting avoids an overweighting of features occurring very frequently within a document.

- Restricting the indexing vocabulary by ommitting features with a high document frequency $df(\varphi_i)$ has a noticeable influence on the average precision.

In what follows, we discuss the *influences of the different parameters on the response time* (as shown in Figure 3). Restricting the vocabulary accelerates the query evaluation (by 9-14%) for the reasons described in Section 2. The "ntn" weighting of the query features is also 9-14% faster than the "ltn" weighting. For document feature weighting, the "lnc" weighting is 5-10% slower than the "ltc" weighting. The "ntc" weighting of is even slower. These results can be explained in terms of the approximation error.

- It is obvious that for the "ltc" weighting the approximation error $(a_{ij}^0 - a_{ij}) * b_i$ is smaller than for the "lnc"

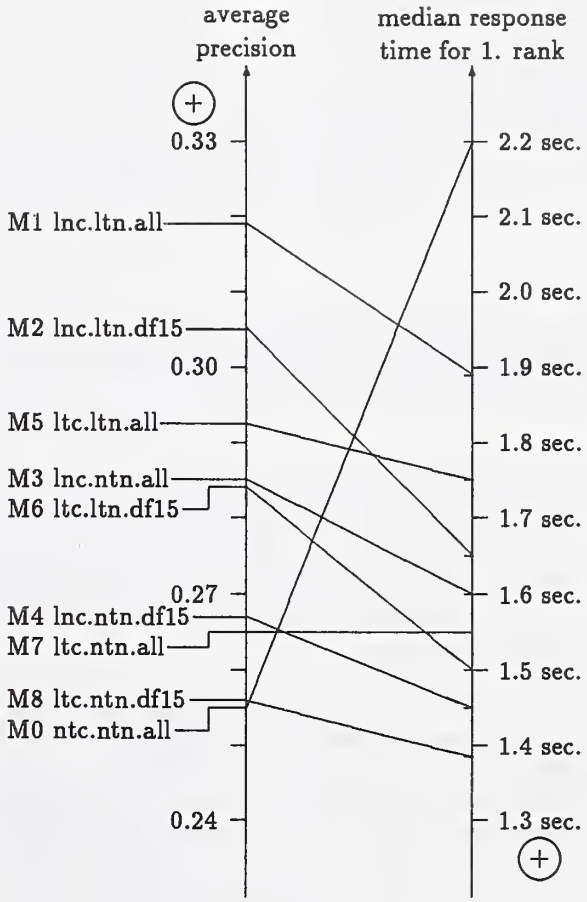


Figure 3: Average precisions and response times of the first ranked document.

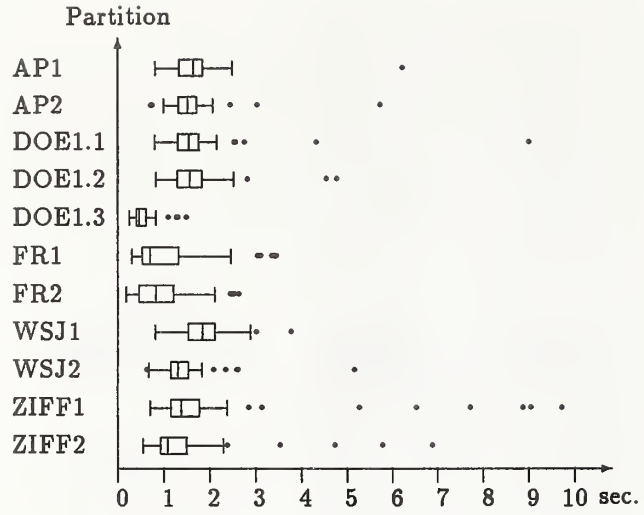


Figure 4: Response times of the first ranked document per run (adhoc queries versus the listed partitions) for the fastest method (M8 nltc.ntc.df15).

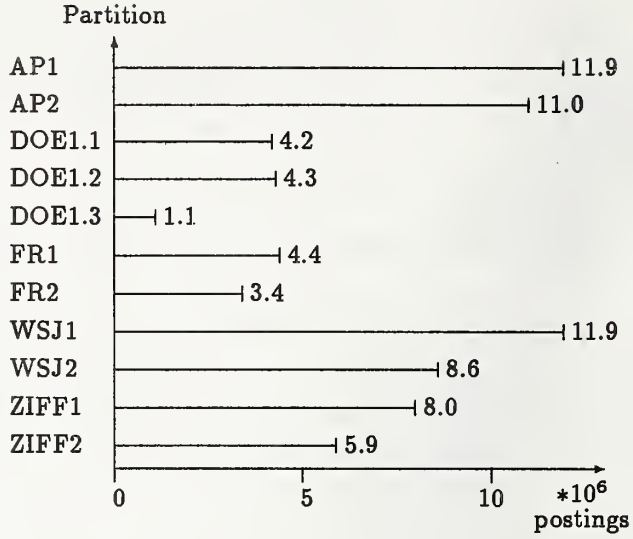


Figure 5: Number of postings per partition.

weighting because of the scaling with the $nidf(\varphi_i)$. On the other hand, the approximation error for the "ntc" weighting is very large compared to both the "ltc" and the "lnc" weighting because of the normalized linear weighting instead of the normalized logarithmic weighting.

The box plots [7, pp. 336] presented in Figure 4 show the distribution of the response times required to determine the top ranked document. A box plot visualizes the median (the line within the box), half of all samples (the box), and outliers (the dots) of a sample collection. In our case we have 50 samples: the response times of the 50 adhoc queries run against a partition. For most queries the top ranked document was retrieved in less than two seconds. The few outliers were all produced by the same queries (#103, #136, #138, #144). In general, the response times become shorter if a partition contains less documents and if the document descriptions consist of less postings on an average (see Figure 5).

5 Conclusions

In our approach we stressed the update efficiency. We have shown that the retrieval effectiveness does not have to be sacrificed to achieve a high update efficiency when coping with highly dynamic document collections. Our approach could probably be further improved by finding a weighting scheme that, on the one hand, achieves a very good retrieval effectiveness and that, on the other hand, can be approximated by frequency independent weights with only little variation from the exact weights. The retrieval efficiency could be improved by better partitioning the document collections according to the lengths of the documents. Our approach seems to be very amenable to parallel processing. We may think of several configurations (partitioning the query, partitioning the document collection, etc.). At this moment, it is not clear which configuration is appropriate for which requirements. Furthermore, we do not know yet how dynamic the document collection must be such that our access structure outperforms inverted files.

References

[1] C. Buckley, G. Salton, and J. Allan. Automatic Retrieval With Locality Information Using SMART. In *TREC-1 Proceedings*, pages 59–72, 1992.

[2] W. B. Croft and P. Savino. Implementing Ranking Strategies Using Text Signatures. *ACM Transactions on Information Systems*, 6(1):42–62, 1988.

[3] U. Glavitsch and P. Schäuble. A System for Retrieving Speech Documents. In N. Belkin, P. Ingwersen, and A. M. Pejtersen, editors, *ACM SIGIR Conference on R&D in Information Retrieval*, pages 168–176, 1992.

[4] D. Harman. Relevance Feedback Revisited. In N. Belkin, P. Ingwersen, and A. M. Pejtersen, ed-

itors, *ACM SIGIR Conference on R&D in Information Retrieval*, pages 1–10, 1992.

[5] D. Harman, E. Fox, R. A. Baeza-Yates, and W. Lee. Inverted Files. In W. B. Frakes and R. A. Baeza-Yates, editors, *Information Retrieval, Data Structures & Algorithms*, pages 28–43. Prentice-Hall, Englewood Cliffs, NJ, 1992.

[6] M. F. Porter. An Algorithm for Suffix Stripping. *Program*, 14(3):130–137, 1980.

[7] J. A. Rice. *Mathematical Statistics and Data Analysis.* Wadsworth & Brooks, Pacific Grove, CA, 1988.

[8] G. Salton and C. Buckley. Parallel Text Search Methods. *Communications of the ACM*, 31(2):202–215, 1988.

[9] G. Salton and C. Buckley. Improving Retrieval Performance by Relevance Feedback. *Journal of the ASIS*, 41(4):288–297, 1990.

[10] G. Salton and M. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, 1983.

[11] P. Schäuble. SPIDER: A Multiuser Information Retrieval System for Semistructured and Dynamic Data. In *ACM SIGIR Conference on R&D in Information Retrieval*, pages 318–327, 1993.

[12] C. J. van Rijsbergen. *Information Retrieval.* Butterworths, London, second edition, 1979.

1. Exact and Approximate Retrieval Status Values

$$\text{"}exact\text{"}: RSV(q,d_j) := \frac{1}{|\vec{d_j}|} * \sum_{\varphi_i \in q : \varphi_i \in d_j} a_{ij} * b_i$$

$$\text{"}approx\text{"}: RSV^0(q,d_j) := \frac{1}{|\vec{d_j}|} * \sum_{\varphi_i \in q : \sigma(d_j)[h(\varphi_i)]=1} a^0_{ij} * b_i$$

2. Document Length

$$|\vec{d_j}| := \sqrt{\sum_{\varphi_i \in d_j} {a_{ij}}^2}$$

3. Weights of Query Features

$$\text{"}ntn\text{"}: b_i := ff(\varphi_i, q) * nidf(\varphi_i)$$
$$\text{"}ltn\text{"}: b_i := (1 + log(ff(\varphi_i, q))) * nidf(\varphi_i)$$

4. Exact and Approximate Weights of Document Features

$$\text{"}ntc\text{"}: a_{ij} := \frac{ff(\varphi_i, d_j)}{\max\{ff(\varphi_h, d_j) \mid \varphi_h \in d_j\}} * nidf(\varphi_i)$$
$$a^0_{ij} := 1 * nidf(\varphi_i)$$
$$\text{"}lnc\text{"}: a_{ij} := \frac{(1 + log(ff(\varphi_i, d_j)))}{(1 + log(\max\{ff(\varphi_h, d_j) \mid \varphi_h \in d_j\}))}$$
$$a^0_{ij} := 1$$
$$\text{"}ltc\text{"}: a_{ij} := \frac{(1 + log(ff(\varphi_i, d_j)))}{(1 + log(\max\{ff(\varphi_h, d_j) \mid \varphi_h \in d_j\}))} * nidf(\varphi_i)$$
$$a^0_{ij} := 1 * nidf(\varphi_i)$$

5. Normalized Inverse Document Frequency

$$nidf(\varphi_i) := 1 - \frac{\log(1 + df(\varphi_i))}{\log(1 + n)}$$

Table 1: Classification of weighting schemes.

N-Gram-Based Text Filtering For TREC-2

William B. Cavnar
Environmental Research Institute of Michigan
P.O. Box 134001
Ann Arbor, MI 48113-4001
cavnar@erim.org

Abstract

Most text retrieval and filtering systems depend heavily on the accuracy of the text they process. In other words, the various mechanisms that they use depend on every word in the text and in the queries being correctly and completely spelled. To get around this limitation, our experimental text filtering system uses N-gram-based matching for document retrieval and routing tasks. The system's first application was for the TREC-2 retrieval and routing task. Its performance on this task was promising, pointing the way for several types of enhancements, both for speed and effectiveness.

1.0 Background

Even though modern character recognition techniques are steadily improving, scanning and recognizing characters in paper documents still results in many errors. These errors can impact the ability of a standard, classical text retrieval system to process the resulting ASCII text by interfering with word stemming, stopword identification, or even basic indexing. Furthermore, as paper documents age, their print quality can degrade, possibly causing further recognition problems. This is becoming a serious problem, especially given the vast amount of material that exists in a paper form that has a limited lifetime.

Another difficulty is that the texts themselves may contain spelling variations (e.g., British vs. American) or outright spelling errors. The same is true of the queries entered by a human user. Discrepancies between the spelling of a word in a document and a word in a query may prevent a match.

A third problem area is that current word stemming and stopword list construction algorithms sometimes involve a significant amount of knowledge about the documents' language. Although there are automatic approaches to acquiring this kind of knowledge, it is complicated [1]. An ideal text retrieval method would require no special language knowledge, and in fact would be able to handle multiple languages (represented, say, in Unicode) simultaneously.

All of these problems have analogs in a somewhat different problem domain, namely, reading and interpreting postal addresses on pieces of mail. For the last eight years, the Environmental Research Institute of Michigan (ERIM) has conducted research for the United States Postal Service (USPS) in automatic address interpretation. The ultimate goal of this research is to build a high-performance address interpretation unit that can reliably and efficiently read the address information on a mailpiece, and determine its correct 9-digit or 11-digit ZIP code, even if that code is not present on the mailpiece or is in error.

There are two parts to interpreting an address from a mailpiece, and both of them are difficult:

- Recognizing the characters in the address block. This is difficult because of poor print quality, poor contrast, complicated backgrounds, presence of logos and non-address information, use of proportional fonts, and a host of other problems.

- Interpreting the lines of the address. This is difficult because of spelling errors, addressing errors, unusual abbreviations, strange local addressing conventions, and errors and omissions in the USPS databases.

Our current address interpretation systems use a number of different techniques to get around these problems. Among these is the use of N-gram-based matching to find matches in postal databases. (Please see [2] and [3] for details of these systems.)

In this work, N-grams are N-character slices of some longer string. We do not use positional N-grams; i.e., what is important is whether a given string contains a particular N-gram, not where it falls in the string. We use bi-grams (N=2) and tri-grams (N=3) together in the same system. Bi-grams and tri-grams complement each other in the sense that bi-grams provide somewhat better matching for individual words, while tri-grams provide the connections between words to improve phrase matching.

N-gram-based matching addresses the difficulties mentioned above for postal addresses in ways that have application for document retrieval:

- It usually finds good matches for postal addresses even in the face of a significant number of individual character recognition errors. This is particularly true if the system can use redundant data for the search key, such as the city, state, and ZIP together. This is identical to the problem of matching document text that was scanned from poor quality images. Words that occur near one another in a text also "reinforce" each other for searching the way that city, state, and ZIP do.

- It easily overcomes problems with spelling and addressing variations and errors on mailpieces or in postal databases. This is analogous to the problem of matching document text that contains spelling variations. It also achieves an effect similar to stemming in that different words having the same root are considered to be similar.

- It can also handle abbreviations fairly easily. This is also similar to the problem of word stemming.

- Some postal address interpretation systems depend heavily on the system's ability to recognize certain key words, such as STREET or BOX. These words play a role similar to stopwords in that they carry little content, and mostly provide structural information. N-gram-based matching can handle errors even in these words.

Drawing on the experience of using N-grams in postal work, ERIM has been adapting these techniques for use in full-text retrieval and filtering. We currently have a working full-text retrieval system, called *zview*, which provides a simple query facility for accessing a wide variety of ASCII documents, including biographies, abstracts, and documentation. The *zview* system's N-gram-based matching provides a very simple, easy-to-use text retrieval system. It is tolerant of spelling variations and errors in both the text and the documents. It also provides a straightforward ranking of documents that enables the user to see which documents best match the query. This system has given us some confidence in the wider utility of N-gram-based matching for information retrieval. Although the TREC task would not seem to directly require inexact matching (the documents were high quality ASCII), our experience with *zview* indicates that this kind of tolerance for variations in user input is very useful for general retrieval.

Incidentally, one criticism that one can make of N-gram-based systems is that they require large amounts of storage. Although this is true of the naive implementation, there are a number of variations using various superimposed coding techniques that provide substantial space savings without impacting matching performance. See [2] for details.

2.0 System Description

The *zview* system mentioned above has two parts: an off-line index builder, and an on-line retrieval interface. In its current form, *zview* does not yet use our most compact N-gram representation techniques, and is thus not well-suited for very large (i.e., gigabyte size) databases. Also, it can handle only single query strings, not the multiple query strings necessary for TREC. Instead, for this task we opted to use a simple filter architecture, and concentrated on making it as fast as possible. This architecture has several advantages:

- It requires no disk space for an index. In fact, we were able to save even more disk space by leaving the original documents in their compressed form, and uncompressing them on the fly.

- It is conceptually simple, requiring only a very modest amount of code (987 lines of C source). The whole system took only a few days to design and build.

This architecture is similar in concept to that used by Mark Zimmerman's PARA system, which participated in last year's TREC. That system used a set of handcrafted Unix *awk* scripts to filter the documents. The PARA system, although it got good results, took 22 days on a more or less dedicated NeXT machine, even though its match criteria generally specified fewer match terms than our system did. Also Zimmerman's system used the notion of proximity to identify sets of matching lines that occurred near one another, whereas our system counts matches throughout the document without regard to the proximity of match terms. See [4] for details.

Obviously, a significant disadvantage of such a filter-type architecture is that the system has to read all of the data to simulate a retrieval. This is very computationally intensive, but we were able to mitigate this somewhat in our system by handling much of the actual N-gram-based matching with a highly parallel bit-vector approach, which we describe below. Also, we were able to split the processing load up among a suite of computers and run them all in a coarse-grained parallel fashion.

Figure 1 gives an overview of our system in dataflow fashion. We discuss each of the processes in this diagram in the following sections.

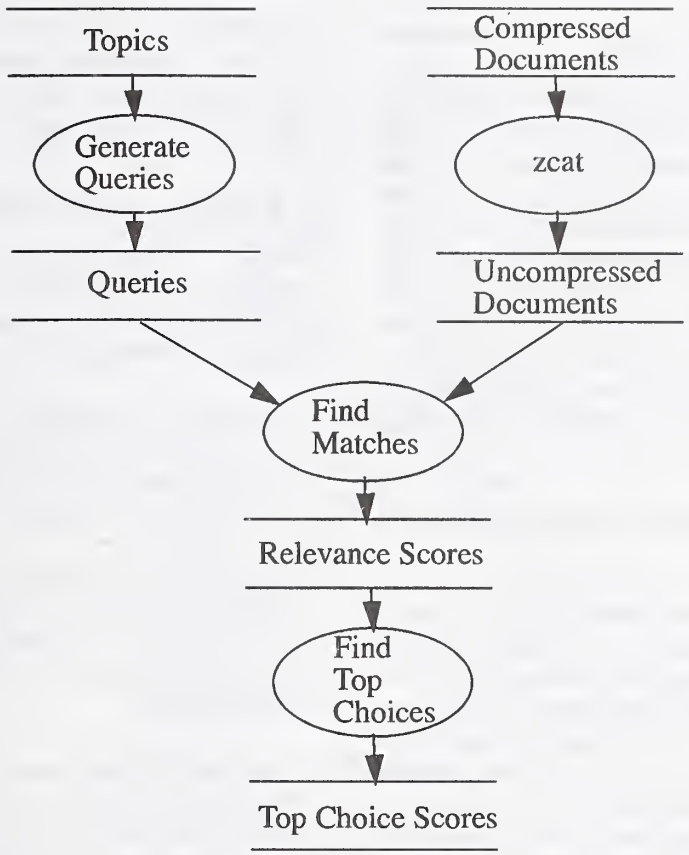


Figure 1: Dataflow Diagram for N-Gram-Based Text Retrieval

2.1 Generating Query Strings From TREC Topics

The process labelled "Generate Queries" represents a program *gen_query*, which takes the TREC topics file and generates a set of query strings for each topic. We considered several different schemes for extracting query strings from the topics, but finally settled on using just the phrases in the concept and nationality sections. The following is a typical topic from TREC-1:

<num> Number: 007
<title> Topic: U.S. Budget Deficit
<desc> Description: Document will mention a proposal to decrease the U.S. budget deficit.
...<con> Concept(s):
1. U.S. budget deficit, federal budget shortfall
2. foreign affairs budget, defense budget, entitlements
3. increased revenues, tax increase, tax reform, auction quota
4. reduction in expenditures, spending cuts, cutting domestic
programs, eliminating government subsidies
5. NOT financing the U.S. budget deficit
...
<nat> Nationality: U.S.

Given this topic, the *gen_query* program generates the following set of query strings:

007 000 U.S.
007 000 U.S. budget deficit
007 001 federal budget shortfall
007 002 foreign affairs budget
007 003 defense budget
007 004 entitlements
007 005 increased revenues
007 006 tax increase
007 007 tax reform
007 008 auction quota
007 009 reduction in expenditures
007 010 spending cuts
007 011 cutting domestic programs
007 012 eliminating government subsidies
007 013 NOT financing the U.S. budget deficit

Each line of the query output corresponds to one nationality or concept string from the topic. The nationality strings are listed first, followed by the concept strings in the order they appeared. There is also a rank number associated with each query string. Generally, as concept strings were used in the topics, strings that appeared earlier in the list were more important than strings that appeared later. The actual filtering program also uses these rank numbers to weight the query strings in importance. Each string receives a weight equal to 2k - r, where k is the number of query strings for the topic and r is the string's rank value. Any nationality or concept string can also have a NOT prefix, which triggers a slightly different kind of processing for these strings, described below.

2.2 The Problem of Finding Matches While Filtering Text

In Figure 1 above, the process labelled "zcat" represents a Unix utility program that takes a compressed file and returns its uncompressed form. The process labelled "Find Matches" represents a program *find_matches*, which takes a set of queries for all of the topics at once, and applies them to a single uncompressed text file. Given the filter architecture for this system, it was important for us to minimize the total amount of I/O. By letting *find_matches* process the queries for all topics simultaneously, the system has to pass the data only once.

To make this process as efficient as possible, *find_matches* does most of its matching using bit-vectors representing the presence or absence of particular N-grams. Suppose that we had a string that we wanted to search for in a document using N-grams. An obvious way to do it would be to perform the following steps:

- Break the query string up into N-grams.
- For every line in the document, count how many of those N-grams occurred in the line. This count would be the match score for that line.
- Keep a sorted list of the top-scoring lines, bumping the lowest scoring entry on this list when a new line has a better score.

When this algorithm finishes, we will have a list of the top scoring matches for our query string. We could make this process a little more sophisticated by putting a threshold of some sort on it. If a line's match score was below this threshold, say 80% of the number of N-grams in the query string, we could ignore this line as simply not containing enough of the query string to consider.

Unfortunately, this naive implementation of N-gram-based matching, which uses nested loops to compare every N-gram in the query string with every N-gram in a line, is computationally very costly.

2.3 Matching and Scoring Lines

We can do significantly better than the naive N-gram-based algorithm described above by using hashing. Figure 2 illustrates this process for bi-grams. The key idea here is to break the matching process into two parts:

- Initialize a hash-code-based look-up table for all of the N-grams in the query string. Each slot simply records whether *any* N-gram hashed to that position (1) or not (0). Thus the hash table is a simple bit vector.
- For each line in the document, make a single pass for all of the line's N-grams, checking each one to see if it is in the table. In other words, see if the slot (bit) corresponding to the N-gram contains a 1. If it does, add 1 to the line's score.

If we make the hash table big enough, we can make the probability of collisions arbitrarily small. For example, consider that there are only 52,022 possible bi-grams and tri-grams for the alphabet consisting of A-Z, 0-9, and space. A hash table size of, say, 20,000 slots (625 32-bit words or 2500 bytes) provides far more than ample space to avoid collisions. If there is a collision between the hash of an N-gram in the query string and that of some different N-gram in the line, its only effect is to cause that line's score to be one higher than it should be. It is even more unlikely that two or more of the N-grams would also collide. Thus by allowing collisions in the hash table, we get a simple, quick algorithm at the cost of a small probability of small distortions in the line scores. See [2] for a further discussion of how to estimate the probability of these kinds of collisions. Note that we include leading and trailing spaces for words in counting N-grams. Also recall that in our full implementation of this we process tri-grams the same way and at the same time. The match score between two strings includes the count of both matching bi-grams and matching tri-grams.

To get an additional gain in efficiency, we can carry this hash table scheme for counting N-grams another step by using parallelism to match a single line from a document against a number of query strings all at the same time. Figure 3 illustrates this scheme. Again, matching is a two-step process:

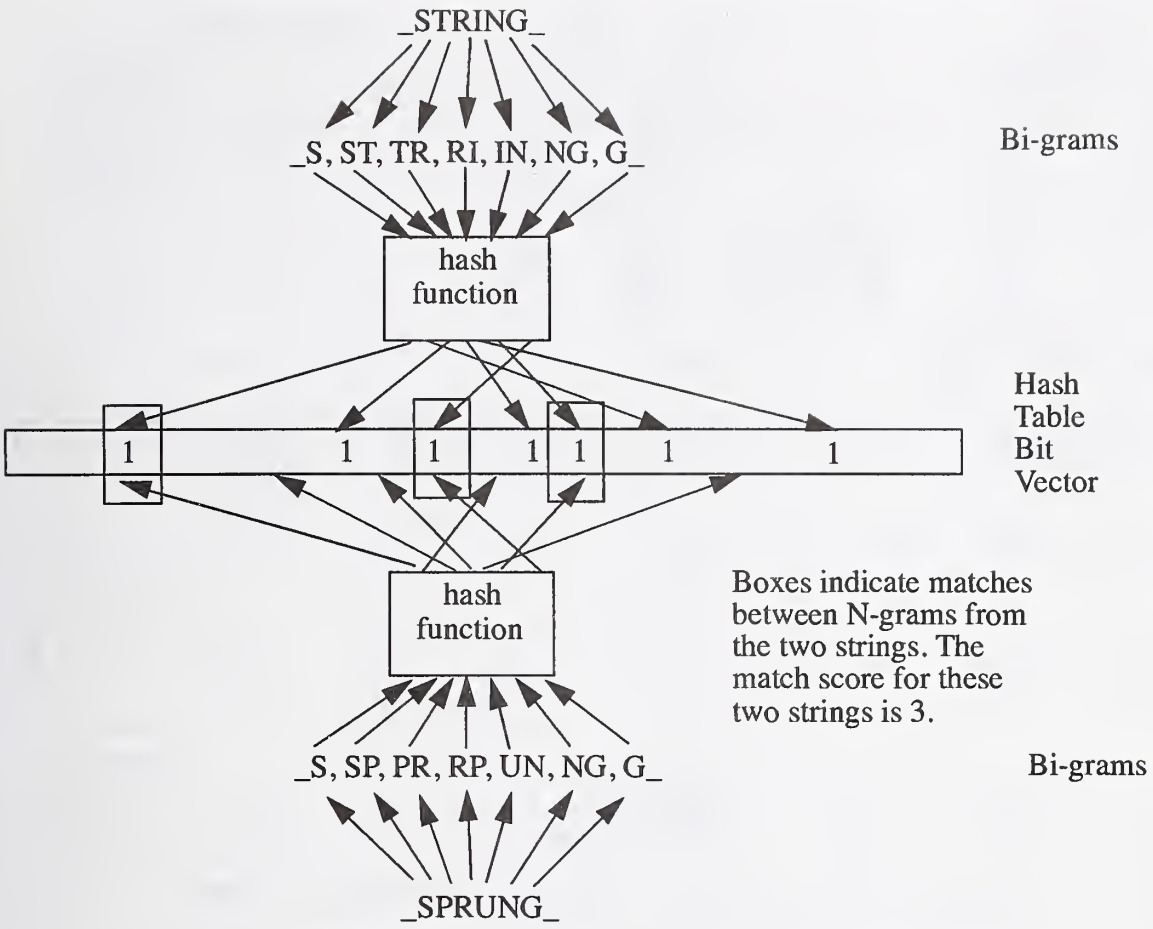


Figure 2: Using Bi-Grams to Compare "STRING" and "SPRUNG"

- Initialize a hash table for the entire set of query strings. Each slot in the hash table again represents one N-gram, but it is now a bit vector in which the kth bit indicates whether the N-gram occurs in the kth query string.

- For every line in the document, hash each N-gram to find the corresponding bit-vector. Then add that bit vector to the current score vector for the document. The kth element of this score vector keeps the current match score for the document against the kth query string.

One other significant aspect of this multi-query-string implementation is that we also represent the final score vector for all of the queries as a set of parallel bit-vectors. This effectively allows us to perform 32 additions in parallel by adding one 32-bit word of an N-gram bit-vector from the table to one 32-bit word representing the 1-bits of 32 query string scores, and then propagate the carry bits up to bit-vectors representing the higher order bits of the scores. Effectively, this lets us compute scores for 32 query strings in close to the same amount of time as it takes to compute the score for a single query string.

2.4 Scoring and Ranking Documents

After the system computes the N-gram match scores of a single line against all of the query strings, it applies a threshold criterion. If the score for a particular query string is below its threshold, the system treats it as a zero score. The threshold is defined as a percentage of the maximum possible N-gram score for the query string. For example, if a particular query string has a maximum possible score of 20 matching N-grams, and the threshold percentage (as determined by a command line argument to the program) is 70%, then the system will ignore any line whose score is less than 14. There is a similar command line argument which determines the threshold percentage for negation query strings (those tagged with a NOT prefix in the query string set). If a document has a line whose score for a negation query string exceeds its threshold, the system will discard the document. For most runs of the system, we got the best results with a match threshold at 70% and a negation threshold at 95%. In other words, we were willing to accept a line as a match for a given query string if it had 70% of the

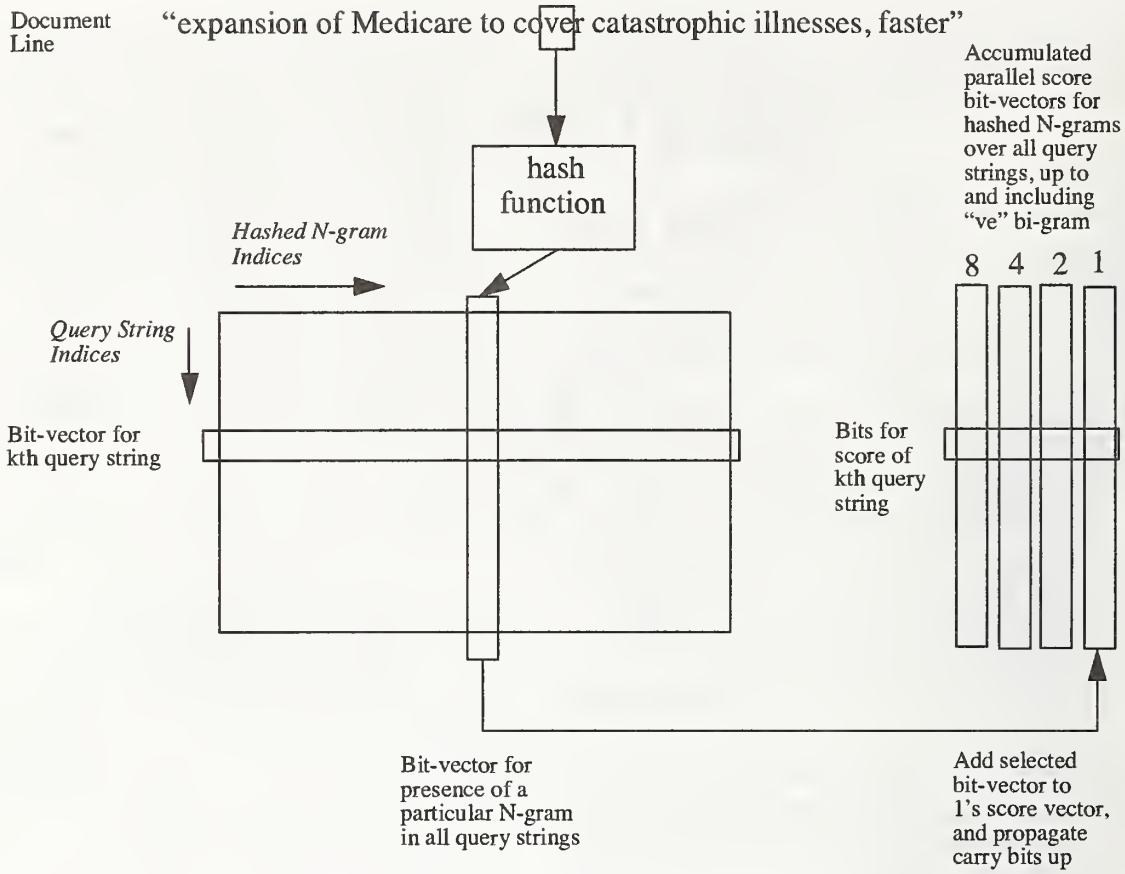


Figure 3: Parallel Implementation for Multiple Query Strings

requisite N-grams, but needed a nearly exact match in order to discard it for matching a negation query.

To compute the scores for a whole document, we accumulated the line scores for each query string, but applied a cap at twice the maximum possible N-gram score. In other words, if a document mentions a particular query string twice, that is twice as good as mentioning it once. However mentioning it three or more times is of no additional value. We put this cap into the system after observing that initial versions of the system would overrate some documents because they mentioned some low ranking query string for a given topic many times, but completely missed any of the higher ranking query strings for the topic.

To generate the final scores for a document against a topic, the system accumulates the scores for each query string in the topic, multiplying them by the weighting factor computed earlier. If this aggregate weighted score exceeds a hard-coded threshold of 40, the system then writes out this score as a relevance judgement for the document. Later a separate program reads the file containing the relevance scores, computes the top 1000 scores for each topic, and writes them out as the final system result.

3.0 Multi-Processor Execution

Even with all of this attention to efficient implementation, the system still required a large amount of computation. Fortunately, ERIM had available a network of Sun workstations over which we could distribute the processing load. This network consisted of two Sun SPARCstations and five or six Sun SPARCstation 2 machines. We used a very simple partitioning scheme, assigning each machine a fixed set of documents to evaluate against all of the topics. Given this arrangement, a full run of topic set 3 against Disks 1 and 2 took 12 to 13 hours if all of the machines were fully available. When there was significant contention on some of the machines, the run might take as much as 24 hours. An obvious drawback with using the fixed partitioning scheme was that often one or more of the machines would not finish until many hours after all of the others had. This was usually due to unexpected heavy contention on one or more machines. If we had used a more intelligent and dynamic document file assignment mechanism (implemented in, say, the Linda coordination language), we would have had better elapsed execution times.

TABLE 1. Selected TREC-2 Test Results

Test Number	Note	Test Type	Query Thresh.	Negation Thresh.	Average Precision	At 10 Docs	At 100 Docs	R-Precision
erima1	official	ad hoc	80	80	0.1589	0.3960	0.3016	0.2179
erima2	official	ad hoc	75	80	0.1885	0.4480	0.3426	0.2494
a30	official	ad hoc	70	95	0.1604	0.4040	0.3038	0.2198
a31	official w/ X queries	ad hoc	70	95	0.1153	0.4000	0.259	0.1904
a33	spanning lines	ad hoc	70	95	0.0734	0.15	0.1484	0.1333
a34	span lines; exp. decay weighting	ad hoc	90	95	0.1099	0.314	0.2276	0.1776
a35	span lines; exp. decay weighting	ad hoc	80	95	0.1336	0.3000	0.2442	0.2004
a35-AP	AP only	ad hoc	80	95	0.2717	0.4740	0.2838	0.3140
a36	span lines; exp. decay weighting	ad hoc	70	95	0.0792	0.1780	0.1678	0.1387
erimr1	official	routing	80	80	0.1219	0.3580	0.2304	0.1814
erimr2	official	routing	75	80	0.1415	0.4240	0.2524	0.2031
r5	official	routing	70	95	0.1225	0.3600	0.2310	0.1818
r6	span lines; exp. decay weighting	routing	80	95	0.1015	0.2880	0.1954	0.1604
r7	span lines; exp. decay weighting	routing	70	95	0.0602	0.2200	0.1388	0.1123

4.0 Results

To test our system, we ran it a number of times, varying different system parameters. After the conference, we were also able to make a few changes and run it again. Table 1 above summarizes some highlights of our results, which show a number of interesting points:

- The tests numbered erima1, erima2, erimr1 and erimr2 were the official results turned in on June 1. The first two were the ad hoc results, and the second two were the routing results.

- The tests numbered a30, a35, a36, r5, and r7 were all attempts to determine good settings for the query threshold and negation threshold parameters. Unfortunately, the results from these test runs simply do not provide anywhere near enough data to perform a complete sensitivity analysis for these parameters. Also, we noticed in some of our testing on the TREC-1 queries that there is considerable difference in the optimum values of these thresholds for different topic sets/data set combinations.

- One of the motivations for using N-gram-based matching was that it provides good matching performance in the face of textual errors. To test this idea, we ran test a31 using deliberately damaged query strings. In this test, we took each query string produced by *gen_query*, and replaced the third character with the letter "X". This works out to be an effective character recognition error rate of 4.5% over the whole body of query strings. Although the system took a considerable hit in performance, the interesting thing was that it still functioned at all. Many of the other systems in the TREC evaluation would most likely have completely failed in an analogous test, since they depend heavily on exact word matches.

- One serious drawback to the original system was that it did not span lines when matching. That is, if the text that

a particular query should match was split across two document lines, then the match would fail. After the conference, we found a relatively easy way to rectify this design deficiency and were able to run tests a33 though a36, and r6 and r7. Unfortunately, this change greatly affected the system's response to the query and negation thresholds, and we were not able to run enough tests to find the optimum values for these parameters. The end result is that the best results we have for this improved system are still not as good as the official results we reported.

- Another serious problem with the original system came to light during the official conference. As Donna Harman pointed out in her closing talk, no one has yet really explored the full ramifications of changes in term weighting strategies. Our original system used a very simple-minded linear-falloff weighting scheme. Our assumption was that concept strings appeared in reverse order of importance. However, we began to suspect, based on different concepts that were mentioned in a number of the conference presentations, that this was overly simplistic. We decided to implement a straightforward exponential-decay weighting scheme. In this approach, the first query string gets a weight of 100, the second gets a weight of, say, 90, the third a weight of 81, the fourth a weight of 72, and so on, with each succeeding weight taking 90% of its predecessor's value. Unfortunately, we did not have time to tune the system's response to this change either, and its results (a34, a35, a36, r6, and r7) are worse than the official results as well. However, it appears that there is plenty of room for experimentation with different term weighting schemes and we will continue working with them.

- In our early work with the system before sending in the official results, we did a fair amount of testing using just the Associated Press document set. We were perhaps misled by how well the system did on these documents, and missed some chances to improve the system earlier. The test labelled a35-AP shows what the results look like for a35 if we restrict the new system to just returning AP documents, and restrict the relevance judgements to just AP documents. Even with this imperfectly tuned new version, we see that the system is capable of significantly better performance. It is unclear why there should be such variation between the retrievability of the AP documents and the other document collections.

At its best, our system performed as well as most of the systems that participated in TREC-1. However, there is ample room for improvement, as we have noted above, especially in comparison to many of the systems that came back for TREC-2.

5.0 Further Research

The TREC-2 task is the first real application for our N-gram-based multiple-query system. As in any experiment of this nature, the results and problems suggest many more possible avenues of research. These ideas fall into two categories.

5.1 Analyzing the Current System's Performance

Further analysis of the existing system will allow us to better understand its behavior and limitations. Some ways to do that include:

- It is likely that generating query strings from the topic concept strings may have significantly limited performance. For example, Topic 74 about instances where the U.S. government propounds conflicting policies completely failed to mention terms such as *policy* or *regulation* in the concept list. Thus, our system had only a very small chance whatsoever of finding matching documents. Zimmerman's filtering system [4] did well with handcrafted queries, so we should also try manually generated queries.

- Currently the system has a hard-coded cutoff threshold of 40 for the weighted aggregate score. The purpose of the threshold was to prevent the system from returning results that were guaranteed to be noise because of their very low score. This value was set more or less arbitrarily, so we should experiment with changing this threshold to determine its true effect. In all likelihood, it could be a fair amount higher, preventing the system from generating other useless low-scoring results.

- Currently the system sets a cap of three times the maximum N-gram score for any query string score. Again, this value was determined only by a very rough empirical process, so we should experiment with changing this cap, to see how much impact it has.

5.2 Extending the System

We can also make some significant changes to the system to explore possibilities for other performance improvements.

- Currently the system treats upper and lower case alike for both documents and queries. Since acronyms and brand names have different meanings sometimes from uncapitalized words having the same letters, perhaps there is a way to take the case of letters into account when computing a match. That is, we could count a

match between two words with different capitalizations as a good match, but a match between two words with the same capitalizations as a better one.

- Our system is basically just a text filter. As such, it is already close to being useful for various routing tasks. However, to use N-gram-based matching for on-line retrieval, we will have to implement a true index-based system. Ideally, we would like to integrate the text retrieval capability of our *zview* system with the flexibility of the multi-query filtering system described here. Although we have an initial design sketched out, it will take an investment of further time and machine resources to implement this idea and test it. We will also be using some of our new compact N-gram representation techniques to reduce the large amount of index storage and computation required.

Acknowledgments

We would like to thank our sponsors at DARPA/SISTO for allowing us the opportunity to participate in TREC-2. We would also like to thank Donna Harman and her staff at NIST for all of their help. Finally, we are grateful for the United States Postal Service's sponsorship of the original N-gram-based matching technology that inspired our TREC research.

References

[1] W. B. Frakes. Stemming Algorithms. In William B. Frakes and Ricardo Baeza-Yates, Editors. *Information Retrieval: Data Structures & Algorithms*, pages 131-160. Prentice Hall, Inc. Englewood Cliffs, NJ, 1992.

[2] William B. Cavnar and Alan J.Vayda. Using superimposed coding of N-gram lists for efficient inexact matching. In *Proceedings of the Fifth USPS Advanced Technology Conference*, pages 253–267, Washington, DC, 1992.

[3] William B. Cavnar and Alan J.Vayda. N-gram-based matching for multi-field database access in postal applications. In *Proceedings of the 1993 Symposium On Document Analysis and Information Retrieval*, pages 287–297, University of Nevada, Las Vegas.

[4] Mark Zimmerman. Proximity-correlation for document ranking: The PARA Group's TREC Experiment. In *Proceedings of the First Text REtrieval Conference (TREC-1)*, NIST Special Publication 500-207, 1992.

Retrieval of Partial Documents

Alistair Moffat* Ron Sacks-Davis† Ross Wilkinson‡ Justin Zobel§

Information systems usually retrieve whole documents as answers to queries. However, it may in some circumstances be more appropriate to retrieve parts of documents. These parts could be formed by arbitrary division of running text into pieces of similar length, or by considering the document's hierarchical structure. Here we consider how to break documents into parts, how to implement retrieval of parts, and the impact of division of documents on retrieval effectiveness.

1 Introduction

Provision of answers to informally phrased questions is a central part of information retrieval. These answers traditionally take the form of documents retrieved from a text database, but documents will often be unsatisfactory as answers. They may be large and unwieldy; the answer they represent may be diffuse, and therefore hard for the user to extract; and word-based retrieval systems may be misled by the breadth of vocabulary of a long document into believing it to be relevant.

Indexing and returning parts of documents addresses these problems. We have approached the problem of partial documents in two ways. The first approach is to regard documents as an unstructured series of "pages" of text of similar length, each of which can be returned as an answer to a query. We would expect, under this approach, that any bias in the retrieval mechanism towards documents of a particular length should be eliminated. By regarding an answer to be the document from which an answer page is drawn, paging can be used even in contexts where documents are required as answers, as is the case for the TREC experiments.

Breaking documents into pages, however, has implications for implementation: the growth in the number of candidate answers is such that current approaches for evaluating queries have unacceptable memory requirements and response times. We have developed new algorithms for implementing information retrieval methods on large collections, concentrating on the cosine measure with IDF term weights as a typical example. These include techniques for efficiently constructing and compressing large inverted files, and for restricting the amount of memory space and processing time required during query evaluation. The result is that we are able to identify answers in a fraction of the space and time required by previous methods. Using these techniques on a version of the TREC data in which the documents are broken into over 1.7 million pages, answers can be found more quickly than they could previously be found on the unpaged data, even though the latter has a smaller index and fewer records. Section 2 gives an overview of these techniques.

Our second approach to the problem of partial documents was to regard documents as hierarchical structures. Some of the documents in TREC are very large. It is not clear that it is desirable to return such documents as a whole, nor is it clear that these documents should be indexed as a whole. Most of the long documents in TREC are from the Federal Register collection and all have some degree of structure associated with them. We conducted a set of experiments that attempted to determine whether these documents should be indexed as single objects, and whether the documents' structure could be used in conjunction with the contents of its elements.

We also experimented with retrieval of partial documents and investigated whether context, that is the rank of the whole document, helped improve ranking of sections. The experiments with hierarchical structures are necessarily based on the small set of longer documents in the TREC

*Dept. of Computer Science, The University of Melbourne, Parkville, Victoria, Australia 3052; alistair@cs.mu.oz.au

†Collaborative Information Technology Research Institute, 723 Swanston St., Carlton, Victoria, Australia 3052; rsd@kbs.citri.edu.au

‡Dept. of Computer Science, RMIT, GPO Box 2476V, Melbourne, Victoria, Australia 3001; ross@cs.rmit.oz.au

§Dept. of Computer Science, RMIT, GPO Box 2476V, Melbourne, Victoria, Australia 3001; jz@cs.rmit.oz.au

collection. We would expect that, in a large database of such documents, the implementation techniques we developed for unstructured text could be used for structured documents almost without change.

2 Retrieval of paged text

The first issue that must be considered when accessing a set of documents as a collection of pages is the pagination strategy; one possible method for pagination is discussed in Section 2.1.

Then, given paged text, answers must be found in response to documents. In an inverted file text database system of the kind we have been developing [6, 9], the implications of paging are a large increase both in the number of accumulators used to hold the intermediate cosine values and in the length of the inverted file entries. The former means that evaluating a ranking will require more memory; while the latter implies a longer time to resolve queries. Methods for avoiding these increases are outlined in Sections 2.2 and 2.3; a full description is given elsewhere [10].

Finally, there is the impact on effectiveness of the decision to paginate the documents. Experimental results comparing document retrieval with page retrieval are described in Section 2.4.

2.1 Breaking text into pages

One way to store text in a database is as a set of records, each containing a single document. Such an organisation implies that entire documents must be retrieved in response to queries. Moreover, a query can match long documents in which its words are widely separated and could well be unrelated, so this storage strategy can result in irrelevant material being retrieved.

An alternative way of presenting text is as a series of *pages*. There are several advantages to using pages:

- they are of a more manageable size than whole documents, and the size variation between pages can be constrained to be far less than the size variation between documents;
- if a user looks for answers matching a query, it is likely that in relevant material the query's words will occur close together in the retrieved text;
- retrieving a page of text to display may be considerably cheaper than retrieving an entire document;

1. Set $prev \leftarrow$ *undefined* and $curr \leftarrow w_1$.
2. For each j from 2 to n,
   (a) If $curr \geq B$, emit $prev$ if it is defined, set $prev \leftarrow curr$, and set $curr \leftarrow w_j$.
   (b) Otherwise, if $w_j > prev$ then set $prev \leftarrow prev + curr$ and set $curr \leftarrow w_j$.
   (c) Otherwise, set $curr \leftarrow curr + w_j$.
3. If $curr < B$ then set $prev \leftarrow prev + curr$ and emit $prev$. Otherwise, emit $prev$ followed by $curr$.

Figure 1: Paging algorithm

- in many applications it is natural to regard documents as consisting of parts rather than as a whole; for example, the nodes in a hypertext system or the sections of a book; and
- only parts of documents can, in general, fit on a screen, and people can only comprehend part of a document at a time.

For these reasons—and because of the challenge posed by the sheer size of a paged version of the TREC collection—experiments were undertaken with paged text.

The paging strategy adopted first breaks documents into minimal units likely to be useful for ranking. In most collections this unit would probably be the paragraph. The algorithm shown in Figure 1 is then used to gather paragraphs into pages, where w_i is the length of the i'th paragraph of the original document and B is the target length of each constructed page. Documents of fewer than B bytes must, of course, be allocated singleton pages, but all other pages are B bytes or more. Moreover, the aim is that only a small number of pages are significantly longer than B bytes. The algorithm requires time linear in the length of the text and a constant amount of space, and so is a small additional step during database construction. For the experiments described below, length was measured in bytes (the alternatives being words, or some more abstract length value based upon term frequencies), and $B = 1{,}000$ was used.

The pagination of the 2,055 Mb TREC collection resulted in the 742,358 documents being converted into 1,743,848 pages of average size 1,235 bytes each.

2.2 Management of accumulators

An effective way to rank documents against a query is with the cosine measure, defined by the function

$$C_d = \frac{\sum_t w_{q,t} \cdot w_{d,t}}{\sqrt{\sum_t w_{q,t}^2} \cdot \sqrt{\sum_t w_{d,t}^2}},$$

where q is the query, d is the document, and $w_{x,t}$ is the weight of word t in document or query x. We assume that $W_d = \sqrt{(\sum_t w_{d,t}^2)}$ is the length of document d, and that the top r answers are to be retrieved and returned to the user.

In our experiments term weights were calculated using the IDF rule $w_{d,t} = f_{d,t} \cdot \log(N/f_t)$, where $f_{d,t}$ is the number of appearances of term t in document d, N is the number of documents in the collection, and f_t is the number of documents that contain term t. Harman [3] gives a summary of ranking techniques and a discussion of the cosine measure and IDF weighting rule.

To support the cosine measure in an inverted file text database system, each inverted file entry contains a sequence of $\langle d, f_{d,t} \rangle$ pairs for some term t. The value f_t, the number of documents containing t, is stored in the lexicon, and from these two the cosine contribution $f_{d,t} \cdot (\log(N/f_t))^2$ of term t in document d can be calculated. A structure of accumulators is used to collect these contributions. As the inverted file entry for each query term is processed, the accumulator value for each document number in the inverted file entry is updated by adding in each partial cosine value as it is calculated, so that by the time all inverted file entries have been processed the accumulator for document d contains the value

$$\sum_t f_{d,t} \cdot (\log(N/f_t))^2 = \sum_t w_{q,t} \cdot w_{d,t}.$$

The number of accumulators required to process a query can be large. Queries were created from TREC topics 51–100 by simply extracting and stemming all of the alphanumeric strings, and stopping 601 closed-class words. The generated queries had an average of over 40 terms, and each resulted in about 75% of the pages of TREC having a non-zero accumulator value. With this ratio the most memory-efficient accumulator structure is an array. But at 32 or 64 bits per accumulator, this structure is formidably large, and dominates the retrieval-time memory requirements. Even initialising it is time-consuming.

1. Order the words in the query from highest weight to lowest.
2. Set $A \leftarrow \emptyset$; A is the current set of accumulators.
3. For each word t in the query,
   (a) Retrieve I_t, the inverted file entry for t.
   (b) For each $\langle d, f_{d,t} \rangle$ pair in I_t,
      i. If the accumulator $A_d \in A$, calculate $A_d \leftarrow A_d + w_{q,t} \cdot w_{d,t}$.
      ii. Otherwise, set $A \leftarrow A + \{A_d\}$, calculate $A_d \leftarrow w_{q,t} \cdot w_{d,t}$.
   (c) If $|A| \geq L$, go to step 4.
4. For each document d such that $A_d \in A$, calculate $C_d \leftarrow A_d / W_d$.
5. Identify the r highest values of C_d using a heap.

Figure 2: *Quit* algorithm for computing cosine using approximately L accumulators

A simple strategy for restricting the number of accumulators is to order query terms by decreasing weight, and only process terms until some designated termination condition is met. One such condition is to impose an *a priori* bound L on the number of non-zero accumulators, and to stop processing terms when this bound is reached. Such a ranking process, which we call *quit*, is illustrated in Figure 2. Other possible termination conditions would be to place a limit on the number of terms considered or on the total number of pointers decoded; or to place an upper bound on the term frequency f_t, and only process terms that appear in fewer than x% of the documents, for some predetermined value x. Buckley and Lewit [1]; Lucarella [8]; and Wong and Lee [13] have also considered various stopping rules, and derive conditions under which inverted file entries can be discarded without affecting the r top documents (although their order within the ranking may be different). Here we are prepared to allow approximate as well as exact rules, acknowledging that the cosine measure is itself a heuristic.

Quitting has the advantage of only processing the inverted file entries of a subset of the query terms, and hence of faster ranking, but at the possible expense of poor retrieval performance, depending upon how discriminating the low weighted terms are.

An alternative is to continue the processing of

1. Order the words in the query from highest weight to lowest.
2. Set $A \leftarrow \emptyset$.
3. For each word t in the query,
   (a) Retrieve I_t.
   (b) For each $\langle d, f_{d,t} \rangle$ pair in I_t,
      i. If $A_d \in A$, calculate $A_d \leftarrow A_d + w_{q,t} \cdot w_{d,t}$.
      ii. Otherwise, set $A \leftarrow A + \{A_d\}$, calculate $A_d \leftarrow w_{q,t} \cdot w_{d,t}$.
   (c) If $|A| \geq L$, go to step 4.
4. For each remaining word t in the query,
   (a) Retrieve I_t.
   (b) For each d such that $A_d \in A$, if $\langle d, f_{d,t} \rangle \in I_t$, calculate $A_d \leftarrow A_d + w_{q,t} \cdot w_{d,t}$.
5. For each document d such that $A_d \in A$, calculate $C_d \leftarrow A_d / W_d$.
6. Identify the r highest values of C_d.

Figure 3: *Continue* algorithm for using approximately L accumulators

inverted file entries after the bound on the number of accumulators is reached, but allow no new documents into the accumulator set. This *continue* algorithm is illustrated in Figure 3. The algorithm has two distinct phases. In the first phase, accumulators are added freely, as in the *quit* algorithm. In the second stage, existing accumulator values are updated but no new accumulators are added. Both *quit* and *continue* generate the same set of approximately L candidate answers, but in a different permutation, so when the top r documents are extracted from this set and returned, different retrieval effectiveness can be expected, particularly if $r \ll L$.

A similar strategy to that of *continue* is described by Harman and Candela [3, 4], although their motivation is somewhat different—they desire a small number of accumulators to reduce the sorting time required for a total ranking of the collection, whereas we assume that only a small fraction of the collection is to be presented. In this latter case, to find the top r documents a heap is the appropriate data structure, and $O(N + r \log N)$ time is required, a small fraction of query processing time.

Figure 4 shows retrieval effectiveness, measured as an 11pt average precision, as a function of $k = |A|$, the number of accumulators actually used. The small numbers beside the *continue* curve show the average number of terms processed in phase one of each query. For example, only 8.2 terms are needed to generate 27,000 accumulators. The difference between *quit* and *continue* is marked, and, perhaps surprisingly, even the mid to low weight terms appear to contribute to the effectiveness of the cosine rule. Ignoring them leads to significantly poorer retrieval.

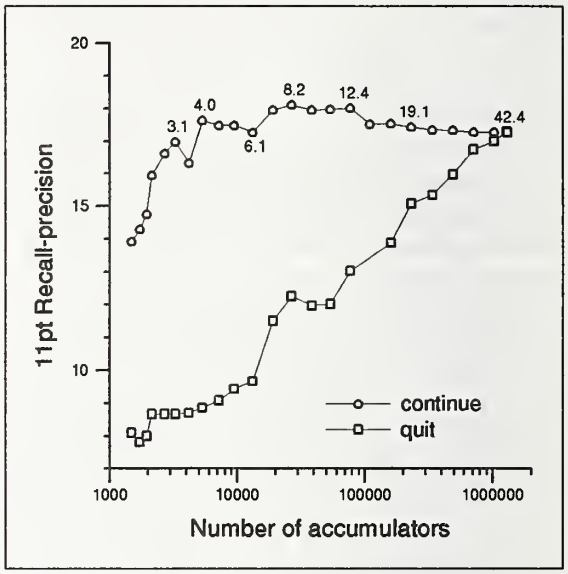


Figure 4: Effectiveness of *quit* and *continue*

In these experiments direct assessment of the relevance of pages was not possible because the relevance judgements were for whole documents. To measure effectiveness, it was assumed that if a document was relevant then each page from it was relevant, but only the most highly ranked page from each document was returned and counted. We believe this method to be fair, since one way in which pages might be used would be to rank on pages but return whole documents, with perhaps some highlighting of the top-ranked pages. Other strategies for using parts of documents to select whole documents are discussed in Section 3.

Also surprising is that the *continue* strategy, with restricted numbers of accumulators, is capable of *better* retrieval performance than the original cosine method, in which all pages are permitted accumulators. It would appear that the mid to low weight terms, while contributing to retrieval effectiveness, should not be permitted to collaborate and select documents that contain none of the more highly weighted terms.

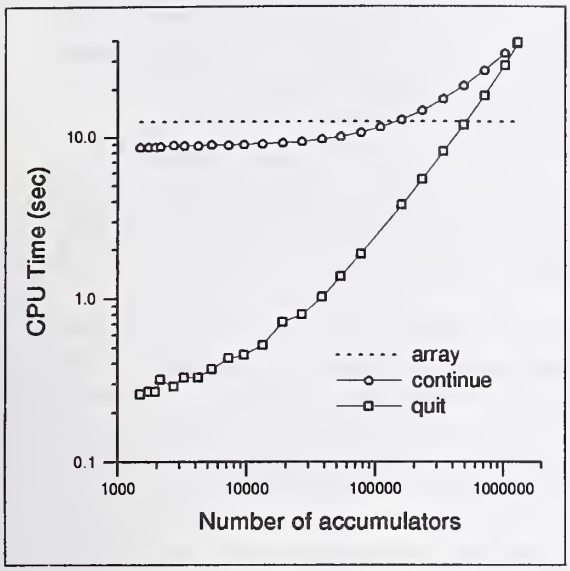


Figure 5: CPU time of *quit* and *continue*

Whilst the results for retrieval effectiveness are encouraging, the results for time are not. Figure 5 shows the cost of these two strategies in terms of cpu time, with the set A of accumulators stored as a hash table. The times shown include all processing required to identify the top $r = 200$ ranked documents, but do not include the cost of actually retrieving those documents, which, in a system such as ours, in which the text is also compressed, involves further decoding effort. All timings are for a lightly loaded Sun SPARCstation Model 512 using a single processor; programs were written in C and compiled using gcc. While for values of L less than about 100,000 the *continue* method takes no longer than the simple cosine measure when implemented using an array of accumulators, it is clearly unsatisfactory compared with the *quit* technique. In the next section we discuss methods by which this situation can be improved.

2.3 Processing long inverted file entries

Compression can reduce index size by a factor of six; for example, for the paged form of TREC the size reduces from 1,120 Mb to 184 Mb, an irresistible saving. Compression, however, prohibits random access into inverted file entries, so that the whole of each inverted file entry must be decoded, even though not every $\langle d, f_{d,t} \rangle$ pair is required. This is the reason that the *continue* strategy is slow.

The need to decompress an inverted file entry in full can be avoided by including a series of synchronisation points at which decoding can commence [10]. These can be arranged as a series of pointers, or *skips*, as illustrated in Figure 6.

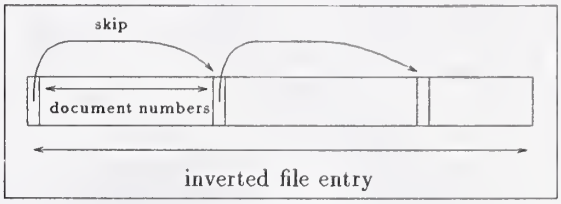


Figure 6: Adding skips

The skips divide the inverted file entry into a series of blocks, and to access a number in the entry, only the skips and the block containing the number need to be decoded. Appropriately coded, such skips increase the size of the compressed inverted file by only a few percent, but can drastically reduce the amount of decoding required. Decode time for a skipped inverted file entry is given by $T = t_d(2s + Lp/2s)$, where t_d is the time needed to decode one $\langle d, f_{d,t} \rangle$ pair, p is the number of such pairs in the inverted file entry, L is the number of accumulators, and s is the number of skips. This time is minimised at $s = \sqrt{Lp}/2$. For 10,000 accumulators and an inverted file entry of length 100,000 (a common figure for queries to the paged TREC), total time including reading from disk on a typical system drops from 0.30 seconds to 0.22 seconds [10]. Moreover, under the same assumptions an uncompressed inverted file entry of this length would take 0.30 seconds to read, and so the skipped compressed inverted file provides faster ranking than even an uncompressed inverted file.

To test skipping, inverted files were constructed for several different fixed values of L and then, for each inverted file, the cpu time for inverted file processing measured for a range of actual L values. To ensure that inverted files did not become too large, a minimum block size was imposed, requiring that every skip cover at least four $\langle d, f_{d,t} \rangle$ pairs. The results of these experiments are shown in Figure 7. As can be seen, substantially faster query processing is the result when the number of accumulators is less than about 100,000. As predicted by the analysis, the index constructed assuming that $L = 1{,}000$ gives the best performance when the number of accumulators (the variable L in Figure 3) is small.

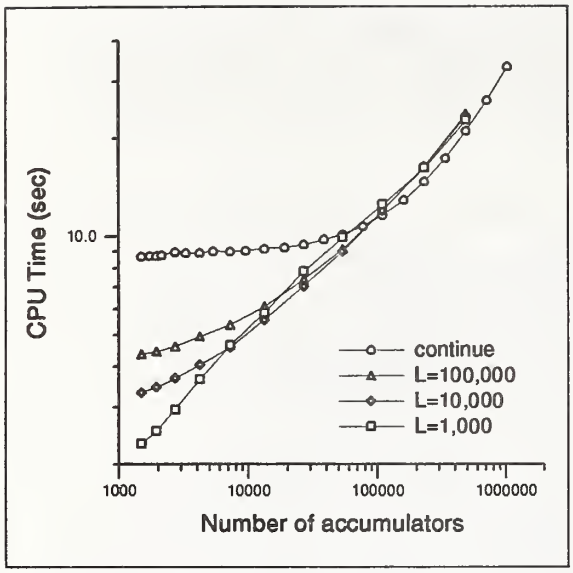


Figure 7: Time required by skipping

2.4 Retrieval experiments

The implementation techniques described above, skipping and limitation of the number of accumulators, can be applied to document retrieval as well as to page retrieval. Table 1 summarises the various resources required and performance achieved by both document and page retrieval, for various values of L, the nominal number of accumulators. The following points should be noted in connection with the data in Table 1.

File sizes Stemming was applied during inverted file construction, but no words were stopped. All alphanumeric strings were indexed, totalling 333,000,000 term occurrences of 538,244 distinct terms. The document index contained 136,000,000 $\langle d, f_{d,t} \rangle$ pointers, the paged index 196,000,000 such pairs. Each is compressed to occupy less than one byte. The variation in inverted file size is due to the insertion of skips, or, in the case of the "All" row, their absence.

The text itself was also stored compressed using a word-based model [9], allowing the 2,055 Mb to be reduced to 605 Mb. A further 27 Mb of smaller auxiliary files were also required. In total, the complete retrieval system for the paged TREC occupied about 817 Mb, or 40% of the initial unindexed text.

Retrieval time The SPARCstation 512 used for these experiments is approximately 2.2 times faster than the SPARCstation 2 used for the first round of experimentation [6, 9]. The times listed cover all activity from issue of query until a ranked list of 200 document numbers is calculated.

Fetching of the 200 answers, which occupy 226.9 Kb compressed, adds a uniform 1.9 sec per query including the cost of decompression. Decompressed, there is an average of 829.9 Kb of output text per query.

Elapsed times during the ranking process are generally about 1.6 sec greater than cpu time, in the course of which an average of 42 disk accesses are made to the lexicon; 42 disk accesses made to the inverted file itself (fetching a total of 1.65 Mb of data, containing about two million compressed $\langle d, f_{d,t} \rangle$ pairs); and 275 accesses to the combined file of document weights and addresses. All of these files except the inverted file are small, and likely to have been buffered into main memory, hence the small overhead.

Elapsed time for the presentation of documents was about 5.0 sec per query greater than cpu time, caused by the need to perform a further 200 seeks into the file containing the compressed text. This file is too large for there to have been any buffering effect.

Memory space When skipping is employed (the first three rows in each section of the table) memory space during query processing is proportional to the number of non-zero accumulators. In these experiments a hash table was used, and an average of 14 bytes per accumulator required. For the "All" experiments an array of accumulators was used, 2.8 Mb for document retrieval, and 6.6 Mb for page retrieval.

An array of 6-bit approximate document lengths was used to guide the retrieval process [11], requiring 0.5 Mb for document retrieval and 1.2 Mb for page retrieval.

Terms processed The values for "Terms processed" indicate the average number of terms processed before processing switched from the first to the second phase of *continue*. For each query a whole number of inverted file entries were processed, and this is why the "non-zero accumulators" average is greater than the target value L. Not surprisingly, with page retrieval fewer terms were re-

	Inverted file (Mb)		Retrieval Time (cpu sec)		Non-zero accumulators	
	Doc.	Page	Doc.	Page	Doc.	Page
$L = 1{,}000$	145.8	205.4	2.36	2.95	2,330	2,701
$L = 10{,}000$	156.7	220.3	4.46	5.57	12,855	13,238
$L = 100{,}000$	161.6	230.2	9.80	12.00	107,737	109,370
All (array)	128.6	184.4	7.66	12.48	582,783	1,307,115

(a) Resources required

	Terms processed		Precision at 200		Pessimal 11pt average	
	Doc.	Page	Doc.	Page	Doc.	Page
$L = 1{,}000$	3.08	2.84	0.271–0.612	0.260–0.639	0.164	0.166
$L = 10{,}000$	6.80	6.06	0.346–0.530	0.319–0.554	0.185	0.173
$L = 100{,}000$	16.84	14.26	0.333–0.446	0.326–0.504	0.175	0.175
All (array)	42.44	42.44	0.331–0.444	0.321–0.502	0.174	0.173

(b) Retrieval effectiveness

Table 1: Document vs. paragraph retrieval: (a) resources required, and (b) retrieval effectiveness

quired, on average, to consume the available accumulators.

Precision at 200 The range of values in this column is to show the fraction of unjudged documents that were accessed. The lower number assumes that all unjudged documents are irrelevant; the upper is calculated assuming that all are relevant.

11pt effectiveness The "11pt average" values are calculated based solely upon the top 200 ranked documents, assuming that all remaining relevant documents are ranked last, and that all unjudged documents are not relevant.

Index construction New algorithms have been developed for index construction. Using these algorithms, the index was built in under 4 hours, using a peak of 40 Mb of main memory and less than 50 Mb of temporary disk space above and beyond the final size of the inverted file. The compression of the documents took a further 4 hours, for a total database build time of under 8 hours.

Based upon these experiments we conclude that:

- use of a limited number of accumulators does not appear to impact retrieval effectiveness, and so is an extremely attractive heuristic for ranking large collections because of the dramatic savings in retrieval-time memory usage that result;
- introduction of skips to the compressed inverted file entries significantly reduces processing time in this restricted-accumulators environment;
- index compression can, in this way, become "free": if only partial decoding is required, the input time saved by compression can be more than enough to pay the cpu cost of fractional decoding;
- all non-stopped terms should be allowed to contribute to the ranking, and that the *quit* strategy is inferior to *continue*; and
- even relatively simple pagination gives retrieval not measurably inferior to document retrieval.

When dealing with pages, the retrieval system was implemented to display the entire document, but highlight the page or pages that had caused the document to be ranked highly. This decision made the paged collection particularly easy to use, and when we browsed the answers to the queries it was very satisfying to be able to find the exact text that had triggered the match. This advantage of pagination became clear even in the early stages of the project, while we were attempting to debug the implementation of the cosine method.

On the other hand, a problem that was brought out by these experiments was the difficulty of comparing retrieval effectiveness in

the face of non-exhaustive relevance judgements. When precision rates are around 30%, and a further (in the $L = 1{,}000$ case) 30% of documents are unjudged, there can be no significance whatsoever attached to the difference between even 30% precision and 40% precision. Indeed, assuming that 27.1% of the unjudged documents are relevant for the "$L = 1{,}000$; Doc" combination gives a final precision of 0.411; the corresponding number for the "All; Doc" pairing is only 0.373. Thus, the precision figures of Table 1 are sufficiently imprecise that no conclusion can be drawn about the appropriate value of L that should be used, and about the merits of document vs. paged retrieval. There is clearly scope for research into other methodologies for comparing retrieval mechanisms.

3 Structured documents

Many of the documents in the TREC collection are very large and have explicit structure, and it may be possible to use this structure—rather than the statistically based pagination methods described above—to break documents into parts. In particular, many documents can be broken up into a set of sections, each section having a type. There has been relatively little work done on retrieving or ranking partial documents. However, Salton et al. [12] have demonstrated that document structure can be valuable. Sometimes this structure is explicitly available [2], and sometimes it has to be discovered [5], but the knowledge of this structure has been shown to help determine the relevance of sub-documents. In this part of the work we used a small database to investigate whether retrieval of sections helped document retrieval, and whether retrieval of documents helped section retrieval. By way of a benchmark, the paged retrieval techniques described earlier were applied to the same database.

3.1 The database

Since we needed information about the relevance of sections to queries it was not possible to use the full TREC database. Instead, we used a database consisting of 4,000 documents extracted from the Federal Register collection. These documents were selected as being the 2,000 largest documents which were relevant to at least one of topics 51–100 provided for the first TREC experiment. Another 2,000 documents were randomly selected from the Federal Register collection to provide both smaller documents and non-relevant documents. The average number of words in these documents was 3,260.

These documents were then split into sections based on their internal markup. The documents had a number of tags inserted that defined an internal structure. It appeared that only the T2 and T3 tags could be reliably used to indicate a new internal fragment. Section breaks were defined to be a blank line, or a line containing only markup, followed by a T2 or a T3 tag. This led to a database of 32,737 sections. Each of these sections had a type based on its tag. The types were ⟨purpose⟩, ⟨abstract⟩, ⟨start⟩, ⟨summary⟩, ⟨title⟩, ⟨supplementary⟩, and a general category ⟨misc⟩ that included all remaining categories.

Having made the document selections, only 19 of the queries 51–100 had a relevant document in the collection. Each of the sections for documents that had been judged as relevant was judged for relevance against these queries so that finer grained retrieval experiments were possible. One difficulty that arose was that quite a few documents that had been judged relevant appeared to have no relevant sections—there were relevant key terms in the documents but the documents themselves did not appear to address the information requirement. There were 145 such (query, document) pairs. To be consistent, we took these document to be irrelevant. After these alterations, only 14 queries had a relevant section, and there were an average of 23 relevant sections per query.

3.2 Structured ranking

We carried out a set of experiments on ranking documents using the retrieval of sections. We first compared simple ranking of documents against ranking sections to find relevant documents. Next, a set of formulae were devised that attempted to use the fact that one document has several sections that might be more or less highly ranked. These took into consideration the rank of the section, the number of ranked sections, and the number of sections in the document. Experiment 3 describes one of the more successful formulas.

Further trials were then performed using the type of the section. First, a set of experiments were run that determined which types were better predictors of relevance. These results were then used to devise a measure that used a weight for each type. Finally, we tried to combine these

results to see if it was helpful to use the rank of the whole documents along with the rank of its component parts.

Experiment 1: Rank full documents against the queries using standard cosine measure.

Experiment 2: Split documents into sections. Measure similarity of each section against the queries using standard cosine measure. Order documents based on the highest ranked section.

Experiment 3: Split documents into sections. Measure similarity of each section against the queries using standard cosine measure. Order documents based on

$$\sum 0.5^{n-1} wt(s_n)$$

where s_n is the n^{th} highest ranked section in the document. The effect of this formula is that a document's weight is determined by a decay formula using the weight of all of a document's sections. (Values other than 0.5 were tried, but gave poorer performance.)

Experiment 4: Split documents into sections. Measure similarity of each section against the queries using standard cosine measure. Then weight each section using its type, for example ⟨introduction⟩ or ⟨address⟩. Order documents based on

$$\sum tw(type(s_n)) 0.5^{n-1} wt(s_n)$$

where s_n is the n^{th} highest ranked section in the document, $type(s_n)$ is the type of the section, and $tw(t_i)$ is the weight of the type t_i.

The weights of the types of sections were obtained by conducting a set of experiments where all types but one were given a weight of 1, and in turn, each type was given a weight of 2. Using these experiments it was determined that ⟨purpose⟩ and ⟨summary⟩ were each more helpful, and that ⟨misc⟩ was less helpful. As a result, in this experiment $tw(\langle\text{purpose}\rangle) = tw(\langle\text{summary}\rangle) = 2$, $tw(\langle\text{misc}\rangle) = 0.5$, and other weights were set to 1.

Experiment 5: Rank the documents, and rank the sections. Form a new rank based on the average rank of these two ranks.

Other experiments were carried out using best two sections, and formulas that more closely approximated the cosine measure. None of these experiments achieved better results than the ones displayed here. The obvious conclusion is that if documents are available for ranking as whole documents, then for this collection it is preferable to do so.

3.3 Section retrieval

For very long documents it may be desirable to return relevant sections rather than relevant documents. We were interested to see whether it might be useful to know about the rank of the containing document. In the first experiment documents were ranked, and sections shown in document order. This produced very poor results. Next, we still ranked sections in higher ranked documents ahead of lower ranked documents, but used section ranking for sections in the same document. This was reasonable but there were still many irrelevant sections being examined. Finally, we attempted to delete these irrelevant sections by using document ranking, and then section ranking, but this time discarding sections that had a section rank of greater than 200.

Experiment 6: Rank sections against the queries using standard cosine measure.

Experiment 7: Rank full documents against the queries using standard cosine measure. Order the sections by their appearance within documents.

Experiment 8: Rank full documents against the queries using standard cosine measure. Order sections, first by document, then by rank within documents.

Experiment 9: As in experiment 3, but then delete all but the 200 highest ranked sections.

These experiments show that ranking both documents and sections does help to find more relevant sections. This result is in contrast to the earlier investigation of finding relevant documents. We are able to find relevant sections much easier if the rank of both the sections, and the containing documents are taken into account.

3.4 Paged versus section retrieval

Our results showing that retrieving documents based on section ranking was not as useful as

Documents / Experiment	5	10	15	20	25	30	50	200
1	0.286	0.271	0.248	0.236	0.234	0.229	0.206	0.102
2	0.243	0.221	0.214	0.204	0.191	0.178	0.170	0.092
3	0.271	0.250	0.229	0.221	0.206	0.202	0.184	0.094
4	0.329	0.257	0.233	0.229	0.206	0.202	0.184	0.085
5	0.343	0.264	0.238	0.236	0.234	0.231	0.209	0.099

Table 2: Comparison of ranking formula for fixed number of documents returned

Documents / Experiment	5	10	15	20	25	30	50	200
6	0.186	0.164	0.181	0.161	0.160	0.152	0.140	0.120
7	0.100	0.121	0.105	0.100	0.094	0.088	0.090	0.083
8	0.171	0.121	0.114	0.100	0.091	0.083	0.100	0.082
9	0.214	0.171	0.186	0.189	0.189	0.193	0.179	NA

Table 3: Comparison of ranking formula for fixed number of sections returned

document ranking was perhaps surprising—other studies have indicated that considering smaller fragments was helpful, although in combination with larger contexts. We wondered whether the section boundaries that had been imposed were inappropriate. We thus applied the pagination techniques described in Section 2.1 to the 4,000 document database used here. These results are shown in Table 4 as Experiment 10.

These three experiments are a little difficult to interpret. Dividing documents into sections leads to poorer retrieval performance, but dividing further into pages leads to comparable retrieval performance to ranking whole documents. It may be that the manually supplied divisions are poorer than the divisions generated by automatic techniques [5]. Experiments 2–4 show that some of this performance degradation can be ameliorated by taking documents' structure into account. However, these experiments indicate that there is no retrieval advantage in breaking the document up, should the desired unit of retrieval be whole documents.

4 Conclusions

The combination of a restricted-accumulators policy and the introduction of skips to the compressed inverted file entries allows fast query evaluation on large text collections. Moreover, the ranking can be carried out within modest amounts of main memory. For example, the paged TREC collection contains 1.7 million pages, but ranked queries of 50 or more terms can be resolved within seconds using just a few megabytes of main memory. These two techniques mean that large collections can be searched on small machines without measurable degradation in retrieval effectiveness.

In the second part of the experiment we have concentrated on large documents, breaking them into smaller units for the purposes of indexing. It is not clear that users are interested in retrieving 3 Mb documents, and these experiments were designed to allow users the option of retrieving smaller parts of such documents. The results were mixed. It appears that indexing both sections and documents is helpful in ranking sections. However, it is not clear what an appropriate indexing strategy is if only full documents are to be returned. We are continuing our investigation of partial document retrieval.

Acknowledgements

We would like to thank Daniel Lam and Neil Sharman for their assistance with various components of the experiments described here. This work was supported by the Australian Research Council, the Collaborative Information Technology Research Institute, and the Centre for Intelligent Decision Systems.

GE in TREC-2: Results of a Boolean Approximation Method for Routing and Retrieval*

Paul S. Jacobs
GE Research and Development Center
Schenectady, NY 12301
psjacobs@crd.ge.com

Abstract

This report describes a few experiments aimed at producing high accuracy routing and retrieval with a simple Boolean engine. There are several motivations for this work, including: (1) using Boolean term combinations as a filter for advanced data extraction systems, (2) improving "legacy" Boolean retrieval systems by helping to automate the generation of Boolean queries, and (3) focusing on query content, rather than retrieval or ranking, as the key to system performance. The results show very high accuracy, and significant progress, using a Boolean engine for routing based on queries that are manually generated with the help of corpus data. In addition, the results of a straightforward implementation of a fully automatic ad hoc method show some promise of being able to do good automatic query construction within the context of a Boolean system.

1 Introduction

Full-text search is currently the simplest and most commonly-used method for locating information in large volumes of free text. Because users are accustomed to describing what they are looking for with specific words, and those words are often found in the texts, searching the text for selected words or word combinations is a natural and easy-to-implement method for information retrieval. However, it can be very inaccurate. It can be especially difficult for searchers to compose "queries" that combine the words that are effective in locating relevant material without finding large quantities of irrelevant information as well. One way to cope with this difficulty, while still preserving the advantages of the full-text search engine, is to help to automate the process of generating Boolean queries. This was the focus of GE's TREC-2 effort.

GE's involvement in TREC represents a relatively low level of effort aimed at bringing together natural language text processing, data extraction, and statistical corpus analysis methods. Our project uses innovative approaches for extracting information from text, best exemplified in our results in the MUC and TIPSTER extraction evaluations [7, 3] and in operational text management systems in GE. In TREC-1, we attempted to show the benefit of natural language interpretation by using Boolean approximation to select portions of text that could be further interpreted. The main result of this was that natural language seems to have very little to offer as a precision filtering method, because routing and retrieval problems stem largely from having the wrong terms in the queries [6]. Thus, in TREC-2, we have stuck with the Boolean engine, concentrating on the use of corpus analysis to improve the queries.

Figure 1 summarizes our TREC results. Our results in TREC-2, as in TREC-1, were quite good relative to other systems. The manual routing system, which comprised over 99% of our effort, produced an 11-point average of .3308, with an average of 45 relevant documents in the top 100. This put GE's system at the very top of the manual routing category (the system with the best 11-point average in this category was slightly higher on the 11-point average and had slightly fewer relevant documents, on average, in the top 100).

The residual effort went into a fully automatic ad hoc method, which produced an 11 point average of .2183 and an average of 37 relevant documents in the top 100. As in TREC-1, performance varied dramatically by topic. The routing system showed the best results (in terms of precision at 100 documents) on 8 of 50 topics. Yet it was below median on 17 topics. This not only suggests areas for further improvement, but also shows an important difference between the Boolean approach and some of the statistical retrieval systems. The Boolean approach does much better on certain topics, but the statistical approaches have more consistent performance.

*This research was sponsored in part by the Advanced Research Project Agency. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Advanced Research Project Agency or the US Government.

	AD HOC TEST		ROUTING TEST	
	11–pt. avg.	Rel.@ 100 docs.	11–pt. avg.	Rel.@ 100 docs.
Boolean '92	.2029	47.2	.2078	35.6
Pattern matcher '92	.1961	46.2	.1851	34.6
Avg. median run '92	.1585	39.7	.1246	28.6
Boolean '93	.2183*	37*	.3308	45
Avg. median run '93	.2620	41	.2910	41

*= fully automatic

Figure 1: Summary GE Results on TREC-1 and TREC-2

While it is very hard to measure progress between TREC-1 and TREC-2 because none of the numbers are directly comparable, the difference between our .2078 routing average in TREC-1 and .3308 in TREC-2 is large enough that we are quite pleased with the rate of progress and confident that we are nowhere near the peak that can be achieved with manual, Boolean routing. In addition, the automatic ad hoc method that we tried, while showing terrible performance on some of the more convoluted topic descriptions, had a better average than our manual ad hoc system last year.

Thus, there is a great deal to be gained using corpus analysis to automate or assist in query generation, even within the context of straightforward retrieval methods. In addition to having promise for "legacy" operational systems, these results suggest that natural language methods, focused on corpus analysis and query generation, probably can help in improving the performance of many information retrieval systems.

2 Boolean Approximation

The basic approach in our system is to compile queries into Boolean tables that can be matched at high speed against a stream of input text. This approach is meant for routing, and also to be compatible with "downstream" analysis such as what we do in TIPSTER data extraction. In fact, the Boolean compiler we use is designed for handling the much more complex expressions that our system uses in data extraction.

Figure 2 illustrates the approach. We call this *Boolean approximation* because the Boolean expressions used in the basic matching engine are an approximation to more detailed processing of texts, in the sense that they are guaranteed to admit all text that would be admitted by more detailed processing, but will usually also admit many texts that would be rejected by more detailed constraints. This is a very general method, in that the system can be configured to apply many different stages of analysis, from "shallower" processing to "deeper" interpretation, with each stage applying stricter constraints—for example, word order, proximity, semantic constraints, and so forth. Furthermore, at each stage, the effects of filtering can be measured, generally showing a loss of recall and gain in precision. In TREC-1 [6], we measured this tradeoff and found that the highest 11-point averages came from the first stage of filtering; in other words, the gains in precision in later stages were not enough to make up for the loss of recall on these measures.

The figure also illustrates the flow of information at development time and the sort of knowledge that applies at each stage. For example, at development time, knowledge can be mapped from the deeper levels into shallower levels. At run time, the subsequent stages of language analysis apply this knowledge in stages. For example, our system can analyze joint venture texts (in English and Japanese), looking for, among other things, information about the joint venture company by recognizing that the company is often the object of the verb *establish*. In the Boolean stage, this can be approximated by looking for the combination of words like *establish* with words like *venture*. In the finite state pattern matching stage, the system might look for any word with the root *establish* followed by the venture term (and perhaps the reverse order with the verb in a passive form). In deeper interpretation, the system applies syntactic and semantic constraints to recognize the different ways that the con-

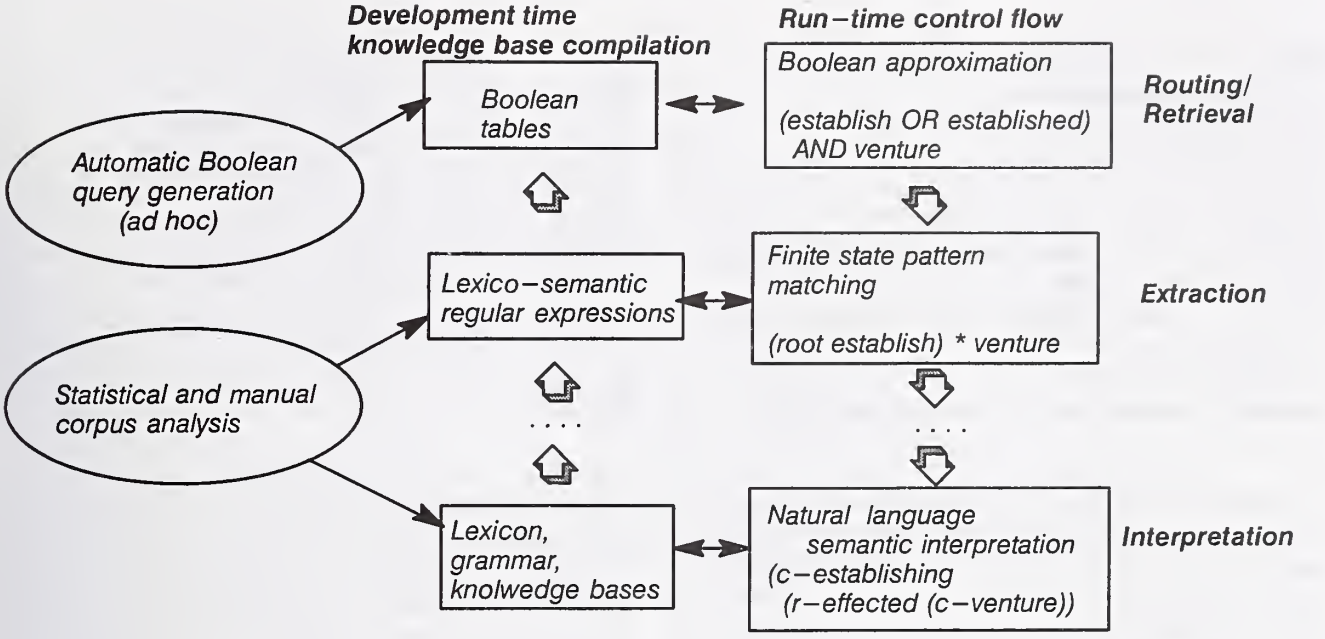


Figure 2: Approximation and Natural Language Processing

cept of establishing can be expressed, and insure that the words appear in a grammatically acceptable way in the input. This more detailed analysis can be crucial in data extraction tasks.

A critical point about this model is that even detailed knowledge about language often ends up contributing to the Boolean tables, so the Boolean expressions are considerable more complex than those that could easily be created by hand. Furthermore, the Boolean expressions are a relaxed form of more complex knowledge-based constructs. It is very difficult to predict the impact of simply relaxing constraints. For example, TREC topic 53 includes the phrase "leveraged buy-out". In a strict syntactic analysis, this phrase would have to appear exactly in that form, enforcing both proximity ("leveraged" adjacent to "buy-out") and order ("leveraged" before "buy-out"). But relaxing such constraints seldom has any severe effect on results: the Boolean expression (leveraged AND buy-out) in our system will recognize the two terms occurring anywhere in the same paragraph, which is actually likely to admit *more* texts about leveraged buy-outs than admit texts in which the words coincidentally appear together.

Even the phrase "prime rate", which matches some irrelevant texts in its Boolean form (including, for example, one or two texts about Japan that mention "prime minister" in the context of "growth rate"), also admits some additional relevant texts in which "prime" appears in the same paragraph as "rate". The well-known effects of word order and proximity on meaning, exemplified by the distinction between "blind Venetian" and "Venetian blind" do not seem to appear very frequently in real examples. At least, these effects may be less frequent

In TREC-2, we did not apply *any* constraints stricter than Booleans at run-time; in other words, we used only a Boolean retrieval engine (because in TREC-1 we proved that the stricter constraints didn't help). We also had to implement a module to relax some of the "hard" Boolean constraints for topics with a very small number of relevant documents, because of the nature of the performance metrics. However, we still used the knowledge that was developed for more detailed processing, including the phrases, semantic groupings and so forth. In addition, we added a more sophisticated ranking mechanism than we used in TREC-1, because ranking is very important in the evaluation. But the only retrieval engine in our TREC-2 system is a Boolean matcher.

2.1 The Boolean Matcher

The Boolean tables are efficiently organized so that a C program (which we now know as NLGREP) can match them against incoming texts at a rate of about 1 million words per minute. This program spends little time on anything other than marking where words in each document match terms in the table. In both routing and retrieval, the total number of terms used for a set of 50 topics is about 2000, or about 40 unique terms per topic. All other words are ignored entirely.

For example, the following is a query for the topic "South African sanctions" (using the enhanced regular expression language of GE's pattern matcher [5]):

```
sanction == [(member sanction sanctions
          disinvestment)
     <Sullivan Principles>
     <punitive *2 measures>] ;

safrica == [(member Buthelezi Pretoria
                   anti-apartheid apartheid)
          <De Klerk>
          <South (member Africa African)> ] ;

;;; rule 1
$sanction * $safrica => (mark-topic 52) ;
```

This description says that any matching text must have *both* an indicator of South Africa (**$safrica**) and one of sanctions (**$sanction**), and that the sanction phrase and South Africa phrase must appear in the same paragraph in the document.

A sanction phrase can be any of the simple words *sanction*, *sanctions*, or *disinvestment*, or any phrase including *punitive measures* with no more than two intervening words (like *punitive economic measures*). A South Africa phrase can also be either one of a group of simple words, or a phrase, like *De Klerk*, *South Africa*, or *South African*.

These queries or topic descriptions can be quite complex, and the method has been designed to handle many queries simultaneously, so the rule compiler is designed to produce expressions that can be efficiently applied within a large set of queries. This is important because many queries can share the same simple terms or combinations of terms, and because the Boolean matcher must match the simplest expressions first.

For the topic description given above, the output of the rule compiler will include the following tests:

```
52          TERM     AFRICAN
 ...
2029        TERM     MEASURES
 ...
2134        TERM     SANCTION
2135        TERM     SANCTIONS
2136        TERM     DISINVESTMENT
2138        TERM     SULLIVAN
2139        TERM     PRINCIPLES
2141        TERM     PUNITIVE
2144        TERM     BUTHELEZI
2145        TERM     PRETORIA
2146        TERM     ANTI-APARTHEID
2147        TERM     APARTHEID
2149        TERM     DE
2150        TERM     KLERK
2152        TERM     SOUTH
2153        TERM     AFRICA
 ...
2137        OR       2134 2135 2136
2140        AND      2138 2139
2142        AND      2141 2029
2143        OR       2137 2140 2142
2148        OR       2144 2145 2146 2147
2151        AND      2149 2150
2154        OR       2153 52
2155        AND      2152 2154
2156        OR       2148 2151 2155
2157        AND      2143 2156
2158        AND      2156 2143
 ...
TOPIC052    OR        2157 2158
```

Each line in the above data gives a unique number (or topic designator) to the test, a test identifier (either TERM for a simple word test, OR, or AND), and a list of simple terms or previous tests. For example, test 2137 depends on tests 2134, 2135, and 2136, and is true if any of those tests is true, namely, if the text includes any of the words *sanction*, *sanctions*, or *disinvestment*. The tests are automatically ordered so that all tests that are dependent on other tests will have higher numbers than the tests they depend on; thus all TERM tests appear first. In this case, the TERM test AFRICAN appears with a much lower number simply because it is used in many different queries.

The matcher, which can work either on complete documents or paragraphs (but we used paragraph matching only in TREC-2) goes through every word in its input and, using a fast table look-up, sets the TERM tests to true for every word it encounters. At the end of input, either the end of the paragraph or end of each document, it runs through the table of possible tests from low numbers to high numbers and sets tests to true if their conditions are satisfied. A topic test produces a match if it has become true at the end of this process, meaning that the paragraph or document has passed the pre-filter for that query. A single paragraph, of course, can satisfy multiple queries.

This portion of the system was implemented in the space of a few days, and is almost entirely the same as in TREC-1. Our focus since last year has been on query construction and ranking rather than matching or retrieval.

2.2 Query construction

Our approach assumes, in general, that manual query construction is acceptable for routing. In ad hoc retrieval, query time can be of the essence, but in many routing applications, queries are developed and refined over time. The amount of time spent on query construction using a manual method in our system is comparable to the amount of time spent on the topic descriptions used for automatic query generation.

2.2.1 "Manual" queries for routing

In manual routing, our approach uses a statistical corpus analysis, developed originally for text categorization [4], to pull out terms based on their relative frequency in relevant documents for each topic. The statistic used combines the entropy-based mutual information statistic (testing the independence of each term with each topic) with a correction for low-frequency terms and for ambiguous words. Words with high weights have a high degree of association with a topic. This statistical analysis is also used in ranking. The base weighting formula is the following:

$$C(\log_2 b)(\log_2 r)$$

where C is a constant, b is number of times a term appears in a story assigned to a particular category, for example, and $\log_2 r$ is the log of the ratio of combined probabilities (i.e., of a particular word or phrase occurring in a text about a particular category) to the product of independent probabilities—the mutual information statistic. This tests the assumption that the use of the word and the category of the text are independent. When this assumption is false, the word gets a high positive or negative weight.

For example, the following are the top words for Topic 51, "Airbus subsidies":

```
A-330                TOPIC51        1263.3
AIRBUS               TOPIC51        1183.1
A-340                TOPIC51        1178.2
INDUSTRIE            TOPIC51        1071.9
A-320                TOPIC51        1067.5
MESSERSCHMITT-BOELKOW-BLOHM TOPIC51 851.3
AERONAUTICAS         TOPIC51        843.3
CONSTRUCCIONES       TOPIC51        807.9
AEROSPATIALE         TOPIC51        762.5
MBB                  TOPIC51        722.6
WIDE-BODY            TOPIC51        617.7
MD-11                TOPIC51        613.8
TOULOUSE             TOPIC51        228.5
JETLINERS            TOPIC51        217.8
LUFTHANSA            TOPIC51        196.6
MD-80                TOPIC51        196.6
```

Clearly, these words all have some reason to be associated with this topic, but adding them to the appropriate group in each query (or ignoring them entirely) is a "manual" process. Our manual routing queries, therefore, are a combination of the regular expressions that were developed from the topic descriptions with terms added that were selected from the automatic training. This is, we believe, a very practical manual approach that has very good performance.

2.2.2 "Hard" vs. "Soft" Booleans

The Boolean matcher uses a "hard" Boolean approach, in that it will admit only texts, for each query, that satisfy the conditions of that query. For example, in Topic 51 above, "Airbus subsidies", the matcher will allow texts only that have *both* and Airbus term *and* a subsidy term in the same paragraph. However, this is a narrow topic, and TREC-2 allows each system to produce 1000 texts for each topic. The evaluation metrics offer no penalty for filling up the list of 1000 with texts that are likely to be irrelevant. So, in order to provide increased flexibility and consider larger numbers of texts for each topic, we used an additional engine only for the purpose of pulling in texts for very specific queries like this one.

The system is still a hard Boolean system in that texts that satisfy the Boolean conditions will always be ranked higher than texts that do not satisfy the conditions; however, texts that do not satisfy the conditions can appear on the final ranked lists. The "soft" engine considers such texts by relaxing some of the Boolean conditions, effectively pulling in texts that have a large number of terms that match the query, but do not necessarily meet all the conditions. This component of the system is more like statistical retrieval engines; however, it does not have a large impact on the overall scores, because it only affects the results at the low-precision extreme (the lowest rankings) for queries that match very few documents. In fact, for Topic 51, the hard Boolean query matches 11 texts, and the soft method pulls in an additional 989 texts. But there are only 11 texts that are judged relevant, of which 10 satisfy the hard Boolean. So enforcing the hard Boolean condition seems to work well for this topic, and the soft Boolean doesn't contribute much.

For some topics, the balance between the hard Boolean condition and the soft Boolean isn't so clear; i.e. not enforcing the hard condition would lead to better ranking. This seems be a function of how well the topics fit with Boolean expressions in general. "Airbus subsidies" is really a Boolean topic, in that a relevant text must say something about Airbus and something about subsidies. Other topics like "automation" are much harder to express in a Boolean form. We will cover these issues in topic-by-topic performance later in this paper.

2.2.3 "Automatic" queries for ad hoc retrieval

The "manual" query method is partly automated in the sense that the corpus-based statistical training suggests many of the terms that are used in the queries. But it is "manual" in that the initial formulation of the queries is done manually from the TREC topic descriptions. We have tried, as a simple experiment, to generate the Boolean term groupings and expand each term automatically from the topic descriptions. In the days before the TREC-2 ad hoc test, we tried several different ways of do-

ing this automatic Boolean query generation, and chose the one that worked best on the sample data.

Our first attempt was to use the common methods for finding collocations and word associations in sentences, and these worked horribly for term expansion. The problem is that this approach finds more associations like "funeral" and "home" than it does "hostage" and "captive", and the latter, text-level associations are what's required to generate good queries.

The "solution" we tried was, for a sample of about 10 million words in the corpus, to choose the top 20 words based on TF.IDF weights for each document, store the frequency of association among these terms, and then weight each pair using the weighted mutual information statistic of the previous section. This was much better than using sentence-level information, although it is still a very straightforward approach. For example, the following are the top 10 terms associated with the word "hostage" (in order):

```
hostages
Lebanon
Beirut
Iran
release
Terry
kidnappers
kidnapped
Jihad
Anderson
```

While this is certainly not the optimal set of terms to use in place of "hostage", it is a good start.

The next problem in automatic query construction is when to use a combination of terms and when to use a single term. For example, the term "weather-related fatalities" is a combination of two word groups (weather and fatalities) while "Iran-contra affair" is really only one group (Iran-contra), even though it might appear that "affair" is a significant term.

Again we took the direct approach, choosing to combine terms whenever there was a reasonable percentage of overlap between their associated terms. This worked surprisingly well in cases where the topic title was a good description (e.g. "welfare reform") and very badly for those with vague titles (e.g. "find innovative companies"). We tried to recover from these by including more words from the description and narrative, but then we had to start recognizing the language of these descriptions, filtering out words like "relevant", "mention" and so forth. At this crude stage, the main problem with the query generation method is using the structure of the topic descriptions.

The second major issue with automatic query generation is that it isn't nearly as good at finding good terms as the process of training from data and relevance judgements, as used in the routing experiments. The relevance judgements used for routing contain large volumes of relatively high-accuracy data, while the training used for term expansion in query generation relied on relatively small volumes of relatively noisy data. For example, the word "welfare" used in one of the ad hoc topics occurred with a high enough TF.IDF weight only 29 times in the training sample, and the most frequently associated term, "children", occurred only 6 times. In order to establish good associations between "welfare" and less frequent terms, we would need much more data. The data from TREC-2 seem to suggest that low-frequency terms contribute more in term expansion than high-frequency terms, so using a "small" training sample (10 million words is only about 3% of the corpus) was a major error. We made many other mistakes in the training method, including mixing samples from the *Federal Register* and DOE sources with other texts that are much more likely to be relevant. This leaves a lot of room for future experiments and improvement.

The fully automatic ad hoc system certainly didn't do as well as the manual routing system, but it was still at or above median for more than half of the ad hoc topics. Considering that this method could be used within the context of most any legacy retrieval system, the result is worth noting. Furthermore, the generation of Boolean queries from natural language descriptions is an interesting, as well as practical, research problem, because many different retrieval systems can make some use of Boolean queries.

3 Ranking

In both routing and ad hoc, we used a set of word weights for ranking, acquired using the relevance judgements in the routing case and from the corpus data in the ad hoc case. In routing, the weights reflect the statistical measure of association between the term and each topic (using the weighted mutual information score given earlier). In the ad hoc case, the weight is a function of the frequency of the term in the topic description, the inverse collection frequency, and an additional factor to weight certain components of the topic descriptions (such as the title and description) more heavily than others. We combined the weighted frequency of these terms with an overall count of the number of topic hits per document, normalizing for document length, to produce a score for each document. This was the result of trying many different approaches on the test data, so it was definitely a good method for our system.

However, in comparing our results with those of other systems, our precision curve across various recall points is not nearly as good as a system that does really good ranking. In routing, we are not sure that ranking is important, but it is certainly important in getting good results in TREC. So, we are inclined to try to combine our

retrieval method with alternative ranking methods to see, for example, whether more terms are really necessary in order to get better ranking results.

The separation of retrieval and ranking seems to be a valuable tool both for experimental research and for identifying different techniques for applications. It is clearly a problem with both TREC-1 and TREC-2 that the routing task requires a comparison of documents across a large collection, when most routing applications deal with a stream of documents individually or in small groups.

4 Analysis of Results

The results raise a number of important issues, especially: why Boolean approximation works as well as it does, particularly why it works for routing; where statistical weighting could help more; what sort of topics this approach does well on (and which topics it does badly on); and other obvious areas for improvement.

One of the most important sources of information about the advantages and disadvantages of each approach comes from comparing the performance of different systems on different topics. Unfortunately, this is also a very difficult task, because, while it is easy to tell which systems did well on which topic, it is often hard to generalize from that evidence *why* the approach worked or why it didn't.

As we have mentioned, the Boolean approach is very erratic with respect to performance by topic, as compared with other systems, particularly the statistical methods that emphasize weighting. For example, our manual routing system, which was clearly one of the best systems, had the top performance (in precision at 100 documents) on 8 topics, but was below median on 17 topics (out of 50). In the 11-point averages, that system was below median on 22 topics—more than 40% of the time—although it outperformed most of the systems on average. By contrast, one of the Cornell systems [1] was above median on every topic! This suggests that our approach degrades less gracefully than other approaches, and that it is important to explore Boolean methods as an adjunct to other methods that work in the cases where the Boolean approach seems to fail. Conversely, our system had top or near-top scores on a significant number of topics; it is important to know how to take advantage of this within the context of weighting systems.

There seem to be several different explanations for variation on the topics. First, there are topics, as we have discussed, that are particularly well suited to Boolean methods (and others that are not well suited at all). Second, there are cases where the training method seems to work particularly well. Third, there are cases where the manual approach might work well because there are terms in the topic description that are particularly misleading. Finally, there are many reasons why our approach can fail, particularly on topics with very small numbers of relevant documents and in cases where the topics are very vaguely specified.

One of the topics where GE had the best results was Topic 53, "leveraged buy-outs". The topic description specified that relevant documents had to describe an LBO above $100 million in value, and give the terms of the buy-out. Apparently, the $100 million figure is not important, because most of the LBO's that are reported are major buy-outs. However, the terms (the specification of the dollar amount) are required. Many articles about LBO's do not report dollar amounts. This is similar to the "Airbus subsidies" topic, where many articles that talk about Airbus do not mention subsidies, and they are not relevant. The advantage here seems to be that the hard Boolean outperforms the weighting approaches because weighting, without Booleans, is likely to give an article with many LBO words, but no dollar figures, a high weight, just as it could a high weight to an article about Airbus that doesn't mention subsidies.

The effect of training seems to help in the "leveraged buy-out" case as well. The training picked up many names of companies involved in buy-outs, like "Safeway" and "Dart", and these were included in the queries. This perhaps helped to separate articles about specific buy-outs from buy-outs in general. A similar effect came about on Topic 92, "international military equipment sales", where the training pulled in names of many of the weapons typically sold on the international market.

Topic 86, "bank failures" was another topic where the GE system outperformed all others on both the 11-point average and precision at 100 documents. This result is hard to explain, but the one conspicuous fact about the topic is that our query does not include the word "bank". It does include the names of many prominent banks, so it may be, like the LBO case, that good performance on this topic depends mainly on distinguishing specific references to failures from general discussions about bank failures, for example, the S&L crisis.

On the topics that have very few relevant documents, our approach often failed because, in the absence of training data, it tended to undergenerate; thus very few texts (sometimes none) would match the Boolean query and other systems with good weighting would pull in more relevant documents. In these cases, a system that finds one relevant document scores much better than a system that finds zero, so the penalty for undergeneration is very high.

The second class of topics where we seem to go wrong is in those that are vaguely specified. For example, Topic 74, "policy conflict" is a very hard topic, where the description does not include very much information. Texts rarely mention policy conflict, and, when they do, they are rarely relevant. On the other hand, texts about tobacco policies and health are likely to be relevant. This

is probably a case where there is no point in having a Boolean query, and probably systems with the best training and weighting methods do best.

Another important point about the TREC-2 results is that the best automatic systems did significantly better than the best manual systems in the routing task, probably because of the success of the training methods used. This raises an important question with respect to the relative contributions of time spent on query construction versus time spent assembling training data. The volume of training data used in TREC-2, with hundreds of thousands of relevance judgements, is not realistic for most routing scenarios. Thus, while we should continue to rely on good training methods, we should be careful to separate out the effects of training and to develop manual routing methods that work with smaller amounts of training data.

Given that Boolean and manual methods seem to do best on certain topics, and other approaches that emphasize weighting do better on others, it makes sense to combine the best from different approaches. However, this raises the issue of how different results can be incorporated into our model without losing the advantages of the Boolean method, particularly the compatibility with so many existing systems.

5 Future Goals

Many of our results from TREC-2 suggest areas where the method of generating Boolean queries, especially using corpus data, can be substantially improved. There is a great deal of room for future progress, so we believe that this approach will continue to be viable with respect to other routing and retrieval methods.

The main area for research is in continuing to explore new corpus analysis methods. Our corpus analyzer weights single terms. But the Boolean queries depend on combinations of terms, not only in the case of phrases but also to control the effect of ambiguous words. In the context of a given query, a single term can often be roughly comparable to the Boolean AND of two or more other terms. Up to this point, we have not quite been able to automate the process of discovering these relationships in the corpus. This is important for both routing and ad hoc retrieval, but especially for routing.

Both the routing and ad hoc systems can benefit from the use of new ranking methods, and possibly from exploring hybrid approaches that take advantage of the Boolean method on topics that are well suited to Boolean expression and degrade more gracefully to traditional weighting methods on other topics. In general, the combination of methods is something that merits new experiments.

The routing system could benefit from new training methods. Because the Cornell system did especially well using a training method that produced large numbers of terms and used both positive and negative information, it is possible that this general approach could help our system as well.

The ad hoc system suffers mostly from difficulties in handling the topic descriptions; our method of deriving Boolean expressions from the topic descriptions is still extremely crude, and there are many topics for which the approach produced almost useless queries. The performance of the automatic ad hoc system was even more erratic than that of the manual routing system, but it is very likely that many of the problems can be solved with a lot more work on processing the topic descriptions. Although we are loath to direct research at issues that are particular to the formulation of the TREC topics, this work may be necessary to determine the real power of automatically generating Boolean queries.

Finally, we are interested in exploring many ways that our corpus analysis and query generation component can be combined with other systems. Because we have focused our attention on query content rather than ranking or retrieval models, we believe that our results could quite likely be used within many other retrieval systems. It is natural to look for such synergy. We have had some preliminary collaboration with the UMass team to try to use our queries within the INQUERY system [2], but we still have a long way to go. We will continue to explore such collaborative efforts and to concentrate our own efforts on corpus analysis and building queries.

6 Summary

GE's participation in TREC involved the implementation of a number of strategies for creating Boolean queries from the topic descriptions. A statistical corpus analyzer helped to refine queries for both the routing task, and to generate them automatically for the ad hoc task. The simple Boolean retrieval engine performed well, especially in routing. As before, there is tremendous variation in the topic-by-topic results, suggesting that a great deal more research is needed to find how to get the best results in different routing and retrieval scenarios. We are encouraged by the progress of our system, as well as of the overall field, in these experiments, and are hopeful that in the coming years we will learn how to combine our promising results in Boolean approximation and corpus analysis with the more mature ranking and retrieval models of some of the other systems.

References

[1] C. Buckley, J. Allan, and G. Salton. Automatic routing and ad-hoc retrieval using SMART: TREC-2. In Donna Harman, editor, *Proceedings of the Second Text Retrieval Conference (TREC-2)*, Gaithersburg, MD, 1993.

[2] W. Bruce Croft. The University of Massachusetts TIPSTER project. In Donna Harman, editor, *Proceedings of the Second Text Retrieval Conference (TREC-2)*, Gaithersburg, MD, 1993.

[3] P. Jacobs, G. Krupka, L. Rau, M. Mauldin, T. Mitamura, T. Kitani, I. Sider, and L. Childs. The TIPSTER/SHOGUN project. In *Proceedings of the TIPSTER Phase I Final Meeting*, September 1993.

[4] Paul S. Jacobs. Using statistical methods to improve knowledge-based news categorization. *IEEE Expert*, 8(2):13–24, April 1993.

[5] Paul S. Jacobs, George R. Krupka, and Lisa F. Rau. Lexico-semantic pattern matching as a companion to parsing in text understanding. In *Fourth DARPA Speech and Natural Language Workshop*, pages 337–342, San Mateo, CA, February 1991. Morgan-Kaufmann.

[6] George R. Krupka Paul S. Jacobs and Lisa F. Rau. A Boolean approximation method for query construction and topic assignment in TREC. In *Second Annual Symposium on Document Analysis and Information Retrieval*, May 1993.

[7] Lisa F. Rau, George R. Krupka, and Paul S. Jacobs. GE NLToolset: MUC-4 test results and analysis. In *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, San Mateo, CA, June 1992. Morgan Kaufmann Publishers.

TREC-II Routing Experiments with the TRW/Paracel Fast Data Finder

Matt Mettler
TRW Systems Development Division
Redondo Beach, CA

Fritz Nordby
Paracel Inc.
Pasadena, CA

1.0 Introduction

For TREC-II, we were interested in experimenting with improved methods of constructing queries for the Fast Data Finder (FDF) text search coprocessor. We learned from TREC-I that while the pattern matching ability of the FDF can sometimes be put to significant advantage (we had the high score on 8 of the 50 routing topics in TREC-I), this wasn't sufficient overall to overcome the weaknesses traditionally associated with the boolean approach to text retrieval. Many of the TREC topics are too abstract and ambiguous to respond well to a boolean query formulation.

Our goal for this year therefore, was to apply the FDF hardware to a more statistical or soft boolean retrieval approach while not giving up on our ability to make use of specific features or patterns in the text when they are obviously important.

We experimented with two different schemes. In the first scheme, we utilized subquery proximity to rank hit documents. We developed the subqueries manually, then determined the optimum proximity values by test runs on the training data. The most effective values were then used in the official routing queries. The second scheme was an FDF adaptation of the traditional Information Retrieval (IR) term weighting approach. In addition to single word terms, we also included two and three word phrases, and FDF subqueries designed to detect special features in the text.

While in the terminology of TREC both are examples of manual query formulation with feedback, we believe these techniques can be evolved to create queries automatically from samples of relevant text and to also incorporate user knowledge of specific text features of interest when it exists. We also continue to believe that the utilization of a hardware accelerator such as the Fast Data Finder, enables the implementation of high performance routing or dissemination applications at a far lower cost than can be achieved with conventional general purpose processors.

2.0 The FDF Text Retrieval Approach

The Fast Data Finder is a hardware device that performs high-speed pattern matching on a stream of 8-bit data. It consists of an array of identical programmable text processing cells connected in series to form a pipeline processor. The cells are implemented using a custom VLSI chip designed and patented by TRW. In the latest implementation, each chip contains 24 processor cells and a typical system will have 3,600 cells. Each cell can match a single character of query or perform all or part of a logical operation. The processors are interconnected with an 8-bit data path and approximately 20-bit control path. To perform a search, a microcode program is first downloaded into the pipeline to direct each processor. The database is then streamed through the pipeline. The data bytes clock through each processor in turn until the whole database has passed through all processors. As the data is clocking through, the processors alter the state of the control lines depending on their program and the data stream values.

When the pipeline's processor cells detect that a series of database characters match the desired pattern, a hit is indicated and passed by external circuitry back to the memory of the host processor and to the user. The FDF pipeline runs at a constant speed as it performs character comparisons and logical operations, regardless of query complexity.

The queries or patterns are specified in the FDF's Pattern Specification Language (PSL). The hardware directly supports all the features in the PSL query language without the need for software post-processing. The processors in the pipeline may all be used to evaluate a single large query or may be assigned to evaluate numerous smaller queries. The number of pipeline cells a query needs is proportional to the size of the query. PSL provides numerous search functions, which may be nested in any combination, including:

- Boolean logic including negative conditions
- Proximity on any arbitrary pattern
- Wildcards and "don't cares" anywhere in the word
- Character alternation
- Term counting, thresholds, and sets
- Error tolerance (fuzzy matching)
- Term weighting
- Numeric ranges

The Fast Data Finder was originally designed and developed at TRW. In 1992, TRW licensed the FDF technology to Paracel Inc., which now sells a commercial product called the FDF-3.

3.0 Proximity Query Generation

Our first set of experiments revolved around the use of subquery proximity to rank hit documents. We began with a simple observation: topics are often a conjunction of ideas or concepts. For example, Topic 51, Airbus Subsidies, is a conjunction of the idea "Airbus" (a particular aircraft manufacturer and European consortium) and the idea "subsidy" (in particular, subsidies from the nations belonging to that consortium). Other articles about Airbus Industrie (new planes, fly-by-wire in the A320, accident reports, financial health

reports, etc.) are not relevant to this topic; nor are articles which describe subsidies not directed toward Airbus.

Traditional IR term weighting techniques do not give any explicit benefit to articles which conjoin ideas. Articles which include terms relevant to each of the component sub-topics will receive high scores; but so will articles which include many terms relevant to only one sub-topic. Recent efforts implicitly include conjunctions through the use of phrases as terms in otherwise traditional statistical methods.

An alternative is the use of boolean operators. This has the desired effect -- an AND of terms forces a conjunction -- but the use of booleans in IR has been viewed with some skepticism and disfavor. Boolean operators often find a conjunction of terms where none truly exists (for example, Airbus and subsidies might be mentioned in two separate and unrelated portions of an article); or, if made sufficiently restrictive to eliminate spurious matches, boolean-based searches often miss relevant articles.

We have followed an approach which incorporates both ideas. Rather than focus on specific phrases, we search for terms in proximity to one another. The terms in the query are chosen to represent each of the constituent sub-topics, just as in a boolean search. The specificity of the query is adjusted by varying the required proximity of the terms. Thus, for Airbus subsidies we might search for terms representing "Airbus" in a range of proximities to terms representing "subsidies".

This approach allows conjunctions to be graded. A small proximity restriction (say, 3 words) yields results similar to a keyphrase search, indicating that the two concepts are indeed associated in the article and that the article is relevant to the topic. A large proximity restriction (1 article) is analogous to a simple boolean keyword search and retreives articles in which the concept terms may be only loosely associated. Intermediate proximities (1 sentence, 1 paragraph, etc.) indicate intermediate degrees of association and intermediate recall/precision trade-offs.

It is also possible to use multiple proximities in a single query with this method, or to use proximities and occurrence frequencies together, to form multi-dimensional arrays of query parameters. For example, for Topic 62, Military Coups D'etat, the number of conjunctions was traded off against the proximity of the conjunction to form a two-dimensional query set.

For the initial experiment, lists of synonyms representative of each idea in a topic were manually built, and one- or two-dimensional query sets were built from these lists. These queries were then run against the training database, and after some feedback, the query sets were finalized. Each finalized query set was run against the training database to determine a ranking of the queries based solely on selectivity.

Table I shows a sample proximity query and Table III shows our TREC-II results. The number of relevant documents retrieved by the proximity method queries are labeled TRW1.

4.0 Statistical Query Generation

Our second set of experiments revolved around the use of term weighting. Our basic approach was to follow the well researched path of generating term weights proportional to the occurance of words in a sample of relevant text and inversely proportional to the occurance of words in the database as a whole. Since we were doing the routing topics, we gathered statistics on Volume II and hoped that they would be valid for Volume III. For sample relevant documents we used the NIST TREC-I relevant documents from Volume II. We did not use the topic narratives or descriptions. In addition to single word terms, we also considered two and three word phrases. We used the FDF itself to scan the training corpus and determine the phrase frequencies of interest.

To adapt this standard approach to the FDF, we needed to make three algorithmic modifications. First, we needed to adapt to the limitations imposed by the FDF hardware. While extremely effective for pattern matching, the FDF is not a general purpose computer. While the FDF processor cells can perform basic addition/subtraction, the datapath available to accumulate an aggregate score for a document is limited to 8 or 9 bits. Thus we had to restrict the term weights (and the range of their sums) to integer values between 0 and 255 or 0 and 511. This had the effect of truncating most topic's query terms at 10-20 terms (words, phrases, or special features). We also excluded terms from our queries that did not appear in at least 30% of the relevant sample documents.

Second, we were striving to not give up the strengths of the FDF's pattern matching capabilities to pinpoint special features in the text which have a large impact on document relevance. We manually reviewed the topics and prepared special feature subqueries in an attempt to increase the precision for particular topics. For Topic 59, Weather Related Fatalities, we manually prepared a special feature subquery to detect phrases detailing a numeric value of people killed. We determined the frequency of each special feature, both in the sample relevant documents and in the training database as a whole, and just added these into the word list as if they were regular single word terms. In some instances our manually prepared subqueries jumped to the top of the list of statistically relevant terms for a topic; in others they didn't.

Third, we observed that some topics had particular words, phrases, or special features that were present in almost all relevant documents. We converted terms that occurred in > 90% of the relevant documents to boolean ANDs in our queries. This was intended to improve precision for topics like 62, Military Coups D'etats. The topic narrative specifically stated that the country involved must be named. One of our special feature subqueries was a list of known foreign country names. While of no statistical significance as a term, this subquery did hit on almost every sample document. We thus ANDed it into the query as a required boolean term.

Table II shows a sample statistical query. DocCount is the number of documents in the sample that included the term, phrase, or special feature. DbCount is the number of documents in our training sample that included the term. Weight is DocCount divided by DbCount. PslWeight is the integer coefficient based on the Weight. The relevant documents retrieved by the statistically generated queries are labeled TRW2 in Table III.

TABLE I - Sample Proximity Query

Query Template for Topic 75, Automation

```
/* Concept 1: automation */
define x_auto 'automat[e|ion|ed|es|ing]' end

/* Concept 2: change in economics */
define x_up_dn  '[increas|decreas][e|es|ed|ing]'
      |'rais[e|es|ed|ing]'          |'lower[|s|ed|ing]'
      |'ris[e|es|ing]'              |'fall[s|ing]'
      |'drop[|s|ped]'               |'hike[|s|d]'
      |'dip[|s|ping]'               |'cut[|s|ting]'
      |'expan[d|ds|ded|sion[|s]]'   |'reduc[e|es|ed|tion[|s]]' end

define x_econ 'cost[|s]' | 'payroll*'
      |{ 3 words -> 'work' and 'force') } end

/*
 * Query template used for Topic 75
 */
{ (prox1) ->
      @x_auto and
      { (prox2) -> @x_up_dn and @x_econ } }
```

TABLE II - Sample Statistical Query

Term weighting table for Topic 93, NRA Political Backing

Doc sample size is 150

Term	DocCount	DbCount	Weight	PslWeight
nra	87.0	130.0	0.66923077	62
national_rifle_a	145.0	255.0	0.56862745	53
assault_weapons	51.0	161.0	0.31677019	30
semiautomatic	68.0	332.0	0.20481928	19
gun_control	73.0	387.0	0.18863049	18
handguns	51.0	319.0	0.15987461	15
rifle	146.0	1336.0	0.10928144	10
handgun	52.0	550.0	0.09454545	9
firearms	63.0	811.0	0.07768187	7
rifles	46.0	1318.0	0.03490137	3
gun	126.0	3866.0	0.03259183	3
guns	96.0	3020.0	0.03178808	3
law_enforcement	51.0	2509.0	0.02032682	2
assault	67.0	3491.0	0.01919221	2
ban	91.0	5510.0	0.01651543	2
enforcement	54.0	4825.0	0.01119171	1
weapons	99.0	8993.0	0.01100856	1
association	148.0	16780.0	0.00882002	1
legislation	64.0	7984.0	0.00801603	1
laws	45.0	7662.0	0.00587314	1

TABLE III - TRW/Paracel TREC-II Routing Results

Qry	NIST Rel.	Relevant Retr @100 Best	Median	Worst	TRW1 Rel	TRW2 Rel	Description
51	11	11	10	0	10	10	Airbus Subsidies
52	454	100	97	31	99	99	S. Africa Sanctions
53	154	84	50	1	52	55	Leveraged Buyouts
54	124	65	57	0	21	47	Sat. Launch Contracts
55	320	100	87	0	54	96	Insider Trading
56	395	96	88	3	89	96	Prime Rate Moves
57	319	91	73	0	61	91	MCI
58	76	62	39	4	49	28	Rail Strikes
59	574	97	80	1	95	95	Weather Fatalities
60	18	9	4	0	2	3	Merit-Pay vs. Seniority
61	67	60	24	2	7	24	Israel & Iran-Contra
62	426	92	75	9	82	81	Military Coups D'etat
63	74	40	24	0	24	29	Machine Translation
64	282	68	53	1	29	57	Hostage-Taking
65	214	55	29	0	2	46	Info Retrieval Systems
66	86	40	19	0	20	23	NLP
67	365	64	52	0	10	61	Civil Disturbances
68	76	61	38	0	58	55	Health Hazards
69	1	1	0	0	0	0	Reviving SALT II Treaty
70	34	32	28	0	31	32	Surrogate Motherhood
71	300	73	35	1	21	32	Border Incursions
72	91	43	31	0	17	30	Demographic Shifts/U.S.
73	355	74	49	0	23	74	Demographic Shifts/World
74	323	60	32	5	6	35	Conflicting Policies
75	372	59	28	1	59	31	Automation
76	163	34	24	0	11	34	Original Intent
77	85	49	36	0	36	40	Poaching
78	83	66	58	0	40	57	Greenpeace
79	341	85	75	0	81	77	FRG Party Positions
80	143	54	8	0	54	8	Candidate Platforms
81	4	4	2	0	1	2	PTL Fallout
82	203	89	65	0	30	87	Genetic Engineering
83	235	64	31	2	54	25	Protect the Atmosphere
84	101	27	16	0	22	14	Alternative Energy
85	670	88	69	38	59	84	Official Corruption
86	40	27	16	0	19	12	Bank Failures
87	151	75	42	2	42	36	S&L Prosecutions
88	32	29	21	0	11	26	Crude Oil Price Trends
89	17	8	2	0	3	7	OPEC Investments
90	75	39	10	0	39	9	Oil & Gas Reserves
91	9	6	3	0	5	5	Acq of Advanced Weapons
92	27	17	6	0	7	9	Military Equip Sales
93	94	65	61	9	61	62	NRA
94	300	76	47	0	57	62	Computer Crime
95	359	77	45	1	63	40	Computer Crime Detection
96	310	86	42	2	23	49	Computer Medical Diag
97	319	59	21	0	20	41	Fiber Optics Appl
98	722	96	67	0	83	84	Fiber Optics Manuf
99	291	99	92	10	91	89	Iran-Contra
100	204	94	84	0	52	89	Controlling High Tech

5.0 Analysis of Results

The results from our two TREC runs (Table III) are summarized below. The proximity queries (TRW1) scored at or above the median on 28 topics (including three topics which achieved the best score) and below median for 22 topics. While not bad, our proximity queries did not perform as well as we'd hoped. Where the proximity queries did poorly, we attribute this primarily to poor term selection. One such case was Topic 65, Information Retrieval Systems. We made two errors: first, due to an oversight, one of the manually entered query terms was overly broad; second, the query author considered database systems to be "information retrieval systems". We feel that this is a fault of the query formulation, not of the assessments. If we had checked the training assessments, we would have eliminated the terms from our query. Any system which relies solely on the topic statement will run afoul of this problem. Systems which make use of user-supplied relevance information will achieve better performance. Our results again demonstrate the inherent problems with basically boolean query formulations. Our efforts at using proximity to soften the boolean were not sufficient to overcome this weakness.

Run	High	Above Med	Median	Below Med	Low	11-pt avg
TRW1	3	19	6	22	0	0.2525
TRW2	5	27	5	13	0	0.3459

The statistical queries (TRW2) did much better, scoring at or above the median on 37 of the topics. The 11-pt average and total relevant documents retrieved figures were excellent and are close to the best academic groups. The adaptations to run the statistical queries on the FDF hardware evidently did not hurt performance. We again observed that the dominant factor in achieving good performance is proper term selection. The details of the term weight calculations didn't seem to make much difference except to influence which terms were selected. We tried a number of different schemes for generating term weights including using various statistical parameters, log weighted coefficients, and converting terms present in all sample documents to boolean ANDs in the query. For a given set of terms, we did not find much difference in performance between these schemes. The scheme used for TRW2 was one of the simpler ones we tried.

We were also interested in evaluating the use of phrases and special features as additional terms in our statistical queries. They seemed to help, but not dramatically. This was a disappointment. Looking at the results topic by topic however, we observed a lot of variation. For some topics, the addition of a key phrase or special feature helped a great deal. This indicates that use of phrases and special features has promise for improving performance, but that we just have not learned how and when to employ them. For example, our term weighting scheme this year didn't account for term interdependence. Particularly when we start mixing single word terms with phrases and special features that contain those same terms, it would seem the algorithm could be improved by explicitly accounting for this redundancy.

6.0 Conclusions and Future Plans

Overall, our results for TREC-2 are encouraging. We were successfull in adapting the FDF hardware to running a soft statistical information retrieval algorithm. Not only did the FDF run all the final searches with no significant post-processing, we also found it a useful tool for experimenting with different proximity windows and deriving database frequencies for phrases and special features. We plan to continue our work and to participate in TREC-III next year. The lessons we learned from TREC-I have proven very valuable, and so too should the lessons of TREC-II. For TREC-III, we will continue to examine the range of available query formulation, execution and evaluation models, with an eye to adapting those models for use with the FDF search hardware. We are looking at methods to use the raw horsepower of the FDF to expand existing techniques in ways which had previously been considered too computationally expensive.

We are also struck by the observation that different query generation techniques are effective on different topics. A system that could execute a range of methods might be able to match the query generation approach to the topic type. We plan to consider possible schemes for characterizing the topic in advance so that the system might be able to use the best method.

Knowledge-Based Searching with TOPIC®

John W. Lehman, Clifford A. Reid, et al.

Verity, Inc.
1550 Plymouth Street,
Mountain View, CA 94043
(415) 960-7620 / jlehman@verity.com

1. OBJECTIVE OF VERITY'S TREC-2 EXPERIMENTS

Verity, Inc. is the first major commercial product participant in TREC. Verity's product is TOPIC®.

Verity participated in TREC-2 as a Category A Site. This participation was Verity's first TREC, and we encountered many of the logistical problems of other sites in their TREC-1 experience.

Topic's search users wish to understand the search result quality to expect in their personal searches on their (large) collections. Verity also expects to obtain insights for future product improvements.

Topic is a mature commercial-off-the-shelf manual text search program combining the results of human expertise with a powerful search expression language and fast search algorithms. Topic's installations use manually or semi-automatically developed libraries of searches (topics) , which are instances of the search expression language and which are supplied to all users.

Verity begins its TREC experiments with a gathering of "ground truth" regarding unaided adhoc end user search result quality. Future experiments will incorporate predefined searches (topics) and other Topic search aids to determine their level of improvement/impact on search result quality.

2. TOPIC SEARCH APPROACH

The Topic philosophy: Domain knowledge, both descriptive and content-based, using constructs specifically designed to discriminate between full text material, is the only way to consistently obtain high recall/precision on large heterogeneous collections. Search result quality may be enhanced by the employment of collection-specific statistics to locate additional domain-relevant terminology. Searches are repeated and subject-matter expertise is a scarce resource. The problem that Topic addresses is the effective use of a human's time in analyzing search results to locate the preponderance of relevant details in the fewest possible documents, and therefore the smallest possible elapsed time.

2.1 TOPIC KNOWLEDGE REPRESENTATION

The Topic product employs several approaches to individual term search, organized by a rule-based, or concept-based, approach to search term aggregation. In Topic, the search focus is the topic, (concept, notion, idea, or subject), and the topic is the user-specified "smart" description of all of the evidence "about" or "of" the topic as it (the evidence) would be found in text documents.

2.1.1 TOPIC INDICIES

The Topic product line catalogs and indexes both fielded (structured) data, and full-text. Topic automatically extracts structured data (such as title, author, etc.) into searchable fields, using a lexical analyzer. Fielded data is searchable separately or in combination with full-text.

Indexes on the full-text are (for all non-stopped characters and strings):

- -word/string
- -stemmed word (morphological variant)
- -soundex (phonetic spelling variety)
- -statistically correlated terms (called the suggestion index)
- -typographical error index
- -thesaurus
- -wildcard (universal character/group expansion)

An index on all values (choices) for fielded data is also produced.

2.1.2 TOPIC SEARCH RULES

Search rules consist of relational comparisons to field values, exact or fuzzy matches on full-text search terms,

aggregated by boolean and evidential reasoning operators and point value uncertainty at the term level (each piece of evidence has a strength/uncertainty attached to its predictability of its parent concept).

Topic provides search rule management functions to support the creation, repeated use, modification, sharing and display of one or more libraries of related search rules. The search rule libraries are themselves searchable, including text annotations of the rules. Search rules are interactive queries, automatic queries, and a training mechanism for the installation's domains.

A search rule definition may include several thousand pieces of evidence in over one hundred levels of detail. One search rule library may contain twenty thousand rules. Search rules (topics) are named, and a reference to the name in a search expression inherits all lower levels of evidence. Any query which includes a search rule name will automatically receive the full definition of the rule in the search. The lowest level of evidence is the text expression. Search rules may be composed of other named search rules.

Search rules appear as an alphabetical list of topic names, an indented outline showing the levels of rules, or a graphical "family tree" display of rules and their parents/children, including evidence combination operators and evidence "weights". Searches may be executed directly from any node (name) in the search rule family. A topic search rule graphic display example appears in Figure 1.

The search rule syntax consists of an exact or fuzzy match (pattern match) capability for individual terms (case sensitive); a boolean combination (and (all), or (any), not), of terms; dual direction, nested, grammatical (paragraph, sentence, phrase) proximity operators; a relative (fuzzy) proximity operator for two or more terms, an evidence aggregation operator (accrue) for both full-text and structured field data, and inexact match techniques as follows:

1. wildcard expressions for term expansion; single character, character group, or character class

2. soundex (first letter common) expressions for morphological term expansion

3. source language-specific stemming (morphological variants) expressions for term expansion

4. typographical expressions for term expansion (n-character infidelity to search term)

5. multi-direction thesaurus (user-modifiable) for term expansion

6. suggestion (statistical correlation) for term expansion

7. evidence appearing in a field value, or as the field value (contains, matches, substring, starts, ends).

Each of the above inexact match techniques may be executed automatically. Negative evidence may be applied on a term-by-term basis with any operator. The structured field data types are character, number and date. Date arithmetic is provided, as well as relative date expressions such as "yesterday" ,"today" etc.

2.1.3 SEARCH RESULT RANKING

Results of searches are relevance ranked lists of documents, with displayed titles or other descriptive information. The numeric score, and the accompanying rank, are the result of a best fit comparison of the full-text document and descriptor content and the search rule evidence. The ranking is subject to an optional threshold, used primarily to limit output, but the threshold may be used to describe search recall and precision. The relevance threshold is always used in dissemination/notification.

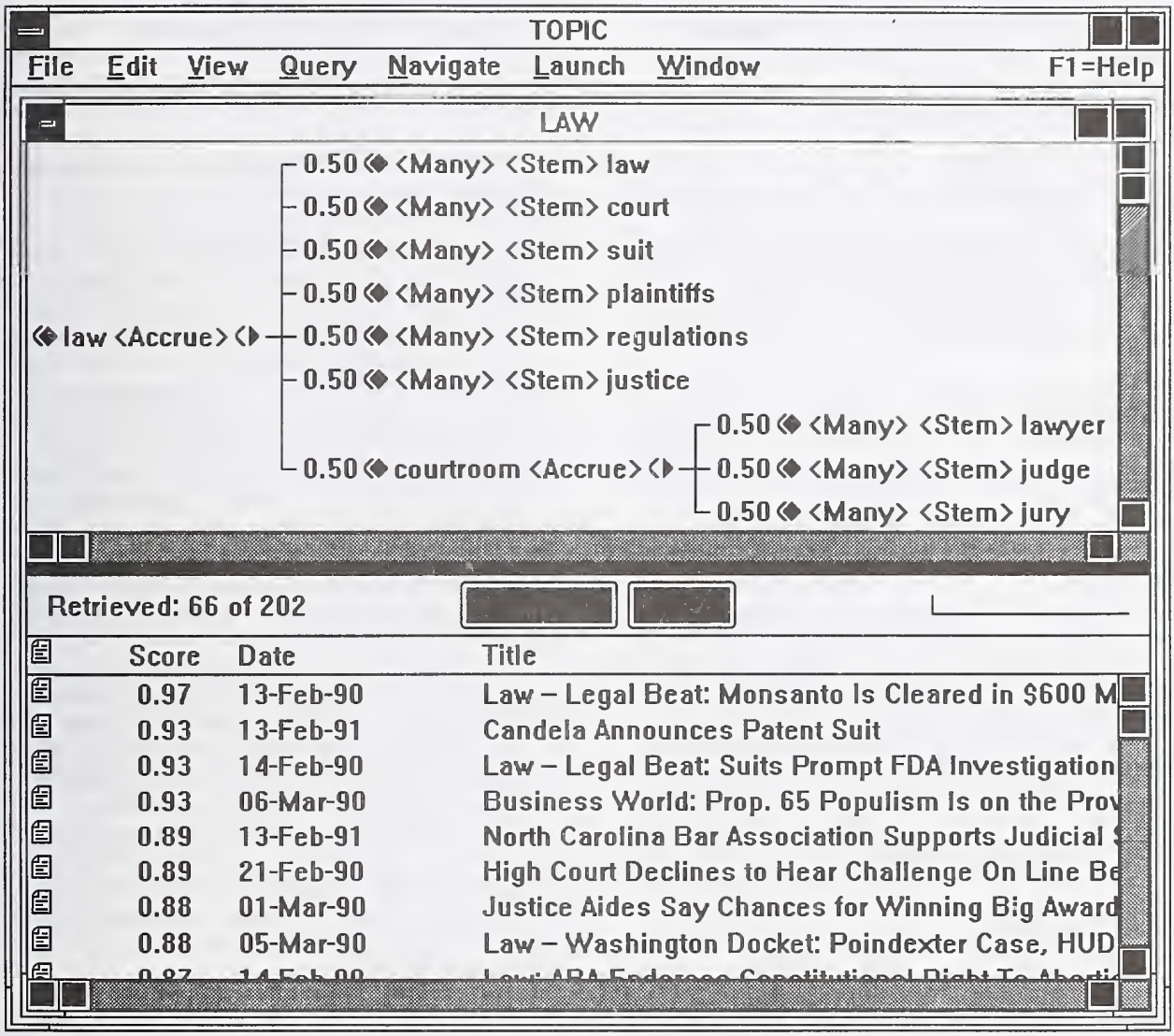
Evidence consists of terms, operators (syntax) and the numeric strength of the relationship between the evidence and its (next higher level) search rule. The evidence may be aggregated or evaluated with boolean operators. Aggregation involves giving relevance score credit for each piece of evidence found (breadth of evidence first). As each level is evaluated in a search rule (tree), potential document score modification occurs (since successive levels may be weighted evidence for their next broader concept). The scoring of an individual term may include a frequency-of-occurrence factor (a normalized concentration factor) , a less powerful scoring factor than the absolute presence of the evidence in the document. A document score explanation function is included.

2.2 AGGREGATE SEARCH FUNCTIONS

Searches may iterate on the results of the previous search.

Any search may be named/saved along with its results manipulation criteria (sorting by fields, grouping) for later execution. Any search criteria may be interactively defined as a logical view of the collection, which then provides many alternative search universes for the user population. All Topic activities are audited. A search which supports discretionary access control may be transparently appended to any users search.

Figure 1
TOPIC Search Rule and Result



2.3 USER INTERFACE TO SEARCH

Every search is automatically configured into a rule. The simplest search is a list of terms, which may be entered at the keyboard, selected from displayed document(s) content, or selected from lists of terms. This list is automatically enhanced by term expansion, expansion to existing named rules whenever the rule name appears in the search expression, and evidence aggregation. Searches involving structured fields are generally addressed by a form interface, which aggregates field and full-text content. Any list of terms, rule-names, or extensions such as thesaurus/soundex may be used to initiate a search or add to a search expression.

2.4 SEARCH RESULT ANALYSIS AIDS

The Topic philosophy of minimizing the elapsed time to obtain the necessary relevant details that constitute an answer or support a decision necessitates analysis aids beyond the search composition and result list display.

The Topic result list may be browsed (page, result number etc.). A document selected for display produces the full text display with all search evidence highlighted (e.g. in reverse video or color). The display may be the native form of the document, which for most of today's collections means a marked-up format with useful user guidance in the markup itself (e.g. sections, paragraph headings etc.). The user may choose to browse or to move directly to the first/next/previous occurrence of a search term in the document. Similarly, the user may move through the document using various document enhancements such as hypertext links, may follow hypertext links to other documents, including graphics and other media. Previously generated annotations are available for browsing. Queries or other applications may be linked to document content. A specific search term (not necessary to be a part of original search) may be used as a browsing aid to the document.

2.5 SECURITY

Users may be prevented from accessing information via operating system permissions, and built-in access controls, including discretionary. The product processes have been certified at system high in many installations, and some sponsors have applied for MLS certifications based upon the delivered product.

2.6 DATA ARCHITECTURE/ PERFORMANCE/ CONFIGURATION

Topic enables the logical division of a collection of documents into "partitions", which are document descriptions and indexing data about the arbitrary/ intentional subset. Partition size, purpose and characteristics are under the application administrator's control. The raw documents are not "owned" by the Topic application. Topic will produce indicies which are approximately 70% of the size of the native text size (the TREC-2 index size was approximately 50%). This includes fielded, word, and subject (rule evidence) level indicies.

The partition data is platform-independent (i.e. the documents and their associated partitions may be moved/accessed from any Topic platform.

Searches may be performed on the served desktop, on a host or both.

Normal performance on a personal computer is in the thousands of document-rule nodes per second, up to many tens of thousands of nodes per second on current workstations. The search rule low level evidence is contained in a size/speed-optimized index (topics), which is essential to rapid response on complex rules. This index is automatically modified each time topic evidence is added, so the word positional information is searched only on the first use of the term. The topics index normalizes document size so that all search response times are predictable. Partitions enable incremental (ranked) results, guaranteeing few-second time-to-first-result, regardless of the size of the collection. The response characteristic which Topic optimizes is the time-to-first-meaningful-result. The rule evidence index may be centralized or distributed, and when distributed, it provides the ability to produce a ranked results list with a minimum of network access.

Integration with third party components is available from the end user interface, or shared libraries. The program provides logical links between document-image, document-document, document-annotation, document-search request. Some links may be automatically determined at indexing time (image, cross-reference).

The structured field values may be entered interactively, or filled automatically from a lexical analyzer. The program provides an enduser process interface between scanning, OCR/ICR and indexing.

3. THE TREC EXPERIMENTS

3.1 DATA PREPARATION

The TREC-2 texts data preparation processing was performed on a Sun SPARC 10 (UNIX 4.1.3). Cataloguing and indexing was performed at the rate of approximately 100 Mbytes per hour. This process included the automatic extraction of 10 fields from the ASCII content. Partitions were set at 8000 documents for all data. There were no processing errors.

No markup language (SGML) interpreter was used during data preparation, and the optional alphabetical word list (used only for display) and typographical error index (used almost exclusively for OCR'd data) were not employed. Special indicies such as correlated terms, and paragraph/sentence positioning were not produced. As the fuzzy proximity operator was used in the tests, only a word position index was produced. No document was divided into logical or arbitrary sections for processing or search result enhancement, although that approach is used in virtually all non-newswire Verity installations. The purpose of logical division (a forerunner of the intelligence available in a standard markup language) is to create domain-specific logical documents, and therefore to reduce the impact of larger, multi-subject documents on results (they would appear in search results simply because of their breadth of words).

3.2 TOPIC CONSTRUCTION

Verity personnel manually constructed the search rules from the subject area descriptions and the training data. No rule developer was identified or chosen as a subject matter expert, and for certain of the contributors, this was their initial interface with using Topic. [Search rule libraries are created by approximately 6% of Topic's user population and the remainder of Topic's users employ the topics developed by others]. On the average, the TREC-2 volunteers were considered novices on the Topic product, particularly the search rule development area. Volunteers were not encouraged to use specific features of the product, and in at least one case, inadequate communication produced potentially inaccurate search expectations. As search rules were interactively developed, the rule evidence was automatically indexed for repeated use of the rule. The twenty volunteers each produced between 3 and 8 retrospective and routing queries. The range in time spent on individual query development, and result production was from fifteen minutes to eight hours, over a several week period. The average time to produce the TREC-2 result, obtained from interviewing the volunteers, was approximately one hour.

3.3 EXPERIMENT PERFORMANCE

Typical response time performance on the searches was two seconds per 8000-document partition, or approximately two minutes to search the entire collection. A single term, indexed as rule evidence, was used to search the entire collection, and the 1.1 million document collection was searched in 21 seconds.

For routing queries, the score threshold was set to zero; any document containing evidence entered the routing result list.

3.4 ANALYSIS OF OFFICIAL RESULTS

The post hoc analysis of Topic's TREC-2 results generally found that the Topic system performed well. When compared with other manual systems, the scores are amongst the best. I the few cases where Topic appeared to fail, we have generally been able to identify easily correctable deficiencies, that, had they been noticed during the experiment proper, would have resulted in superior performance by Topic in TREC-2.

Based on our analysis, we believe that the prospects for TREC-3 look very bright.

Our analysis of selected results from our TREC-2 submissions focuses mainly on the "failure cases" since these are most likely to give us insights in how to improve Topics (and users) performance in future TREC experiments. This also allows us to investigate whether there are any fundamental issues with using Topic to model the information need statements used in TREC.

We analyzed two routing and three ad-hoc topics in detail. Our summary follows.

The following general observations applied to all searches:

-Adhoc searches were submitted against all three disks, which produced poorer quality results generally, as documents from disc three appeared in some search results.[1]

-Field value evidence was not used, and in some domains/subject areas, domain knowledge about the sources of information would favor (rank higher) sources with the appropriate use of terminology. (e.g. business sources about financial performance, or foreign datelines have higher likelihood of describing foreign prominent persons/activity, as in topic's 66 or 121)

-The queries which used attempted to use nomenclature with hyphens (e.g. M-1) failed to return an exact match as the hyphen was not included as an indexed character.

-The fuzzy proximity (near) operator was undocumented, only one volunteer used it and other users expected sentence / paragraph proximity in their searches. The index did not contain sentence / paragraph positional data, and all uses of sentence or paragraph operators produced erroneous results because the search arbitrarily assigned sentence and paragraph boundaries.

---

[1]Reprocessing the adhoc searches against only disks 1 and *2 produced a numberic result improvement of 0-70 percent, with a *few changes from under the median to over the median.

3.4.1 ROUTING TOPICS

Overall, Topic's performance on the routing topics was rather good. We count that 21 of the 50 results were at or above median, and three were actually the bet score. Most of the other results were on the low side of the median. The relevant comparison to the median is summarized in Figure 2. The exceptions were topics 66, 67, 69, 74, 90 and 91, for which the Topic search used could be said to have failed. Several of these were straightforwardly explained. For example, in the case of topic 67 the wrong results were submitted. Our independent scoring of the correct results set would give the Topic search below median score. For topic 69 there was in fact only one relevant document, but, at least in our reading of the definition, this seems to be a false positive. In the case of topics 90 and 91 the Topic search definitions were, in our opinion, over-constrained. Further, in the case of topic 91 an index creation decision prevented a quite reasonable Topic definition from performing as well as it could.[2] The other two topics are of more interest.

No clear pattern emerged between the type of search although, in the routing augmentation category, the Topic performance was well above the median on 20 of 33 searches.

3.4.1.1 ROUTING TOPIC 66

A relevant document for this topic is one that identifies a type of natural language processing technology that is being developed or marketed in the United States. The original definition of the Topic is basically a conjunction (AND) of a natural language concept and a products/technology concept.

Performance was very poor, viz:

Relevant = 86
Rel_ret = 1
R-Precision = 0.0000

Inspection of the Topic revealed that one of the conjuncts (the products/technology concept) had a weight of 0.05 - thus effectively limiting the range of scores that Topic could produce to be in an extremely narrow range.

We changed the 0.05 to 0.5 and produced the following:

Relevant = 86
Rel_ret = 44
R-Precision = 0.2442

which is a median result.

We concluded that for Topics to be effective we need to ensure a sufficient range of scores to give us the discrimination needed for the TREC scoring algorithm.

3.4.1.2 ROUTING TOPIC 74

A relevant document for this topic is one that cites an instance in which the U.S. Government propounds two conflicting or opposing policies. The routing task is complicated because this conflict may not necessarily be mentioned in the same document.

In our opinion, this is a case where no amount of sophistication in Topic construction would enable Topic to do very Well. The information need is simply outside the scope of a retrieval system that uses non-NLP techniques. The best one could hope for is to model a document that talks about the meta-idea of conflict (i.e., find documents that talk about the US having conflicting policies, rather than documents that reference the specific conflicting policy). This is, in fact, what was done in the original submission. The results were:

Relevant = 323
Rel_ret = 18
R-Precision = 0.0464

which is, of course, rather poor.

The original statement of need actually mentions three examples of conflicting policies so, as an experiment, we ran the following query:

* <Many><Stem>
  /wordtext = "tobacco"
* <Many><Stem>
  /wordtext = "pesticide"
* <Many><Phrase>
* <Many><Stem>
  /wordtext = "infant"
* <Many><Stem>
  /wordtext = "formula"

that is, just an ACCRUE of "tobacco", "pesticide" and "infant formula" (which the modification that the <Stem> and <Many> operators produce.

This gave the following results:

Relevant = 323
Rel_ret = 107
R-Precision = 0.2660

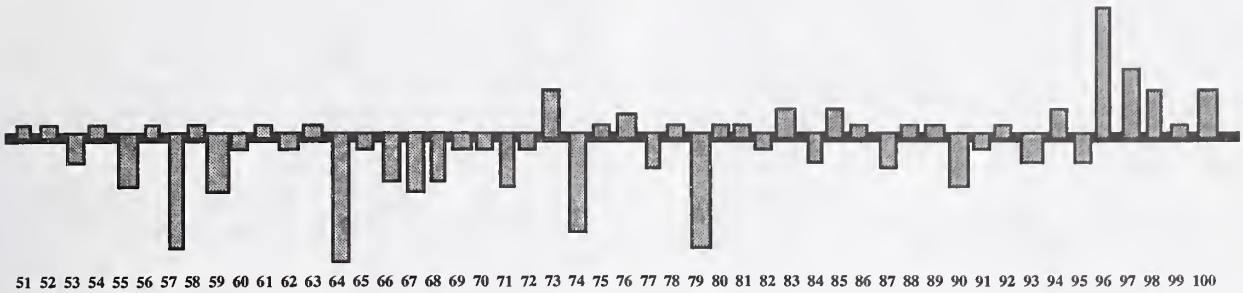
which puts the score slightly above median. We expect that most TREC-2 participant sites probably did just this, and those that did much better than median found some other specific examples of a conflicting policy and modeled these in their routing queries.

---

[2]This topic is about the acquisition of advanced weapons by the U.S. Army. One of the weapons systems mentioned in the information need statement is the M-1 tank. This was included in the Topic definition as the word "M-1"; but since the "-" symbol was interpreted as with like space at database build time, there was no possibility of retrieving documents based on "M-1" as a word.

Figure 2

TOPIC2 Relevant vs. Median - Routing Topics



3.4.1.3 TOPIC 67

Our analysis located weak topic formulation examples, such as query 67, illustrated in Figure 4. In this query, a set of optional, auxiliary evidence was "ANDed" with a small set of required evidence. The weight, or strength assigned to the auxiliary evidence was .05, which means that if all auxiliary terms were located, the highest possible score for a document would be .05, severely limiting the range of scores, and thus the occurrence of random false hits in the top 1000.

To make a cosmetic improvement, only the value of the auxiliary evidence node was changed, to a value of .5, as shown in Figure 5. This change alone brought the Topic relevant document count to the median.

3.4.2 AD HOC TOPICS

Overall, Verity's performance on the ad hoc topics was adequate. Performance was poorer than on the routing topics, but this is to be expected since there was less time available to build the Topics and no ground truth against which to test the Topic trees. The relevant comparison to the median is summarized in Figure 3. We count that 13 of the 50 results are at or above median. In contrast thought, there were only two outright failures here, topics 124 and 139. We did not look at topic 139, but topic 124 involves searching for documents that discuss innovative approaches to cancer therapy that do not involve any of the traditional treatments. This is a very hard topic because nearly all mentions of the innovative treatments are in the context of discussion of traditional therapies The approach adopted by Verity of simply looking for documents that talk about innovative treatment produces a large number of false hits (giving poor precision), and since there is an artificial cut-off at 1000 documents in the TREC experiments, this model also produces poor recall. We do not see an obvious solution to this.

We picked three ad hoc topics to analyze in detail.

3.4.2.1 AD HOC TOPIC 109

A relevant document for this topic simply needs to mention one of a list of six companies given in the information need statement. A simple Topic that is the disjunction (OR) of the company names should be all that is needed here. However, the official result is:

Relevant = 742
Rel_ret = 192
R-Precision = 0.2588

which is well below median. furthermore, given the simplicity of the topic, this is surprisingly low recall. Examination of the official Topic showed that company acronyms we used for three of the companies (i.e., 3M, OTC, ISI) were given equal weight to the fully spelled out company names. A cursory review of the original hit list showed that ISI was a poor choice since it has multiple interpretations. Less important, but for the same reason, OTC is a poor choice in the Wall Street Journal corpus since it can mean "over the counter", and in the DOE corpus 3M is part of a designator for a particular particle accelerator and is also used as an abbreviation for "three meters".

We modified the Topic by eliminating the ISI acronym and by giving OTC and 3M reduced weights. This produced the following:

Relevant = 742
Rel_ret = 480
R-Precision = 0.5512

which would have been the best score.

An interesting note here is that original and modified Topics had perfect precision and recall for the first 100 documents. Our conclusion is that this indeed was an easy topic - the false hits produced by ISI were what impacted Topics score.

3.4.2.2 AD HOC TOPIC 121

A relevant document for this document had to mention the death of a prominent U.S. citizen due to an identified form of cancer.

This is an interesting topic consisting of two major components - the idea of a prominent citizen, and the idea of a specific cancer.

In the official Topic, prominence was modeled using a number of words that indicate prominence (e.g., "prominent", "celebrity") together with words that indicate prominent roles (e.g., "Nobel Prize", "actor", "actress"). Cancer death was modeled by various combinations of death words (e.g., "death", "died") and cancer words (e.g., "cancer", "tumor", "leukemia"). The official score was:

Relevant = 55
Rel_ret = 27
R-Precision = 0.1455

which, while not good in absolute terms, was well above the median.

We observed two problems with this definition. First, it uses generic cancer terms rather than the specific cancer types required by the information need statement. So, we made all the cancer terms specific by using a list of common cancers (e.g., lung cancer, breast cancer, stomach cancer, etc.) . We made no attempt to make

Figure 3

TOPIC2 Relevant vs. Median - Adhoc Topics

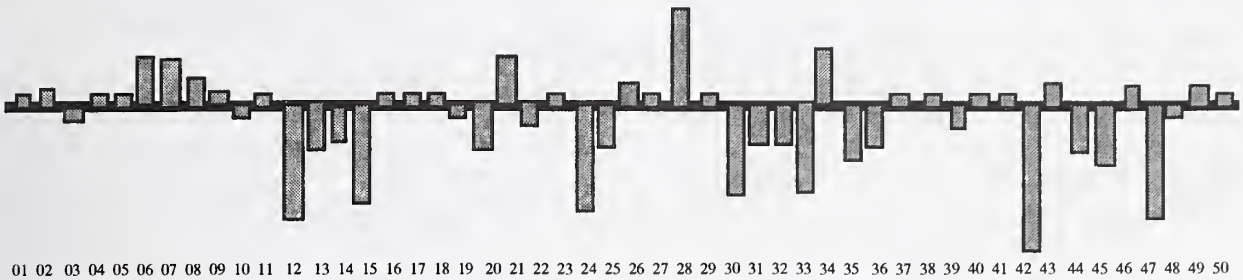


Figure 4

Example of Poorly Specified Search

Routing Query #66

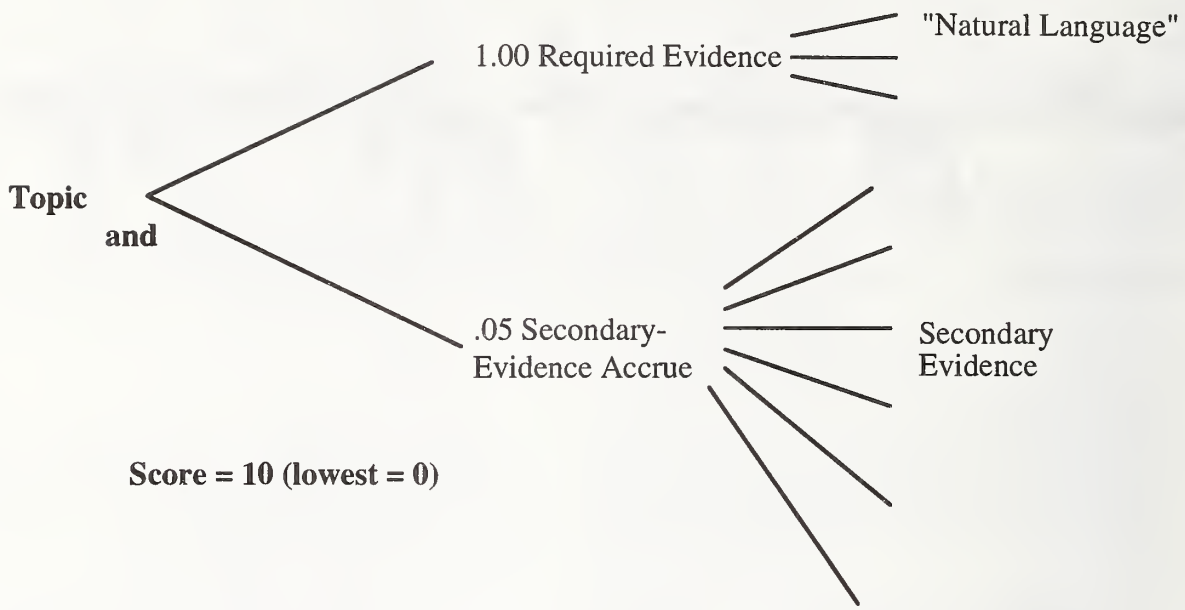
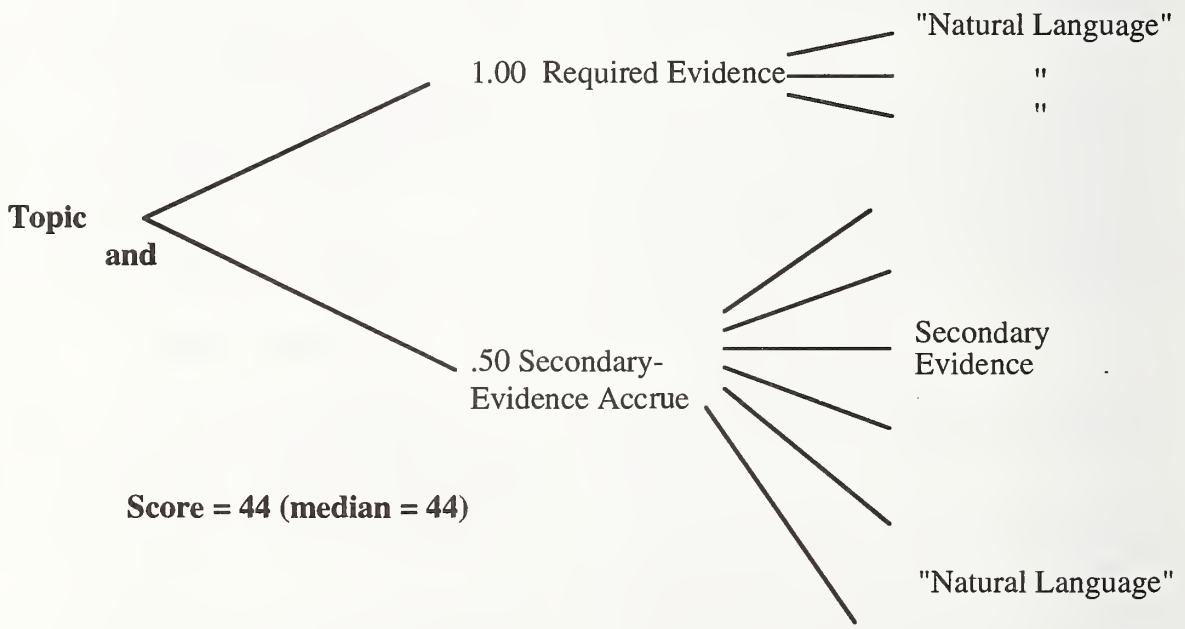


Figure 5

Poorly Specified Search - Cosmetic Repair

Routing Query #66



this list exhaustive. This produced the following results:

Relevant = 55
Rel_ret = 17
R-Precision = 0.2182

Thus we reduced the recall, but increased the precision. Presumably by adding more specific cancers (or at least the ones that statistically are most common) we could have improved the recall here.

The second problem is more severe though. It appears impossible to build any kind of model that would allow us to determine, with any kind of confidence, that the person who has died is a US citizen. In our revised results list we find many prominent persons who died of a named cancer but who are not US citizens (e.g., the Venezuelan Ambassador).

In addition, the notion of prominence is also hard to capture. Of course, we might argue that anyone who's obituary is on the wire service is prominent by definition! Be that as it may, we observed a number of documents that we did not retrieve because we had not included the specific prominent role indicator in our Topic. Thus we added the following roles words - "author", "poet", "writer", "artist", "painter" - to the Topic and got the following results:

Relevant = 55
Rel_ret = 33
R-Precision - .0909

Thus we improved the recall but at the expense of the precision again. Notice that we still have not included any business or government roles, which presumably would help retrieve the relevant documents in the WSJ corpus.

Our conclusion, is that this is a significant challenge for Topic, and all other system. The citizenship question often cannot be resolved by reference to the text alone, and we see no alternative but accept the false hits. Prominence is also difficult, but could conceivably be approached by an extensive list of prominence and role. words. The specific cancer seems tractable since there are only a finite number of cancers and just a small set of those are common.

3.4.2.3 AD HOC TOPIC 133

A relevant document for this topic must describe some design feature of the Hubble Space Telescope, but must not report of the launch activity itself nor the Hubble Constant or Edwin Hubble.

The official Topic was essentially a simple structure of the form: Hubble Space Telescope and not launch and not Edwin Hubble. This gave the following results:

Relevant = 80
Rel_ret = 29
R-Precision = 0.3625

which is surprisingly poor given the apparent simplicity of the topic.

Analysis of the behavior of the negation function in Topic shows that it is too restrictive, and so we eliminated the negated concepts leaving just the phrase "Hubble Space Telescope". Using this as the query gave:

Relevant = 80
Rel_ret = 78
R-Precision = 0.6000

which would have been above median and close to best.

Adding as disjuncts (OR) the words "Hubble" and "HST" gave:

Relevant = 80
Rel_ret = 79
R-Precision = 0.6000

that is we retrieved one extra relevant document with no decrease in precision.

We conclude that although the information need statement is careful to spell out the cases where the document will be non-relevant, the TREC corpus has few documents where these conditions apply, so that a simple query performs very well. This is presumably the approach most sites took.

4. FINAL OBSERVATIONS FROM TREC-2

The TREC-2 topic descriptions, particularly the ad hoc topics, exceed the level of domain knowledge available to most users of heterogeneous document collections.

Most Topic (content-based) search operational users are driven by time pressures to locate/summarize the most relevant details in the fewest possible documents. The exhaustive search result analysis implied by examining hundreds of relevant documents will not be addressed in most user environments; our experience is that ten to thirty documents is the level of search result analysis performed by a user (unless significant duplication of material occurs earlier, which would reduce the number of documents actually analyzed). Ergonomically, high precision in the first (10,20...50) documents is more likely to keep users attracted than high recall at much larger counts.

Although we have yet to perform any analysis of duplicate information on the TREC2 results, our belief is that duplicate data is plentiful in the TREC2 "relevant lists", and that the reading of duplicate data by the human user will cause the result analysis to be (prematurely) terminated.

We are certain that, unless summarization is performed, the relevant search results on most topics are too numerous to warrant user attention. It would seem

reasonable to examine, at least for selected topics, whether the first ten/best ten documents address the domain well from a domain "precision/recall" perspective. To the extent that the domain is well served in a few representative documents, the coverage in the representative documents may be a "better" answer for the user than the numerical count of the number relevant in the first 1000. We recommend adding a measurement of the coverage of the domain as the first ten/thirty/n results documents are examined.

Appendix A
COMPANY AND PRODUCT SUMMARY

Topic is a commercial off the shelf software product line available from Verity, Inc. Topic search technology is a commercial adaptation of ideas extracted from the research of Tong, McCune et. al., in Rule-Based Information Retrieval, which was sponsored by the U.S. Intelligence Community. Topic supports cataloguing, indexing and retrospective search of fixed collections, automatic search of newly indexed documents according to (user) predefined search rules (profiles), and dissemination/notification based upon satisfied search rules. Documents may be batched for indexing/profiling, or processed automatically as they arrive.

The Verity, Inc. market presence in content-based text search/retrieval is described in the Delphi, Inc. 1992 Industry Summary. The Verity Topic product line is considered to have in excess of a ten percent share of the market in commercial-off-the-shelf content-based search/retrieval products for personal computer to minicomputer environments.

Verity was founded in April 1988. The Topic product was first licensed and installed by the U.S. Air Force in June 1987. Verity currently has over 650 installations and some 30,000 users. Many thousands of persons have received training from Verity on the Topic products. Approximately one-third of Verity's installed base uses an event-driven or batch automatic-search-notification function.
Many organizations use the routing mechanism for users who are unable to compose the (appropriate) queries, but require the expert's result quality.

The Topic product line supports nearly twenty varieties of the UNIX operating environment, VMS, OS2, DOS and MacIntosh. The product operates on data stored in the filesystem or in any SQL-based data base management system. The product as shipped supports over twenty formats of native data (markup languages), and provides the ability to insert local/third party markup language interpreters as required. A document in Topic is logical, and may be a file, subfile or any logical decomposition of a physical native document.

The Topic end user (search) product is available in MSWindows, Presentation Manager, X-Windows-Motif, Macintosh, and character (keyboard/terminal) interface styles. There is a 4GL-like command interpreter language for rapid application development and remote command line interactive index/search. There is an Application Program Interface (C- library) to all Topic functions for embedded applications.

On Expanding Query Vectors with Lexically Related Words

Ellen M. Voorhees
Siemens Corporate Research, Inc.
755 College Road East
Princeton, NJ 08540
ellen@learning.scr.siemens.com

Abstract

Experiments performed on small collections suggest that expanding query vectors with words that are lexically related to the original query words can improve retrieval effectiveness. Prior experiments using WordNet to automatically expand vectors in the large TREC-1 collection were inconclusive regarding effectiveness gains from lexically related words since any such effects were dominated by the choice of words to expand. This paper specifically investigates the effect of expansion by selecting query concepts to be expanded by hand. Concepts are represented by WordNet synonym sets and are expanded by following the typed links included in WordNet. Experimental results suggest that this query expansion technique makes little difference in retrieval effectiveness within the TREC environment, presumably because the TREC topic statements provide such a rich description of the information being sought.

1 Introduction

The IR group at Siemens Corporate Research is investigating how *concept spaces* — data structures that define semantic relationships among ideas — can be used to improve retrieval effectiveness in systems designed to satisfy large-scale information needs. As part of this research, we expanded document and query vectors automatically using selected synonyms of original text words for TREC-1 [5]. The retrieval results indicated that this expansion technique improved the performance of some queries, but degraded the performance of other queries. We concluded that improving the consistency of the method would require both a better method for determining the *important* concepts of a text and a better method for determining the correct sense of an ambiguous word.

We took TREC-2 as an opportunity to investigate the effectiveness of vector expansion when good concepts are chosen to be expanded. As in TREC-1, query vectors were expanded using WordNet synonym sets. However, the synonym sets associated with each query were selected manually (by the author). These results therefore represent an upper-bound on the effectiveness to be expected from a completely automatic expansion process.

The results of the TREC-2 evaluation indicate that the query expansion procedure used does not significantly affect retrieval performance even when important concepts are identified by hand. Some expanded queries are more effective than their unexpanded counterparts, but for other queries the unexpanded version is more effective. In either case, the effectiveness difference between the two versions is seldom large. Further testing suggests that more extreme expansion procedures can cause larger differences in retrieval performance, but the net effect over a set of queries is degraded performance compared to no expansion at all.

The remainder of the paper discusses the experiments in detail. The next section describes the retrieval environment, including a description of WordNet. Section 3 provides evaluation results for both the official TREC-2 runs and some additional supporting runs. The final section explores the issue of why the expansion fails to improve retrieval performance.

2 The Retrieval Environment

The expansion procedure used in this work relies heavily on the information recorded in WordNet, a manually-constructed lexical system developed by George Miller and his colleagues at the Cognitive Science Laboratory at Princeton University [4]. WordNet's basic object is a set of strict synonyms, called a *synset*. Synsets are organized by the lexical relations defined on them, which differ depending on part of speech. For nouns, the only part of WordNet used in this study, the lexical relations include antonymy, hypernymy/hyponymy (*is-a* relation) and three different meronym/holonym (*part-of*) relations. The *is-a* relation is the dominant relationship, and organizes the

synsets into a set of approximately ten hierarchies[1]. Figure 1 shows a piece of WordNet. The figure contains all the ancestors and descendents as defined by the *is-a* relation for the six senses of the noun *swing*. Also shown is that one of the senses, a child's toy, is *part-of* a playground.

Given a synset, there is a wide choice of words to add to a query vector — one can add only the synonyms within the synset, or all descendents in the *is-a* hierarchy, or all words in synsets one link away from the original synset regardless of link type, etc. One of the goals of this work is to discover which such strategies are effective. Wang et al. found expanding vectors from relational thesauri to be effective [6], but based those conclusions on experiments performed on one small collection. Experiments we performed as part of our TREC-1 work showed showed serious degradation when anything other than synonyms were used in the expansion — but the TREC-1 results were dominated by the problem of finding good synsets to expand. This work examines the effectiveness of the different relation types assuming good synsets are used as the basis.

Siemens' official TREC-2 runs consist of one routing run (topics 51–100 against the documents on disk three) and two ad hoc runs (topics 101–150 against the documents on the first two disks). All of the runs are manual since the input text of the topics was modified by hand. There are two types of modifications: parts of the topic statement that explicitly list things that are not relevant were removed, and synsets containing nouns germane to the topic statement were added as a new section of the topic text. Document text was indexed completely automatically (once the errors were fixed[2]) using the standard SMART indexing routines [1] (i.e., tokenization, stop word removal, and stemming). In general, only the "text" fields of the documents were indexed. For example, only the title, abstract, detailed claims, claims, and design claims sections were indexed for the patent subcollection. The manually assigned keywords included in some of the Ziff documents were not used, nor were the photograph captions of the *San Jose Mercury* collection.

The goal in selecting synsets to be included in a topic statement was to pick synsets that emphasized important concepts of the topic. One aspect of the problem is sense resolution: selecting the synset that contains the correct sense of an ambiguous original topic word. However, since one purpose of the experiments is to investigate how effective lexical relations are in expanding queries assuming good starting concepts, I did not restrict myself to adding only synsets that contain some original topic word. For example, topic 93 asks for information about the support of the NRA and never mentions the word *gun*. Nevertheless, I believed *gun* to be an important concept of the topic and added the synset containing *gun* meaning "a weapon that discharges a missile from a metal tube or barrel" to the topic. (*Rifle*, a word that does appear in the topic statement, is a grandchild of this synset in WordNet, with the intervening synset being {*firearm, piece, small-arm*}). Synset selection was also influenced by the fact that these synsets would be used to expand the query. Early experiments demonstrated that expansion worked poorly when synsets with very many children in the *is-a* hierarchy (e.g. *country*) were used, so those synsets were avoided. Furthermore, when selecting one sense among the different senses in WordNet was difficult, I frequently used the words related to the synsets as a way of making a decision. Figure 2 shows the original text of topic 93 and the synsets that were added to it.

Some topics contained important concepts that had no corresponding synset. Occasionally, the missing synset was a gap in WordNet; for example, *toxic waste*, *genetic engineering*, and *sanctions* meaning economic disciplinary measures are not in version 1.3 of WordNet. More often, the important concept was a proper noun or highly technical term that one wouldn't expect to be in WordNet. *NRA* or *National Rifle Association*, for example, is an important concept for topic 93 but does not occur in WordNet. Nothing was added to the topic texts for concepts that lacked corresponding synsets in these experiments, although making some provision for them would improve retrieval performance.

Once the text of the topics is annotated with synsets, the remainder of the processing is automatic. Selected fields of the topic statements (the title, nationality, narrative, factors, description, and concept fields) are indexed using the standard SMART routines. The terms derived from these sections are "original query terms". The expansion procedure is invoked when the synonym set section is reached. The procedure is controlled by a set of parameters that specifies for each relation type included in WordNet the maximum length of a chain of that type of link that may be followed. A chain begins at each synset listed in the synset section of the topic text and may contain only links of a single type. All synonyms contained within a synset of the chain are added to the query. Collocations such as *change_of_location* in Figure 1 are broken into their component words, stop words such as *of* are removed, and the remaining words are stemmed. The word stems

[1]The actual structure is not quite a hierarchy since a few synsets have more than one parent.

[2]There were seven errors total in files patn_014 and patn_051 that were not on the official list, but caused the files to not conform to the patent collection's readme file. These errors — missing '/TEXT' tags , 'TEXT' tags preceding 'OREF' tags, and the like — were also fixed manually.

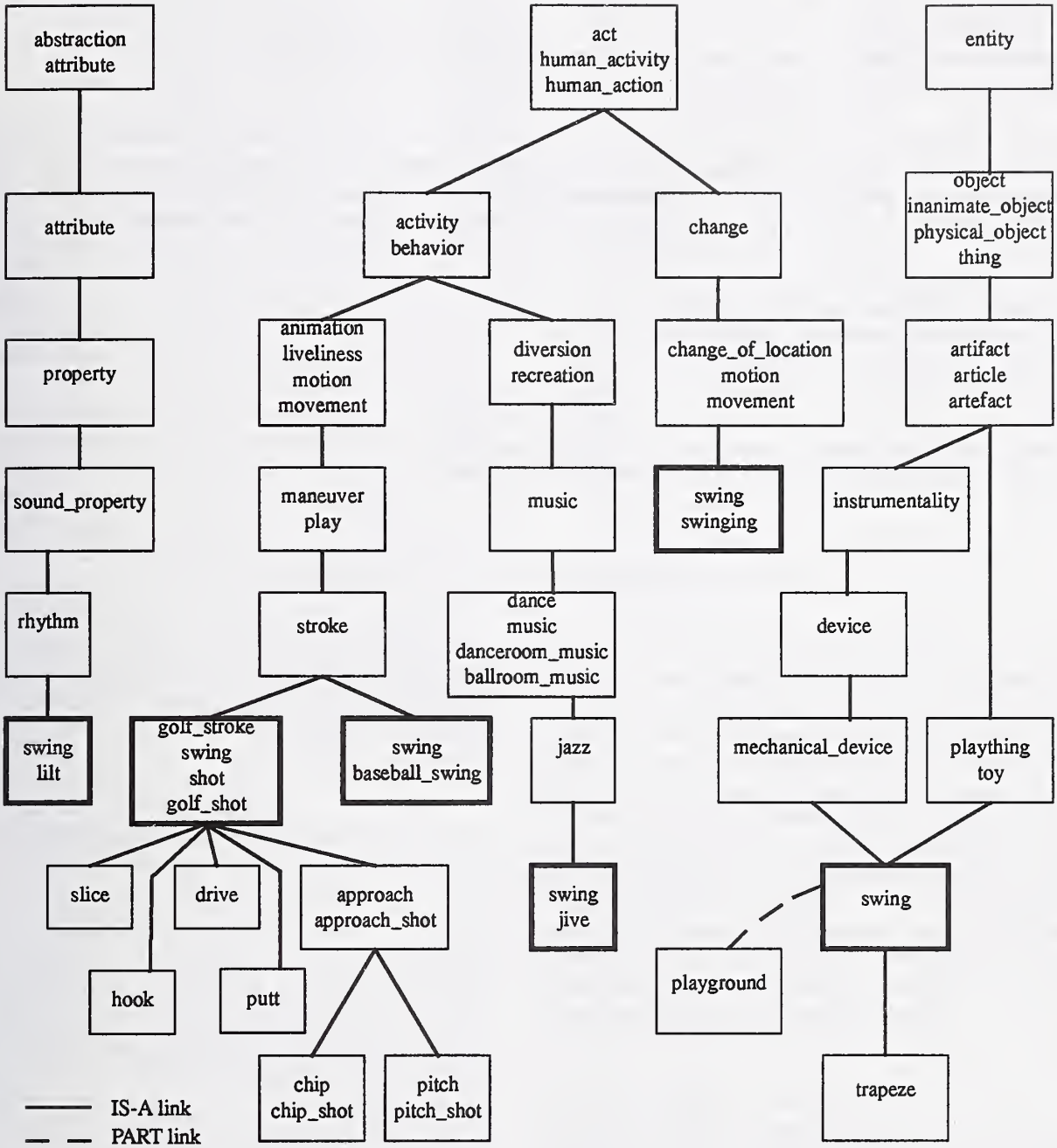


Figure 1: Relations defined for the six senses of the noun *swing* in WordNet.

```
<title> Topic: What Backing Does the National Rifle Association Have?

<desc> Description:
Document must describe or identify supporters of the National Rifle
Association (NRA), or its assets.

<narr> Narrative:
To be relevant, a document must describe or name individuals or organizations who are
members of the NRA, or who contribute money to it.  A document is also relevant
if it quantifies the NRA's financial assets or identifies any other NRA holdings.

<con> Concept(s):
1. National Rifle Association, NRA
2. contributor, member, supporter
3. holdings, assets, finances

<syn>
{funds, finance, monetary_resource, cash_in_hand, pecuniary_resource}
{supporter, protagonist, champion, admirer, booster}
{gun}
```

Figure 2: Topic 093 and the synonym sets selected for it.

plus a tag indicating the lexical relation through which the stems are related to the original synset are then appended to the original query terms.

As an example of the expansion process, consider the synsets for *swing* shown in Figure 1. If the synset added to the topic is the synset containing *golf_stroke*, and any number of hyponym (child) links may be traversed, then the stems of *golf*, *stroke*, *swing*, *shot*, *slice*, *hook*, *drive*, *putt*, *approach*, *chip*, and *pitch* would be added to the query vector. If hyponym chains are limited to length one, then *chip* and *pitch* would not be added. If the synset added to the topic is the one containing swing meaning plaything and any link type may be followed for one link, then the stems of *swing*, *mechanical*, *device*, *plaything*, *toy*, *playground*, and *trapeze* would be added to the query.

Stems added through different lexical relations are kept separate using the extended vector space model introduced by Fox [3]. Each query vector is comprised of subvectors of different concept types (called *ctypes*) where each ctype corresponds to a different lexical relation. A query vector potentially has eleven ctypes: one for original query terms, one for synonyms, and one each for the other relation types contained within the noun portion of WordNet (each half of a symmetric relation has its own ctype). An original query term that is a member of a synset selected for that query appears in both of the respective ctypes. Similarly, a word that is related to a synset through two different relations appears in both ctypes.

The similarity between a document vector D and an extended query vector Q is computed as the weighted sum of the similarities between D and each of the query's subvectors:

$$sim(D,Q) = \sum_{\text{ctype } i} \alpha_i (D \cdot Q_i)$$

where $\cdot$ denotes the inner product of two vectors, Q_i is the ith subvector of Q, and α_i, a real number, reflects the importance of ctype i relative to the other ctypes. Terms in documents vectors are weighted using the *lnc* weights suggested by Buckley et al. [2]; that is, the weight of a term is set to $1.0 + \ln(tf)$ where tf is the number of times the term occurs in the document and is then normalized by the square root of the sum of the squares of the weights in the vector (cosine normalization). Query terms are weighted using *ltN*: the log term frequency factor above is multiplied by the term's inverse document frequency, and the weights in the ctype representing original query terms are normalized by the cosine factor. Weights in additional ctypes are normalized using the length computed for the original terms' ctype. This normalization strategy allows the original query term weights to be unaffected by the expansion process and keeps the weights in each ctype comparable with one another.

3 Experiments

The training data for the routing queries was used both to refine the synsets that were included in the topic text and to select the type of relations used to expand the query vectors. In some cases a synset that appears to be a logical choice for a query is nonetheless detrimental. For example, adding the synset for *death* to topic 59 (weather fatalities) causes the query to retrieve far too many articles reporting on deaths that have no relation to the weather. I produced five different versions of synset-annotated topic texts, although the differences between versions are not very large. The version used in the official routing run added an average of 2.9 synsets to a topic statement, with a minimum of 0 synsets added and a maximum of 6 synsets added.

Of course, the utility of a synset depends in part on how that synset is expanded and the relative weights given to the different link types (the α's in the similarity function above). Table 1 lists the various combinations that were evaluated using the training data. Four different expansion strategies were tried: expansion by synonyms only, expansion by synonyms plus all descendents in the *is-a* hierarchy, expansion by synonyms plus parents and all descendents in the *is-a* hierarchy, and expansion by synonyms plus any synset directly related to the given synset (i.e., a chain of length 1 for all link types). Different α values were also investigated. Assuming original query terms are more important than added terms, the α for the original terms subvector was set to one and the α for other subvectors varied between zero and one as shown in Table 1.

The most effective run was the one that expanded a query synset by any synset directly related to it and had $\alpha = .5$ for all added subvectors. Therefore, this strategy was used to produce the official routing queries from the final version of the annotated text. The scheme added an average of 24.7 words to a query vector (minimum 0, maximum 70), and an average of 20.2 (0, 66) words that are not part of the original text.

The average number of relevant documents retrieved at rank 100 for this run is 40.7 and at rank 1000 is 133.3; the mean "average precision" is .2984. In general, the individual query results are at or slightly above the median, with a few queries significantly below the median. Of more interest to this study is how the expanded queries compare to unexpanded queries. A plot of average recall versus average precision for these two runs is given in Figure 3. As can be seen, the effectiveness of the two query sets is very similar.

Since there was no way to evaluate the relative effectiveness of different expansion schemes for the ad hoc queries, the same same expansion scheme as was used for the official routing run — chains of length one for any relation type and all α's = .5 — was used for the ad hoc run. Furthermore, there could be no refining of which synsets to add, so only one version of synset-annotated text was produced. An average of 2.7 (minimum 0, maximum 6) synonyms was added to an ad hoc topic text. The expansion process added an average of 17.2 (0, 66) terms and 12.8 (0, 55) terms that are not part of the original text.

Siemens actually submitted two ad hoc runs. The first was the expanded queries with α's set to 0, a run that is equivalent to no expansion and is used as a base case. The second Siemens ad hoc run used the .5 α values. A plot of the effectiveness of the two ad hoc runs is given in Figure 4. The differences in effectiveness between unexpanded and expanded queries is even smaller for the ad hoc queries than it is for the routing queries. The average number of relevant documents retrieved at rank 100 is 46.9 for both the unexpanded and expanded queries. The average number of relevant documents retrieved at rank 1000 is 161.4 for the unexpanded queries and 161.3 for the expanded queries. The mean "average precision" is .3408 and .3397 respectively.

A possible explanation for the little difference made by expanding the queries is that the expansion parameters used were too conservative. To test this hypothesis, additional runs were made using the same set of synsets but allowing longer chains of links and/or using greater relative link weights (the α's). Table 2 lists the additional combinations tested using both the ad hoc queries versus the documents on disks one and two, and the routing queries versus the documents on disk 3. As was the case for the routing training runs, the strategy used for the official TREC-2 runs (all links of length one, α's = .5) was the most effective expansion strategy. The more aggressive expansion strategies did make larger differences in retrieval effectiveness compared to the unexpanded queries, but across the set of queries the aggregate difference was negative. Hence it is unlikely that the conservative expansion strategy is the reason for the lack of improvement.

4 Conclusion

The experimental evidence clearly shows this query expansion technique provides little benefit in the TREC environment. The most likely reason for why this should be so is the completeness of the TREC topic descriptions. Query expansion is a recall-enhancing technique and TREC topic descriptions are already large compared to queries found in traditional IR collections. Although most of the expanded queries did have some new terms added to them, the most important terms frequently appeared in both the original term set and the set of expanded terms. This had an effect on the relative weight of those terms in the overall similarity

Expansion by synonyms only	
orig terms α	synonyms α
1	.1
1	.3
1	.5
1	.8

Expansion by synonyms plus all descendents		
orig terms α	synonyms α	descendents α
1	.1	.1
1	.3	.1
1	.3	.3
1	.5	.1
1	.5	.3
1	.5	.5
1	.8	.1
1	.8	.3
1	.8	.5

Expansion by synonyms plus parents and all descendents			
orig terms α	synonyms α	descendents α	parents α
1	.1	.1	.1
1	.3	.1	.1
1	.3	.3	.3
1	.5	.1	.1
1	.5	.3	.1
1	.5	.3	.3
1	.5	.5	.1
1	.5	.5	.3
1	.5	.5	.5
1	.8	.1	.1
1	.8	.3	.1
1	.8	.3	.3
1	.8	.5	.1
1	.8	.5	.3
1	.8	.5	.5

Expansion by synonyms plus any directly related synset		
orig terms α	synonyms α	other α
1	.3	.1
1	.3	.3
1	.5	.1
1	.5	.3
1	.5	.5
1	.3	.5

Table 1: Combinations of expansion strategies and relation weights tested.

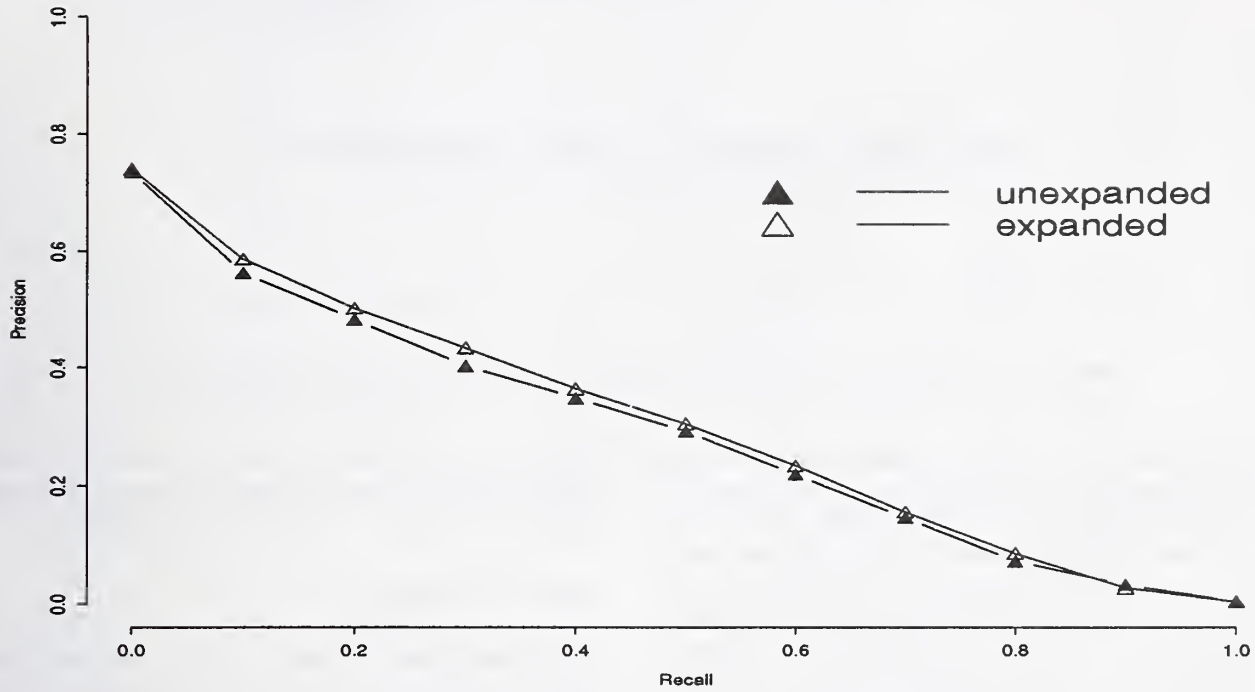


Figure 3: Effectiveness of expanded versus unexpanded routing queries on test documents (disk 3).

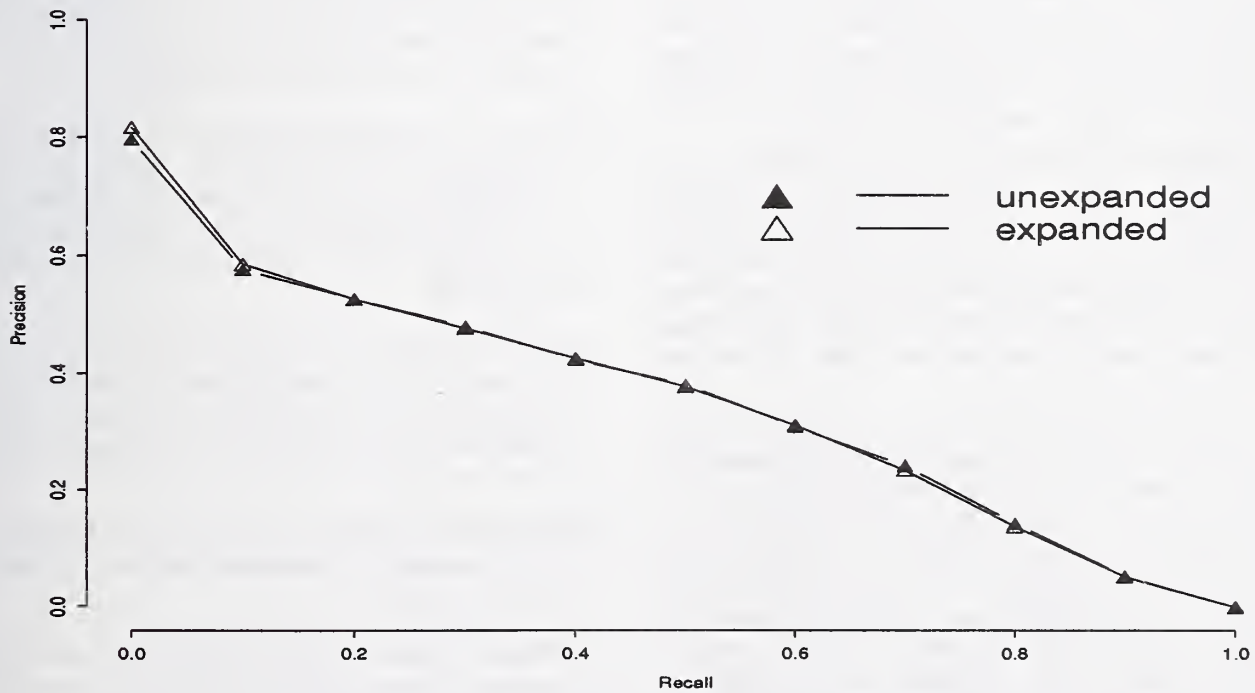


Figure 4: Effectiveness of expanded versus unexpanded ad hoc queries.

Expansion by synonyms plus parents and all descendents			
orig terms α	synonyms α	descendents α	parents α
1	.5	.5	.5
1	1	.5	.5
1	1	1	1
1	2	1	1

Expansion by synonyms plus any directly related synset		
orig terms α	synonyms α	other α
1	0	0
1	.5	.5
1	1	1

Table 2: Additional combinations of expansion strategies and relation weights tested.

computed for a document, especially when some important terms had no corresponding synset. Topic 112 provides an example of this effect. The topic is concerned about the world-wide investment in biotechnology. I added synonym sets for *investment* and *capital* to the topic. WordNet does not contain *biotechnology*, although it does contains *biomedical_science*. Thus, I also added the *biomedical_science* synset and a synset containing *gene*. The resulting expanded query performed significantly worse than the unexpanded version (33 relevant retrieved in the first 100 versus 52 relevant retrieved). The problem is that the expanded query places too much emphasis on money and not enough on biotechnology. Thus these results indicate that simply recognizing which are the important concepts in a query statement is not sufficient to ensure improved retrieval performance. An expansion procedure must also preserve the relative weights of those concepts.

Another possible explanation is that WordNet is not suited for this task — it was not designed to be used in this manner and it may not contain the necessary links. Even if this is true, however, it is unlikely that any other broad-coverage knowledge base would be better suited. The success of relevance feedback and other routing techniques suggests that the most useful relations are specific and idiosyncratic.

A second goal of this work was to characterize the effectiveness of different types of lexical relations when used to expand a query. Assuming the set of words to be expanded is well chosen, any closely related word — regardless of the type of relation — may be a good additional word. Wang et al. reached a similar conclusion [6]. Nevertheless, an added word should be weighted less than the original word that caused it to be included. Runs in which added words were equally or more heavily weighted than original words were consistently less effective than the more conservatively weighted runs. Similarly, runs that added words that were loosely related to original words (i.e., when long paths of links were followed) were consistently less effective than runs that used only near relatives.

Acknowledgements

Geoff Towell carefully read a draft of this paper and suggested changes that improved its presentation and clarity.

References

[1] Chris Buckley. Implementation of the SMART information retrieval system. Technical Report 85-686, Computer Science Department, Cornell University, Ithaca, New York, May 1985.

[2] Chris Buckley, Gerard Salton, and James Allan. Automatic retrieval with locality information using SMART. In D. K. Harman, editor, *Proceedings of the First Text REtrieval Conference (TREC-1)*, pages 59–72. NIST Special Publication 500-207, March 1993.

[3] Edward A. Fox. *Extending the Boolean and Vector Space Models of Information Retrieval with P-norm Queries and Multiple Concept Types.* PhD thesis, Cornell University, 1983. University Microfilms, Ann Arbor, MI.

[4] George Miller. Special Issue, WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4), 1990.

[5] Ellen M. Voorhees and Yuan-Wang Hou. Vector expansion in a large collection. In D. K. Harman, editor, *Proceedings of the First Text REtrieval Conference (TREC-1)*, pages 343–351. NIST Special Publication 500-207, March 1993.

[6] Yih-Chen Wang, James Vandendorpe, and Martha Evens. Relational thesauri in information retrieval. *Journal of the American Society for Information Science*, 36(1):15–27, January 1985.

TREC2 Document Retrieval Experiments using PIRCS

K.L. Kwok & L. Grunfeld

Computer Science Dept., Queens College, CUNY
Flushing, NY 11367. email: kklqc@cunyvm.cuny.edu

Abstract

We performed the full experiments, using our network implementation of component probabilistic indexing and retrieval model. Documents were enhanced with a list of semi-automatically generated two-word phrases, and queries with automatic Boolean expressions. An item self-learning procedure was used to initiate network edge weights for retrieval. Initial results submitted were above median for ad hoc, and below median for routing. They were not up to expectation because of a bad choice of high-frequency cut-off for terms, and no query expansion for routing. Later experiments showed that our system does return very good results after correcting the earlier problems and adjusting some parameters. We also re-design our system to handle virtually any number of large files in an incremental fashion, and to do retrieval and learning by initiating our network on demand, without first creating a full inverted file.

1. Introduction

In TREC1 our system called PIRCS (acronym for Probabilistic Indexing and Retrieval -Components- System) took part as Category B participant, handling only the 0.5 GB Wall Street Journal collection because both our software and hardware were not sufficient for the full set of text files. In TREC2, we participated in category A. However, during a large portion of the time period we have to face fairly uncertain and sometimes difficult conditions. Plans to install a dedicated SPARC10 workstation and associated large memory and disk drives did not materialize until about three weeks from the deadline. Before this period, the SPARC2 workstation that we have been using was also shared with other users during the semester. Certain things that we wished to do were not done, and corners were cut to fit programs and data in the existing system. Much of our time was spent revamping our software to be more efficient in space and time utilization. Our focus remains as in TREC1, namely, to improve representations of documents and queries, to test different learning methods and to combine different retrieval methods to improve final ranked retrieval output. Section 2 summarizes our retrieval network; Section 3 discusses our improved system design; Section 4 is on item representation; Sections 5&6 are about our learning and retrieval procedures; Section 7 discusses the results we submitted and Section 8 contains results of our later experiments. Section 9 follows with the conclusion.

2. A Retrieval Network in PIRCS

Our retrieval process is based on a three layer Q-T-D (Query-Term-Document) network, details of which are given in [KwPK93,Kwok9x]. Here we give a review. From Fig.1, DTQ query-focused retrieval means spreading an initial activation of 1 (one) from a document d_i towards query q_a and gated by intervening edges w_{ki} and w_{ak}. The resultant activation received at q_a is $W_{i/q} = \Sigma_k w_{ak}*w_{ki}$, and is the retrieval status value (RSV) of d_i with respect to q_a. When activation initiated at q_a spreads towards d_i, we obtain activation received at d_i equals to $W_{i/d} = \Sigma_k w_{ik}*w_{ka}$, and is our QTD document-focused RSV for d_i . Combining the two additively: $W_i = W_{i/q} + W_{i/d}$ gives our basic retrieval ranking function. Edge weights w_{ka}, w_{ki} represent items (q_a or d_i) acting on terms t_k and reflect usage of terms within items. Edge weights w_{ak}, w_{ik} (representing t_k acting on q_a or d_i) embed Bayesian inference and are initialized based on a component consideration of probabilistic indexing and retrieval. These weights can improve via a learning process when relevant documents are known to queries and vice versa: DTQ query-focused training when we know the set of documents and their components relevant to a query, and QTD document-focused training when we know the set of queries and their components relevant to a document. Query-focused training prepares queries to match new similar documents better, while document-focused training helps documents to match new similar queries better. With learning capability, the net behaves and can be viewed as a superposition of two 2-layer direct-connect artificial neural networks, one in each direction. If a boolean expression for query q_a is known, it can also be represented as a tree and hung onto the net as shown in Fig.2. Edge weights from

the net are used to initialize the tree leaf nodes, from which activation spreads to the query root node. Processing at the nodes implements soft-Boolean evaluation [SaFW83] in the DTQ direction resulting in a third RSV: S_t. The RSVs are latter combined for ranking retrieval outputs.

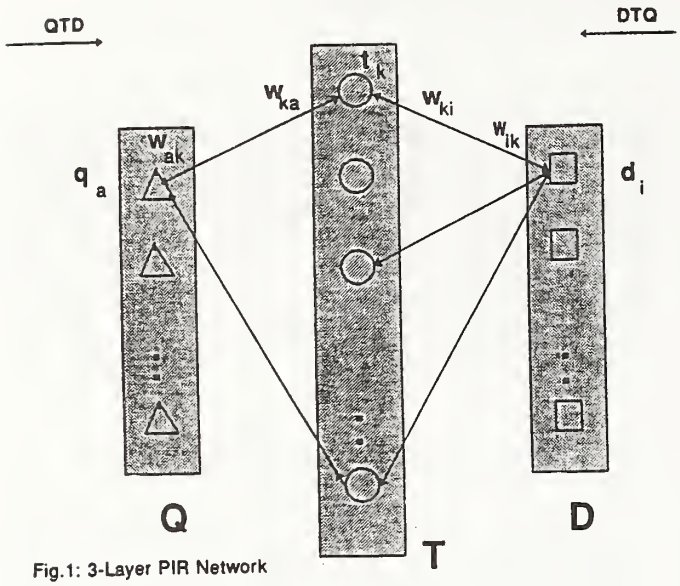


Fig.1: 3-Layer PIR Network

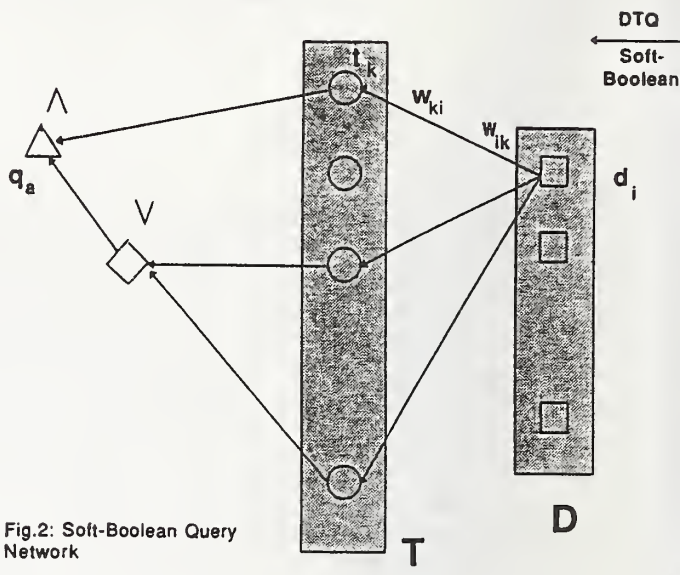
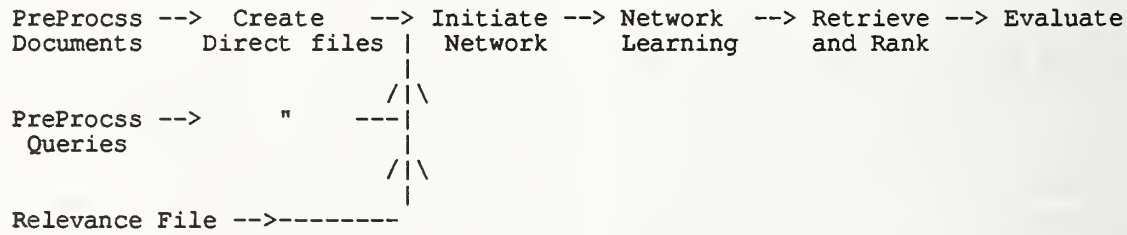


Fig.2: Soft-Boolean Query Network

3. System Design

Our previous software for TREC1 has extraneous processing that produces several intermediate files for other purposes. These consume a lot of disk space for large scale collections. We spent a substantial amount of time to revamp our system, resulting in a more streamlined flow-chart as follows:



In addition, we also took the opportunity to re-design our system resulting in several innovative approaches described in the following subsections:

3.1 Full Inverted File Eliminated

It is useful to view a textbase as a document by term matrix. If one stores the matrix rowwise, we call it a direct file, in contrast to an inverted file which stores the matrix columnwise. The inverted file is useful to support fast retrieval while the direct file is useful for feedback learning and query expansion when given certain documents being relevant to certain queries. In addition, the raw textfile is useful for display purposes after a retrieval ranked list is produced. If one assumes that each of these three files are approximately equal in size of N bytes, we need a minimum of 3N bytes, which is quite substantial. Removing the raw text may not win user support during display unless the direct file encodes all stopwords, punctuations and paragraph structures of the original. Most systems delete the direct file and re-produce a subset of it from raw text when needed. This results in a requirement of 2N bytes. We choose, however, to keep the direct file and produce our network with respect to the queries dynamically as needed without first producing an inverted file first. Our network actually contains both direct and 'inverted' data. It resides in memory to support learning and retrieval. By this

approach we achieve several objectives for our system: a) reduce the 'dead' time between a collection being acquired and its availability for searching, since the full inverted file is not produced; b) satisfy the 2N bytes of space requirement; c) support fast feedback learning and query expansion by having the direct file available. The price we pay is to have to create the network dynamically, unlike the full inverted file which is produced once. However, since the full inverted file is too large to reside in memory, reading parts of it in for retrieval is also time consuming.

3.2 Reduced Network Size

We produce our network in memory according to the queries under attention and the terms used in them. Since memory is limited, documents are divided into subcollections (Section 3.3) and queries are linked in five to ten at a time. We define the active term set (ATS) of a network as the set of terms used by the current queries and their feedback documents if any. The latter will provide terms for query expansion. Only edges that connect items to the active term set are initiated in the network, reducing network space requirements substantially. With the current implementation for 1GB subcollection and 7 queries, the node and edge files together take about 40 MB. Using clock time as a measure, producing the network requires about 40 minutes. Learning is fast, about 2 minutes, and retrieval and ranking another 8 minutes. We hope that further improvements in design and faster hardware in the future can improve these figures substantially.

3.3 Master and Subcollection File Structure

We view the TREC experiments as document retrieval from multiple collections, but reporting retrieval results in one single ranked list of documents for each topic (query). Although only four or five collections such as Wall Street Journal, Associated Press, etc. are given in TREC, in reality it could be many more. We consider three methods in file design to approach the problem:

a) A centralized approach where all documents from all collections are processed as if from a single document stream, producing a centralized dictionary containing full usage statistics and a giant direct file. From these, networks can be initialized. The idea is simple, but it has drawbacks in that eventually the direct file would exceed file/disk size limitations and software has to be designed to handle data crossing file/disk boundaries. Moreover, it is inherently fragile to create single files of this size. The advantage of this approach is that RSVs calculated are directly comparable for all documents and a single ranked list is produced without difficulty.

b) An independent collections approach where each source collection of about 0.5-1.0 GB say, forms a textbase with its own local dictionary and direct file, for network initiation and retrieval. One simply repeats the process for as many collections as necessary. This is the preferred approach, and if one has n processors and sufficient disk space, n separate textbases can be created for learning and retrieval in parallel saving substantial time. The problem is how to combine the retrieval lists from each into a single ranked list, since each textbase has its own term usage statistics and calculates RSVs for ranking within its own environment. Classical Boolean retrieval and coordinate matching pose no problem. Some retrieval strategy may produce RSVs that are comparable across collections in theory; but after approximations are taken, it is questionable that this is still true. Similar problem exists for retrieval from distributed databases such as the WAIS environment.

c) For TREC2 we settle on a hybrid subcollections approach, treating each source as a subcollection within a master. We create a master centralized dictionary as in a) capturing full usage statistics serving all the subcollections, and create separate direct files for each subcollection as in b). The central dictionary has about 620,000 unique terms after processing 2 GB from Disk1 and Disk2, and is relatively small. It captures global term usage statistics, while the individual direct files capture local usage statistics within items. Separate networks are then created for each subcollection with edge weights based on the correct global and local statistics as in a), assuring that retrieval lists contain RSVs that are directly comparable. This approach combines the advantages of both a) and b), and can also function in a parallel distributed environment.

4. Item Representation

As in TREC1, a number of preprocessing mainly for the purpose of improving the representations of documents and queries are done as follows:

4.1 Vocabulary Control

In addition to a manual stopword list of about 630 words and another 528 manually identified 2-word phrases, we also process samples from Disk1 and Disk2 using all five source types (WSJ, AP, FR, ZIFF and DOE, of about 100MB each) to produce two-word phrases based on adjacency within sentence context. Our objective is to remedy losses of recall and precision due to the removal of high frequency terms. Our criteria for phrases is that each word pair must have a frequency of 40, and either one or both components must be high frequency (>=10000). A casual scan of the resultant list generated led us to remove

some obvious non-content phrases (such as 'author describ', paper attempt', 'two way'). The result is 13,787 pairs. These plus the previously prepared manually list (which contains mostly of phrases with at least one stopword as well as some phrases identified in the 'topics') are then treated as if they were single index terms to be identified during document and query processing [BuSA93]. They have their own global and local usage statistics, and can improve individual collection retrieval effectiveness from a few to 10%.

After documents are processed, we invoke Zipf's law to remove low (=4) and high frequency (=16000) words from being used for representation. Low frequency terms lead to too many nodes, and high frequency terms lead to too many edges, both consuming valuable memory space. Unfortunately, our high frequency cut-off of 16000 was a bad choice. In fact, we used high = 12000 for routing because at that earlier period, we were short of resources. The effect is that our queries become too short and many useful terms (such as 'platform' in query #80, 'crimin' for criminal in query #87, etc.) are screened out. We discover that this is a major factor in our disappointing results. Later experiments use a high cut-off of 50000.

4.2 Subdocuments

As in TREC1 we segment documents into subdocument units to deal with the problems of WSJ documents having multiple unrelated stories, and long documents in general. A really long document is FR89119-0111 which has 400748 words. Our criteria is to break documents either on story boundaries or on the next paragraph boundary after a run of 360 words for all collections. We have not found convenient story boundary indicators in other collections as in WSJ (which uses three dashes '---' on a line). With this scheme, the total number of subdocuments from Disk 1&2 becomes 1,281,233 compared with an original 742,611.

After the TREC2 deadline, we have the resources to investigate the effects of subdocuments on retrieval. Experiments were performed on individual subcollections WSJ1, FR2, FR1 and AP2, using segmentation sizes of 360, 720 and 1080 words. For WSJ1, we further break on story boundaries only. Results are tabulated in Table 1. It appears that for the abnormally long documents of FR, breaking into subdocuments is definitely worthwhile, achieving improvements of over 20% compared with no segmentation. However, for the newswire documents of

	AsIs	Break on "Stories"	1080	720	360
WSJ1					
Av11	0.421	0.432	0.428	0.424	0.418
# of docs	98733	127151	134819	149611	193881
FR1					
Av11	0.289		0.351	0.354	0.354
# of docs	26207		50055	64650	108374
FR2					
Av11	0.333		0.372	0.420	0.421
# of docs	20108			51607	86787
AP2					
Av11	0.423		0.423	0.404	0.414
# of docs	84678		85616	95867	146354

Table 1: Document Segmentation Av11 Precision using 50 Queries Q2

WSJ1 and AP2, subdocuments have marginal performance effects: a little better for large chunk sizes, and a little worse for small chunks. It seems that a chunk size of about 720-1000 words would get the benefits of both types. Using different chunk sizes for different collections would probably not be worth the effort.

We like to point out that other than effectiveness, considerations such as isolating relevant sections for output display and for more precise feedback judgment would also make document segmentation worthwhile. In particular, a number of long documents in feedback for query expansion would easily overload memory space in our network.

4.3 Queries

Topics are preprocessed to remove introductory phrases and

non-content terms based on word sequence patterns. We continue to use words from the title, description, narrative and concept sections of the topics to form queries. Experiments without the narrative section show slight decrease in performance for our system in contrast to [Crof93]. We also try to produce automatically Boolean expressions as queries from the description and concepts sections. This is done by using punctuations to delineate phrases and ANDing the words within them. Phrases are then ORed. This is a very crude way of getting Boolean expressions for later soft-Boolean retrieval processing.

5. Learning Procedures

In our network of Fig.1 the edge weights determine the retrieval outcome. w_{ki} and w_{ka} captures the proportion of term t_k in d_i or q_a. They are fixed and obtained from the manifestation of terms in the respective items. w_{ak} (and w_{ik}) has a log odds factor, and an Inverse Collection Term Frequency (ICTF) factor which is regarded as a constant for a term t_k, as follows:

$$w_{ak} = \ln [r_{ak}/(1-r_{ak})] + ICTF_k \qquad (1)$$

Here r_{ak} is the conditional probability that given relevance to q_a that term t_k occurs, and needs to be learnt. It is unknown unless one has a sample of relevant documents to q_a. This is not applicable to initial ad hoc retrievals where a document collection is being processed against a new query with no known results. Relevance feedback information can remedy this later, but it is not available at the beginning. One way is to ignore the log odds factor in Eqn.1, as done in our TREC1 experiments, resulting in ICTF weighting. A better way, which we use for TREC2, is to include item self-learning to determine r_{ak} and initiate the term weights. This is shown in Fig.3 and is based on the following argument. Consider a document d_i containing certain concepts and topics. Imagine this author wishing to inquire the textbase for the same topics as this document s/he has written; what query would be most suitable? Naturally the author's own words in the document can serve as the 'query', and there is also known to be one relevant item to this 'query' in the collection, viz. the document itself. In other words, every item is assumed to be self-relevant. One relevant item is however not sufficient for estimating r_{ak}. Our method is to consider each document as constituted of many independent conceptual components each being described by a list of terms. We therefore work in a component universe rather than in the document collection. Components can be units such as sentences or phrases, but we have used single terms for simplicity. Right from the start then, even without any relevance feedback, we can divide the component universe into two parts: one set relevant, and the other non-relevant to each the 'query'. Standard probabilistic retrieval theory now enables us to define this 'query' optimally -- meaning that the defined 'query' will rank its set of relevant components optimally with respect to the other components when used for retrieval. The definition of this 'query' (i.e. its terms and weights) becomes the initial indexing representation of the document, and it is used in our ad hoc QTD retrieval. This is the principle of document self-recovery introduced in [Kwok90] and implemented as a self-learning process in a network [Kwok89,9x] shown in Fig.4. One can argue that this relevant set of components from one document is too small. But that is all the information one has at this stage. Previous experiments [Kwok90] show that this kind of weighting can outperform ICTF weight by a few percent. Moreover, our network self-learning parameters can be adjusted to provide a smooth transition from ICTF to full self-learn weights, or any value in between. We invoke the

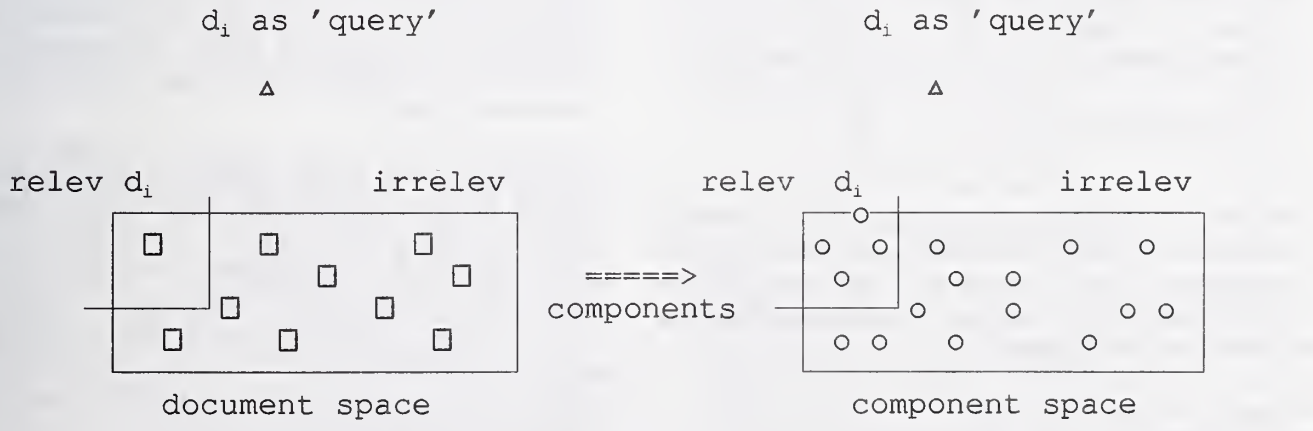


Fig.3 Item Self-Learn Using Its Own Self-Relevant Components

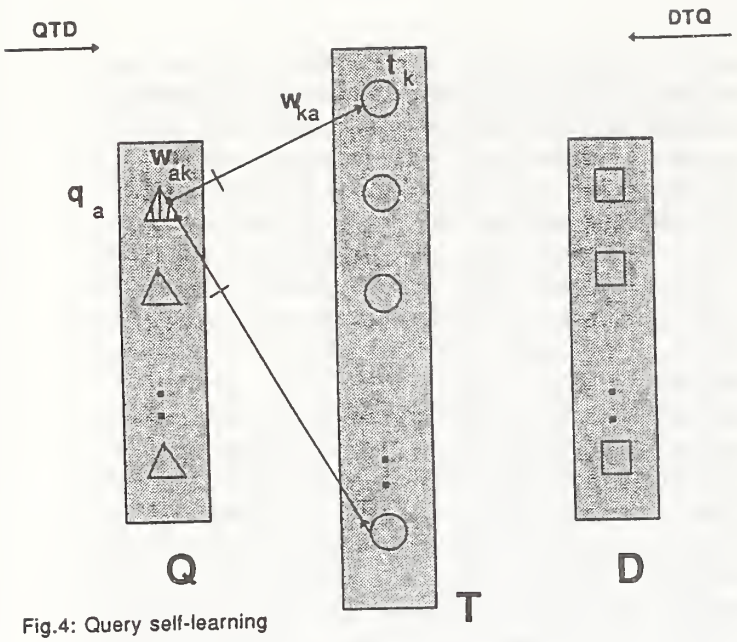


Fig.4: Query self-learning

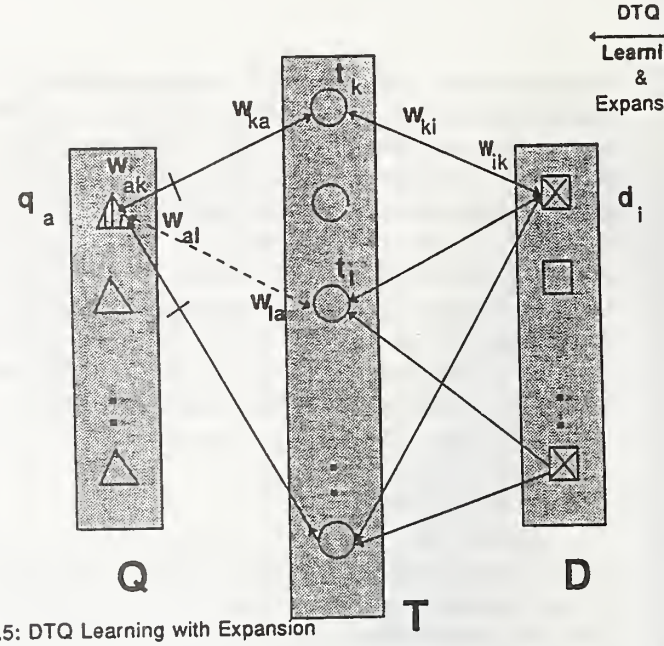


Fig.5: DTQ Learning with Expansion

same self-learning procedure for a query by adding the query to the collection as a 'document' temporarily, resulting in self-learn initial weights for the indexing representation of the query. This weight is used in our ad hoc DTQ retrieval. The bottom line is that this component consideration enables the probabilistic model to self-bootstrap, allows term frequencies in items to be employed instead of 'binary', and takes into account of query-focused and document-focused retrieval in a cooperative fashion.

In the case of routing retrieval, relevant documents are available for training each topic q_a. These documents are again considered as constituted of components, and are used in the network to estimate r_{ak}. The estimate should be better than self-learning because the sample size of relevant components is much larger. Our learning algorithm updates the conditional probability r_{ak} (and r_{ik}) as follows (Fig.5):

$$\Delta r_{ak} = \eta_Q * (x_k - r_{ak}^{old}) \qquad (2)$$

Here, η_Q is a learning rate for training on the query side and x_k is the average activation deposited on term t_k by the given relevant set. If a term t_l not in the original query is highly activated by the relevant set, it can also be linked to q_a with edge weights given by:

$$w_{la} = \alpha * x_l$$
$$r_{al} = \beta * \eta_Q * x_l \qquad (3)$$

This implements query expansion as was also done in TREC1.

6. Retrieval Methodology

To satisfy TREC2 requirements, we submitted results as named in the following:

pircs1: routing, no training;
pircs2: routing, with training from Disk1 relevants, Disk2 not used and no query expansion;
pircs3: ad hoc, no soft-Boolean added;
pircs4: ad hoc, with soft-Boolean added.

Routing allows training the old Q2 topic set (topics 51-100) before doing retrieval on new Disk3 documents. Disk3 term usage statistics are not used. Ad hoc retrieval involves using new Q3 topic set (topics 101-150) to do retrieval on old documents in Disk1 and Disk2. All our queries are automatically constructed. We did not perform feedback experiments using Q3.

For baseline routing (pircs1) and ad hoc (pircs3) retrievals, we use the item self-learn (SL) edge weights. Routing pircs2 denotes retrieval based on further learning from known relevant samples and represent improvements from pircs1. There can be hundreds of known relevants for each of the Q2 topic set from documents in Disk1 and Disk2 as given from the results of TREC1. One way of employing them is to do a retrieval (ranking) of Disk1 and Disk2 documents, and then make use of the first n (say n=100) relevant documents, as for feedback learning. However, we did not have enough resources to create a network and do retrieval (ranking) on 2GB of documents at that time and have to settle on a simplified strategy. First, we decided to use Disk1 only (1 GB) for training. Secondly, we believe

a retrieval (ranking) of Disk1 is still expensive and perhaps not necessary; rather we just select 'nonbreak' relevants from Disk1 for training. 'Nonbreak' means documents that do not get split into multiple subdocuments based on our criteria given in Section 3. The idea is that the quality of documents for training is important, and short relevants are the choice. They may not be those ranked early during a retrieval. With these simplifications, a network is produced with Disk1, the query-term edges are trained, and then stored for later routing retrieval using Disk3.

Ad hoc pircs4 denotes retrieval based on combining the baseline pircs3 with a soft-boolean retrieval. The pircs4 ranking formula becomes $r*W_i+s*S_i$ (see definitions in Section 2). Our Boolean expressions for queries are produced automatically as discussed in Section 4.3, and edge weights are used to initiate the leaf nodes of the Boolean expression tree.

7. Discussion of Submitted Results

From our routing retrieval table in the master appendix of this volume, it can be seen that pircs2 improves over pircs1 by about 7% based on average non-interpolated precision (.266 vs .249) and about 3.8% based on relevants retrieved (6135 vs 5913), showing that our simplified method of using only the Disk1 'nonbreak' training documents still works. We did not do a retrieval and rank. Compared with the other sites, our result is below median both using the average non-interpolated precision for individual queries (18 better, 2 equal and 30 below median), and using the relevants retrieved at 100 documents (18 better, 8 equal and 24 below median). If we assume the existence of an overall 'maxi-system' that produces the best non-interpolated precision values among all sites for all 50 queries, then its average precision over all queries is 0.5054 and 8348 relevants retrieved. Our pircs2 achieves only .266/.505 = 52.7% of the average precision but 6135/8348 = 73.5% of the relevants retrieved.

From the ad hoc retrieval table in the appendix of this volume it can be seen that pircs4, which is pircs3 combined with automatic soft-Boolean retrieval, improves over pircs3 only by about 1%. Processing time however increases substantially. Our automatic Boolean expressions are crudely formed; manual Boolean queries may do better. Compared with other sites, our result is above median both using the average non-interpolated precision for individual queries (34 better, 2 equal and 14 below median), and using the relevants retrieved at 100 documents (36 better, 4 equal and 10 below median). The 'maxi-system' has an average precision over all queries of 0.4354 and 9027 relevants retrieved. pircs4 achieves about 0.298/0.435 = 68.5% of this best precision value and 7464/9027 = 82.7% of the relevants retrieved. They are much better than for routing. It would be most useful and interesting if one can choose the best reported result for each query before the answers are known. For these experiments our high frequency term cut-off is 16000, which is still too low. The next Section discusses our later results.

8. Further Experimental Results

After the TREC2 Conference, we decided to repeat both experiments. We realize that our disappointing results are due to several factors: 1) bad high frequency term cut-off leading to insufficient representation; 2) no query expansion; 3) insufficient training samples; and 4) parameters need tuning. Except for 4) these are remedied as follows: high-frequency cut-off is set at 50000, learning for routing is done from both disk1 and disk2 and only documents that 'break' into six or less subdocuments are used, and query expansion is also done. The runs are named in Table 2 as:

pircs5: routing, with learning but no query expansion;
pircs6: routing, query expansion level of 20;
pircs7: routing, 'upperbound', no expansion;
pircs8: ad hoc without Boolean queries.

As in TREC1, our query expansion level of 20 actually adds less than 20 terms because some of the top-ranked terms may already appear in the query. It can be seen that results are substantially better than those in Section 7. In particular, pircs6 routing with query expansion have average precision of 0.355 and the number of relevants retrieved are 7476 out of 10489. These are 12% and 5% respectively better than pircs5 (0.318, 7098): routing with learning but no query expansion, and achieving 70.3% and 89.6% of the maxi-system values. The same average precision value and relevants retrieved for ad hoc retrieval pircs8 are 0.344 and 8279 out of 10785, representing 79% and 91.7% of the ad hoc maxi-system respectively. At 20 documents retrieved, the precision values for routing and ad hoc are respectively 0.583 and 0.564. This means that averaging over 50 queries, out of the first 20 retrieved over 11 are relevant. Considering the size of these textbases, these are quite good results. These numbers are user-oriented, and users naturally hope to see 100% precision. As discussed in TREC1, from a system point of view the precision at n documents retrieved should not be compared to the theoretical value of 1.0, but to an operational precision value x/n if the total number of relevants x for a query is less than n. For example, at n=100 documents retrieved 20 routing and 16 ad hoc queries have total relevants x less than 100. The operational maximum precision averaged over 50 queries for routing is only 0.8, and that for ad hoc is 0.871. At 100 documents, routing pircs6 value of 0.439 and ad hoc pircs8 value of 0.468 therefore achieves 54.9%

	Revised Routing					New Ad hoc
	pircs5	pircs6	%	pircs7	%	pircs8
Total number of documents over all 50 queries						
Retrieved:	50000	50000		50000		50000
Relevant:	10489	10489		10489		10785
Rel_ret:	7098	7476	+ 5.3	7385	+ 4.0	8279
Interpolated recall - precision averages:						
0.0	.765	.814	+ 6.4	.793	+ 3.7	.823
0.1	.554	.628	+13.3	.594	+ 7.2	.561
0.2	.491	.546	+11.2	.534	+ 8.6	.505
0.3	.421	.469	+11.4	.469	+11.4	.456
0.4	.380	.413	+ 8.7	.416	+ 9.5	.417
0.5	.336	.363	+ 8.0	.368	+ 9.5	.368
0.6	.270	.310	+14.8	.314	+16.3	.320
0.7	.212	.240	+13.2	.241	+13.7	.246
0.8	.150	.165	+10.0	.171	+14.0	.160
0.9	.079	.099	+25.3	.098	+24.0	.087
1.0	.014	.006	-57.1	.015	+ 7.1	.012
Average precision (non-interpolated) over all rel docs						
	.318	.355	+11.6	.350	+10.1	.344
Precision at						
5 docs:	.600	.660	+10.0	.624	+ 4.0	.612
10 "	.582	.632	+ 8.6	.598	+ 2.7	.572
15 "	.545	.617	+13.2	.572	+ 5.0	.573
20 "	.527	.583	+10.6	.563	+ 6.8	.564
30 "	.507	.553	+ 9.1	.534	+ 5.3	.540
100 "	.402	.439	+ 9.2	.427	+ 6.2	.468
200 "	.334	.360	+ 7.8	.353	+ 5.7	.396
500 "	.222	.238	+ 7.2	.232	+ 4.5	.264
1000 "	.142	.150	+ 5.6	.148	+ 4.2	.166
R-precision						
Exact	.358	.385	+ 7.5	.385	+ 7.5	.378

Table 2: Revised Routing and New Ad Hoc Retrieval Results

and 53.7% of the operational maximum. The R-precision Exact calculates the precision value at x retrieved, where x is the known number of relevants for each query, and can be compared with the theoretical value of 1.0.

The 'Upperbound' retrieval pircs7 (suggested by Sparck-Jones) means performing learning from the known Disk3 (not Disk 1&2) relevant documents before retrieval. In other words, we assume the answer documents are known for training and represent the best that probabilistic theory can provide using our system. This is however not the true upperbound [Spar79] for retrieval from Disk3, because the vocabulary and usage statistics are still those of Disk 1&2. The vocabulary is retained for comparison with routing results. pircs7 retrieval achieves average precision of 0.350, which improves over pircs5 (training from Disk 1&2) by about 10% in average precision and about 4% in relevants retrieved. Of course in real life, the answer documents are not known; but it is interesting to note that query expansion using Disk 1&2 documents can provide similar performance, showing the importance of query expansion.

We later concentrate on routing and discover that additional gains can be achieved by fine tuning of the parameters in our model. For learning: 1) we find that our original method of using only 'nonbreak' documents in the given set of relevants actually outperforms other document selection strategies including using all relevants, 'break six' or

	New Routing Results										
	No Train.	Trained Exp 0	%	Trained Exp 20	%	Trained Exp 40	%	Trained Exp 80	%	Trained Exp 100	%
Total number of documents over all 50 queries											
Retrieved:	50000	50000		50000		50000		50000		50000	
Relevant:	10489	10489		10489		10489		10489		10489	
Rel_ret:	**6551**	6961	+ 6.0	7496	+14.0	7646	+17.0	7695	+17.0	7712	+18.0
Interpolated Recall - Precision Averages:											
0.00	0.7200	0.7480	+ 4.0	0.8471	+18.0	0.8475	+18.0	0.8646	+20.0	0.8660	+20.0
0.10	0.5124	0.5815	+13.0	0.6645	+30.0	0.6751	+32.0	0.6810	+33.0	0.6801	+33.0
0.20	0.4431	0.5254	+19.0	0.5981	+35.0	0.6116	+38.0	0.6135	+38.0	0.6115	+38.0
0.30	0.4016	0.4728	+18.0	0.5371	+34.0	0.5413	+35.0	0.5465	+36.0	0.5452	+36.0
0.40	0.3486	0.4402	+26.0	0.4751	+36.0	0.4774	+37.0	0.4829	+39.0	0.4878	+40.0
0.50	0.2970	0.3862	+30.0	0.4167	+40.0	0.4288	+44.0	0.4229	+42.0	0.4214	+42.0
0.60	0.2382	0.3048	+28.0	0.3496	+47.0	0.3681	+55.0	0.3699	+55.0	0.3690	+55.0
0.70	0.1945	0.2430	+25.0	0.2772	+43.0	0.2880	+48.0	0.2815	+45.0	0.2843	+46.0
0.80	0.1284	0.1865	+45.0	0.1911	+49.0	0.1937	+51.0	0.1999	+56.0	0.2005	+56.0
0.90	0.0740	0.0860	+16.0	0.1130	+53.0	0.1144	+55.0	0.1219	+65.0	0.1238	+67.0
1.00	0.0119	0.0187	+57.0	0.0140	+18.0	0.0107	-10.0	0.0114	- 4.0	0.0171	+44.0
Average precision (non-interpolated) over all rel docs											
	0.2905	**0.3517**	**+21.0**	**0.3962**	**+36.0**	**0.4050**	**+39.0**	**0.4084**	**+41.0**	**0.4095**	**+41.0**
Precision at:											
5 docs:	0.5600	0.5760	+ 3.0	0.6960	+24.0	0.7160	+28.0	0.7320	+31.0	0.7280	+30.0
10 docs:	0.5440	0.5820	+ 7.0	0.6880	+26.0	0.6860	+26.0	0.6980	+28.0	0.7000	+29.0
15 docs:	0.5173	0.5627	+ 9.0	0.6573	+27.0	0.6707	+30.0	0.6800	+31.0	0.6813	+32.0
20 docs:	0.4910	0.5510	+12.0	0.6470	+32.0	0.6540	+33.0	0.6630	+35.0	0.6610	+35.0
30 docs:	0.4653	0.5313	+14.0	0.6147	+32.0	0.6173	+33.0	0.6240	+34.0	0.6267	+35.0
100 docs:	**0.3698**	**0.4396**	**+19.0**	**0.4824**	**+30.0**	**0.4930**	**+33.0**	**0.4974**	**+35.0**	**0.5002**	**+35.0**
200 docs:	0.3049	0.3562	+17.0	0.3887	+27.0	0.3945	+29.0	0.4002	+31.0	0.4004	+31.0
500 docs:	0.2038	0.2241	+10.0	0.2452	+20.0	0.2490	+22.0	0.2500	+23.0	0.2498	+23.0
1000 docs:	0.1310	0.1392	+ 6.0	0.1499	+14.0	0.1529	+17.0	0.1539	+17.0	0.1542	+18.0
R-Precision (precision after R (= num_rel for a query) docs retrieved):											
Exact:	**0.3346**	**0.3942**	**+18.0**	**0.4251**	**+27.0**	**0.4283**	**+28.0**	**0.4281**	**+28.0**	**0.4291**	**+28.0**

Table 3: New Routing Results at Several Query Expansion Levels

ranking and selecting the n best. Moreover, these 'nonbreak' documents total only 5225, less than 1/3 of 16114 relevants used and is therefore very efficient. (There are actually 16400 relevants from Disk 1&2, but during processing a small percentage was lost). 2) All edges and their weights on the query side of the network are defined by the activations deposited by the relevant documents; this means the original query plays no part in their definition. 3) Negative edge weights are set to small positive weights of 0.1. For retrieval: 4) after ranking, several subdocuments of the same document ID may rank high, and we combine their largest three RSVs in the ratio of 1:0.2:0.05 as the single reported RSV for the whole document. Previously we ignored the third, and the ratio for combining the largest two was different. We choose to stop at two or three subdocuments because noise from long documents may creep back. Such tuning of parameters led to the results in Table 3 for our latest routing results. We use the convention 'Trained Exp K' to denote query expansion level K, with K=0 meaning weight adaptation without adding new

terms. The 'No Train.' column shows results without using any known relevants for training and serve as the basis for comparison. It can be seen using the measure of average precison over all recall points that training without query expansion improves over no training by 21%, and training with expansion at the 40 level improves over the basis by about 39%. This measure, as well as the R-precision, seems to level off from expansion level 40 onwards. However the number of relevants retrieved improves from 6551 (no training) to 7712 (expansion level 100) in a monotone fashion. Higher query expansion level appears to improve the high-recall region of the precision-recall curve without materially affecting the low-recall region as observed in [Kwok9x] using the WSJ collection only. Precision values at the different cutoffs of documents retrieved seem to level off at the expansion level of 80. At 20 retrieved documents cutoff, we now achieve a precision over 0.65, meaning that more than 13 of the 20 documents retrieved are relevant on the average. The tuning of parameters give us over 10% additional improvements above those obtained in the revised routing results of Table 2. It appears that a query expansion level of 40 achieves a compromise between good effectiveness and good efficiency for our system. We did not do massive query expansion at high levels of 200 or more. However, the results are comparable to the best of those reported in the TREC2 conference.

9. Conclusion

We have upgraded our PIRCS system to use dynamic network creation for learning and retrieval, and to handle files in a master-subcollection design. The former approach allows us to eliminate full inverted file creation resulting in 2 x collection size space requirement, reduced 'dead' time for a collection to be searchable, and provide fast learning. The latter approach renders our system to be sufficiently flexible to handle a large number of files in a robust fashion, yet produce a retrieval ranked list as if all documents were in one file. Although our submitted results for TREC2 were not up to expectation because of insufficient resources at the time of the experiments, the reasons for the behavior of our system were isolated. New experiments show that PIRCS can provide highly competitive retrieval effectiveness in both ad hoc and routing environments.

Acknowledgment

Ms. Lu Chi-Ni provided the program for our two-word phrase detection. We would like to acknowledge the continual support of the Department Chairman and the Dean of Mathematics and Natural Science at Queens College throughout the project. This work is partially supported by a grant from ARPA and a PSC-CUNY grant #663288.

References

[BuSA93] Buckley, C, Salton, G & Allan, J (1993). Automatic retrieval with locality information using SMART. In: The First Text REtrieval Conference (TREC-1). Harman, D.K. (Ed.). NIST Special Publication 500-207. pp.59-72.

[Crof93] Croft, W.B (1993). The University of Massachusetts Tipster project. In: The First Text REtrieval Conference (TREC-1). Harman, D.K. (Ed.). NIST Special Publication 500-207. pp.101-105.

[KwPK93] Kwok, K.L, Papadopolous, L & Kwan, Y.Y. Retrieval experiments with a large collection using PIRCS. In: The First Text REtrieval Conference (TREC-1). Harman, D.K. (Ed.). NIST Special Publication 500-207. pp.153-172.

[Kwok90] Kwok, K.L (1990). Experiments with a component theory of probabilistic information retrieval based on single terms as document components. ACM TOIS 8:363-386.

[Kwok9x] Kwok, K.L (199x). A network approach to probabilistic information retrieval. Accepted for publication in ACM TOIS.

[SaFW83] Salton, G; Fox, E.A & Wu, H (1983). Extended boolean information retrieval. Comm. ACM 26:1022-1036.

[RoSp76] Robertson, S.E & Sparck Jones, K (1976). Relevance weighting of search terms. J. ASIS. 27:129-146.

[Spar79] Sparck Jones, K (1979). Experiments in relevance weighting of search terms. Info. Proc. Mgmnt. 15:133-144.

Combination of Multiple Searches

Edward A. Fox and Joseph A. Shaw
Department of Computer Science
Virginia Tech, Blacksburg, VA 24061-0106

Abstract

The TREC-2 project at Virginia Tech focused on methods for combining the evidence from multiple retrieval runs to improve retrieval performance over any single retrieval method. This paper describes one such method that has been shown to increase performance by combining the similarity values from five different retrieval runs using both vector space and P-norm extended boolean retrieval methods.

1 Overview

The primary focus of our experiments at Virginia Tech involved methods of combining the results from various divergent search schemes and document collections. We performed both routing and ad-hoc retrieval experiments on the provided test collections. The results from both vector and P-norm type queries were considered in determining the probability of relevance for each document in an individual collection. The results for each collection were then merged to create a single final set of documents that would be presented to the user.

2 Index Creation

This section outlines the indexing done with the document collection provided by NIST. Each of the individual collections was indexed separately as document vector files; limitations in disk space prohibited the use of inverted files and the creation of a single combined document vector file.

All processing was performed on a **DECstation 5000/25** with 40 MB of RAM using the 1985 release of the **SMART** Information Retrieval System [2], with enhancements from previous experiments as well as a new modification for our TREC-2 experiments.

The index files were created from the source text via the following process. First, the source document text provided by NIST was passed through a preparser to convert the SGML-like format to the proper format for the 1985 version of **SMART**. The extraneous sections of the documents were filtered out at this point. The TEXT sections of the documents, as well as the various HEADLINE, TITLE, SUMMARY, and ABSTRACT sections of the collections were indexed; all of the other sections were ignored. The subsections of the TEXT fields, where they existed, were considered as part of the TEXT field, with the subsection delimiters removed.

Table 1: **SMART** weighting schemes used for TREC-2.

SMART label	term_weight =
ann	$0.5 + 0.5 * \frac{tf}{max\_tf}$
bnn	1
mnn	$\frac{tf}{max\_tf}$
atn	$0.5 + \frac{tf}{2*max\_tf} * log(\frac{num\_docs}{coll\_freq})$
nnn	tf

The resulting filtered text was tokenized, stop words were deleted using the standard 418 word stop list provided with **SMART**, and the remaining non-noise words were included in the term dictionary along with their occurrence frequencies. Each term in the dictionary has a unique identification number. A document vector file was created during indexing which contains for each document its unique ID, and a vector of term IDs and term weights. The initially recorded weights can be changed based on one of several schemes after the indexing is complete. The various **SMART** weighting schemes referred to within this paper are summarized in Table 1. The dictionary size for each collection was approximately 16 MB, while the document vector files ranged from 31 MB to 124MB (see Table 2).

Table 2: Collection statistics summary. Text, Dictionary and Document Vector sizes in Megabytes.

Collection	Text	Dict.	Doc. Vectors	Total Doc.s
AP-1	266	16.0	120.2	84678
DOE-1	190	15.9	97.9	226087
FR-1	258	15.8	53.8	26207
WSJ-1	295	16.2	124.8	98735
ZIFF-1	251	15.7	88.4	75180
D1	1260	N/A	485.1	510887
AP-2	248	15.9	110.4	79923
FR-2	211	15.6	42.7	20108
WSJ-2	255	16.0	105.5	74520
ZIFF-2	188	15.4	63.6	56920
D2	902	N/A	322.2	231471
D1 & D2	2162	N/A	807.3	742358
AP-3	250	15.9	111.2	78325
PATN-3	254	15.6	31.3	6711
SJM-3	319	16.1	114.4	90257
ZIFF-3	362	16.0	109.8	161021
D3	1185	N/A	366.7	336314
Total	3347	N/A	1174.0	1078672

3 Retrieval

3.1 Queries

All of the queries were created from the topic descriptions provided by NIST. Two types of queries were used, P-norm extended boolean queries and natural language vector queries. A single set of P-norm queries was created, but was interpreted multiple times with different operator weights (P-values). Two different sets of vector queries were created from the topics, one containing information from fewer sections of a topic description. The Title, Description and Concepts sections of the topic descriptions were used in the creation of all three sets of queries, the Definitions section was used also in both sets of vector queries, while the P-norm query set and one of the vector query sets also contained information from the Narrative section of the topic descriptions. The vector query set that included the Narrative section of the topic is referred to as the long vector query set, for obvious reasons, while the other is referred to as the short vector query set.

The P-norm queries were written as complex boolean expressions using AND and OR operators. Phrases were simulated using AND operators since the queries were intended only for soft-boolean evaluation. The query terms were not specifically weighted; uniform operator weights (P-values) of 1.0, 1.5 and 2.0 were used on different evaluations of the query set.

Table 4: Summary of the five individual runs.

Title	Query Type	Similarity Measure
SV	Short vector	Cosine similarity
LV	Long vector	Cosine similarity
Pn1.0	P-norm	P-norm, P = 1.0
Pn1.5	P-norm	P-norm, P = 1.5
Pn2.0	P-norm	P-norm, P = 2.5

3.2 Individual Retrieval Runs

The first step in our TREC-2 experiments involved determination of what weighting schemes would be most effective for P-norm queries. Our TREC-1 experiments with P-norm queries had obtained mixed results, performing poorly based on binary document term weights in our Phase I experiments and performing well for a P-value of 1.0 and very poor with larger P-values in our Phase II experiments using a *tf-idf* weighting scheme [4]. We performed several P-norm retrieval runs on the two AP and two WSJ training collections with topics 51 to 100 to determine the most effective term weighting scheme for P-norm queries with large test collections. The results from these experiments are shown in Table 3 using the standard TREC-2 average non-interpolated precision and the exact R-precision measures. The most effective weighting scheme turned out to be the **SMART** *ann* weighting scheme, which confirmed the result obtained originally by Fox for the much smaller classical document collections [3].

The two sets of vector queries were evaluated using the standard cosine correlation similarity method as implemented by **SMART**. The same **SMART** *ann* weighting scheme used for the P-norm queries was used on the vector queries for several reasons. First, a weighting scheme that did not use any collection statistics was needed for the routing experiments. Second, the methods used in combining runs described in the next section required a similar range of possible similarity values produced by each run. Finally, the necessity of merging results from each collection into a single set of results was simplified since the resulting similarity values were not based on collection statistics which would have differed for each collection. The P-norm queries were evaluated using three different P-values, again using the **SMART** *ann* weighting scheme based on specific P-norm experiments described below. The five individual runs are summarized in Table 4.

The five individual runs were performed and evaluated for each of the nine training collections on topics 51 to 100. The results for these experiments are given in

Table 3: Average Precision and Exact R-Precision for P-norm experiments on weighting with the AP and WSJ collections (Ad-hoc Topics 51-100).

Coll.	P-value	Average Precision			R-Precision		
		ann	bnn	mnn	ann	bnn	mnn
AP-1	1.0	**0.2810**	0.2419	0.1419	**0.2688**	0.2660	0.1689
	1.5	**0.3122**	0.2581	0.1444	**0.2976**	0.2732	0.1757
	2.0	**0.3027**	0.2510	0.1457	**0.2968**	0.2775	0.1707
AP-2	1.0	**0.3004**	0.2672	0.1826	**0.3165**	0.2864	0.2046
	1.5	**0.3332**	0.2999	0.1831	**0.3412**	0.3118	0.2161
	2.0	**0.3300**	0.2922	0.1847	**0.3339**	0.3057	0.2284
WSJ-1	1.0	**0.2941**	0.2485	0.1742	**0.3221**	0.2830	0.2181
	1.5	**0.3199**	0.2753	0.1774	**0.3443**	0.2994	0.2225
	2.0	**0.3217**	0.2752	0.1776	**0.3470**	0.3013	0.2277
WSJ-2	1.0	**0.2206**	0.1881	0.1356	**0.2367**	0.2094	0.1722
	1.5	**0.2327**	0.2013	0.1174	**0.2511**	0.2234	0.1549
	2.0	**0.2325**	0.1970	0.1098	**0.2442**	0.2158	0.1445

Table 5. In general, the P-norm queries performed better than the vector queries. The most effective P-value however differed between the collections: The AP runs performed better with a P-value of 1.5, while a P-value of 2.0 performed better for the WSJ collections.

3.3 Combination Retrieval Runs

Our experiments in TREC-1 involved combining the results from several different retrieval runs for a given collection either simply taking the top N documents retrieved for each run, or modifying the value of N for each run, based on the eleven point average precision for that run. We felt these efforts suffered from considering only the rank of a retrieved document and not the actual similarity value itself. In TREC-2, our experiments concentrated on methods of combining runs based on the similarity values of a document to each query for each of the runs. Additionally, combining the similarities at retrieval time had the advantage of extra evidence over combining separate results files since the similarity of every document for each run was available instead of just the similarities for the top 1000 documents for each run. While our results for four of the training collections indicated that the P-norm queries performed better than the vector queries, this result was likely specific to the actual queries involved and not necessarily true in general. This lead to a decision to weight each of the separate runs equally and not favor any individual run or method. In general, it may be desirable or necessary to weight a single run more, or less, depending on its overall performance; this could be especially useful in a routing situation.

For any given information retrieval ranking metohd, there are two primary types of errors that can occur:

Table 6: Formulas for combining similarity values.

Name	*Combined Similarity* =
CombMAX	$MAX(Individual\ Similarities)$
CombMIN	$MIN(Individual\ Similarities)$
CombSUM	$SUM(Individual\ Similarities)$
CombANZ	$\frac{SUM(Individual\ Similarities)}{Number\ of\ Nonzero\ Similarities}$
CombMNZ	$SUM(Individual\ Similarities) * Number\ of\ Nonzero\ Similarities$
CombMED	$MED(Individual\ Similarities)$

assigning a relatively high rank to a non-relevant document, and assigning a relatively low rank to a relevant document. It has been shown that different retrieval paradigms will perform differently on the same set of data, often will little overlap in the set of retrieved documents. [5] For instance, when one retrieval method assigns a high rank to a non-relevant document, a different retrieval method is likely to assign that document a much lower rank. Similarly, when one retrieval method fails to assign a high rank to a relevant document, a different retrieval method is likely to assign that document a high rank. This characteristic of information retrieval methods indicates that some method for considering both retrieval methods together should help to decease the probability of this happening; of course, it is also possible for both methods to highly rank a non-relevant document or to poorly rank a relevant document.

Six methods of combining the similarity values were tested in our TREC-2 experiments, as summarized in

Table 5: Average Precision and Exact R-Precision for the five individual runs (Ad-hoc Topics 51-100).

Average non-interpolated Precision

Run	Disk 1					Disk 2				Both
	AP	DOE	FR	WSJ	ZF	AP	FR	WSJ	ZF	Disks
SV	0.2387	0.0605	0.0222	0.2203	0.1026	0.2543	0.0330	0.1503	0.0770	0.1418
LV	0.2435	0.0586	0.0302	0.2414	0.0864	0.2664	0.0324	0.1633	0.0753	0.1555
Pn1.0	0.2605	0.0658	0.0611	0.2941	0.1110	0.3004	**0.0879**	0.2206	0.1003	0.1988
Pn1.5	**0.2939**	0.0771	0.0639	0.3199	**0.1278**	**0.3332**	0.0878	**0.2327**	0.1065	0.2242
Pn2.0	0.2849	**0.0847**	**0.0706**	**0.3217**	**0.1278**	0.3300	0.0865	0.2325	**0.1136**	**0.2250**
CombSUM	0.3493	0.1001	0.0741	0.3605	0.1475	0.3748	0.0842	0.2752	0.1273	0.2620
Chg/Max	18.84%	18.18%	4.95%	12.06%	15.41%	12.48%	-4.20%	18.26%	12.05%	16.44%

Exact R-Precision

Run	Disk 1					Disk 2				Both
	AP	DOE	FR	WSJ	ZF	AP	FR	WSJ	ZF	Disks
SV	0.2624	0.0564	0.0183	0.2616	0.1180	0.2649	0.0202	0.1744	0.0922	0.2169
LV	0.2672	0.0493	0.0274	0.2800	0.0802	0.2704	0.0176	0.1860	0.0843	0.2311
Pn1.0	0.2688	0.0661	0.0533	0.3221	0.1123	0.3165	0.0971	0.2367	0.0969	0.2708
Pn1.5	**0.2976**	0.0762	0.0572	0.3443	0.1218	**0.3412**	**0.1016**	**0.2511**	0.1068	0.2962
Pn2.0	0.2968	**0.0765**	**0.0654**	**0.3470**	**0.1254**	0.3339	0.0820	0.2442	**0.1158**	**0.3008**
CombSUM	0.3590	0.0950	0.0619	0.3767	0.1357	0.3732	0.0887	0.2851	0.1216	0.3292
Chg/Max	20.63%	24.18%	-5.35%	8.55%	8.21%	9.37%	-12.69%	13.54%	5.00%	9.44%

Table 6. The rational behind the CombMIN combination method was to minimize the probability that a non-relevant document would be highly ranked, while the purpose of the CombMAX combination method was to minimize the number of relevant documents being poorly ranked. There is an inherent flaw with both of these methods; namely, they are specialized to handle specific problems without regard to their effect on the other retrieved documents: for example, the CombMIN combination method will promote the type of error that the CombMAX method is designed to minimize, and vice versa. The CombMED combination method is a simplistic approach to handling this, using the median similarity value to avoid both scenarios. What is clearly needed is some method of considering the documents' relative ranks, or similarity values, instead of simply attempting to select a single similarity value from a set of runs. To this end, we tried three other methods of combining retrieval methods. CombSUM, the summation of the set of similarity values, or, equivalently, the numerical mean of the set of the set of similarity values; CombANZ, the average of the non-zero similarity values, that ignores the effects of a single given run or query failing to retrieve a relevant document; and CombMNZ to provide higher weights to documents retrieved by multiple retrieval methods. Clearly, there are more possibilities to consider; the advantages of those chosen are simplicity, in terms of both execution efficiency and implementation, and generality, in terms of not being specific to a given method or retrieval run.

These six methods were evaluated against the AP and WSJ test collections for topics 51 through 100, combining the similarity values of each of the five individual runs specified above. The results are shown in Table 7 below the results of each of the corresponding individual runs from Table 5. Note that while the CombMAX runs performed well compared with most of the individual runs, they did not do as well as the best of the individual runs in most cases. The CombMIN runs performed similarly for the AP collection, but performed worse than every individual run for the WSJ collection.

The CombANZ runs and the CombMNZ runs both performed better than the best of the individual runs, with the CombMNZ runs performing only slightly better than the combANZ runs for three of the four collections, and performing basically the same for the fourth. The primary reason for the similar performance of the two runs is that the two methods produce the same ranked sequence of for all the documents retrieved by all five individual runs. Thus, the

The CombSUM retrieval run was performed for each of the nine collections on the two training CD-ROMs. The results are shown in Table 5. Breaking this analysis down to a per topic basis in Table 11, it can be

seen that the CombSUM method performs significantly better than the best single individual run, Pn2.0; a two-tailed paired t test on the CombSUM and Pn2.0 average precisions results in a p value of 3.1e-05, which indicates these results are conclusive. However, comparing the CombSUM results with the best individual runs for each query basis, results in a p value of approximately about 0.16, indicating that there is a 16 percent chance that the CombSUM method is no better than the best individual run, Pn2.0, for any specific query. Performing the same calculation on the R-Precision results in similar significance findings.

While combining all five runs produced an overall improvement in retrieval effectiveness over each of the runs, the same does not always hold true when combining only two or three runs. Each of the ten combinations of two CombSUM runs was performed for both of the AP test collections, as well as a run combining all three of the P-norm runs. The results of these are given in Table 8. Most of the combinations of two runs performed worse than the better of the two runs while performing better than the poorer of the two runs. One notable exception to this is the combination of the two vector runs, which performed noticeably poorer than either of the two runs.

3.4 Collection Merging

The retrieval results for each of the collections were combined by simply merging the results based solely on the combined similarity values. Since the retrieval runs were based on term weights without collection statistics such as inverse document frequency, the similarity values were directly comparable across collections. The results of merging the CombSUM results by summed similarity value for both disks, is shown in the last column of Table 5.

4 TREC-2 Results

The procedure described above was used for both our official TREC-2 routing and ad-hoc results. The exact queries for ad-hoc topics 51 to 100 used for testing our above method were used for the routing queries against the new collections on disk 3. The results obtained from performing the CombSUM retrieval runs for each of the four collections as well as the merged results are shown in Table 9. The two CombSUM entries in the last column of table are the official TREC-2 results. Since we concentrated on the ad-hoc evaluations, these routing results are included primarily for the benefit of other groups, for purposes of comparison. The ad-hoc queries for topics 101 to 150 were evaluated in the same manner, and are reported in Table 10. Again, the official results are the two CombSUM entries in last column of the table.

As can be seen from Table 12, the CombSUM method performs quite poorly for certain topics while performing very well for others, compared to the best single run's results that that topic. Comparing the CombSUM results to the single best individual run (Pn2.0) shows an improvement for 46 out of the 50 topics, which shows that the CombSUM run performs much better than any single individual run. Performing a two-tailed paired t test on the Pn2.0 and CombSUM precisions results in a p value of about 1.1e-11, which indicates these results are very conclusive. However, comparing the CombSUM results with the best individual runs on a per query basis results in a p value of about 0.2, indicating that there is a 20 percent chance that the CombSUM method is no better than the best individual run for each specific query. Again, performing the same calculation on the R-Precision results in similar values.

4.1 The CEO Model

The **Combination of Expert Opinion (CEO)** model [6, 7] of Thompson can be used to treat the different retrieval methods as experts, and allows combining their weighting probability distributions to improve performance. This could be used in a variety of ways to combine results from a variety of runs and indexing schemes (that could include stemming and/or morphological analysis). For TREC-2, the CEO experiments completed consisted of combining seven individual runs, the three P-norm extended boolean retrieval run types described above, and retrieval runs based on the long vector queries, using both cosine correlation and inner product similarity measures for **SMART** system term weighting schemes of *nnn* and *atn*. Further discussion of this process and the results are described elsewhere in these proceedings.

4.2 Evaluation

Improvements in retrieval effectiveness from combining the evidence from multiple sources of evidence has been performed before in various incarnations, most recently by Belkin *et al.* [1] who evaluated the progressive effect of considering multiple soft boolean representations to improve on a base INQUERY natural language retrieval run. In their experiments, the base INQUERY natural language run performed better than any of the boolean representations, and they report that combining the results from the natural language representation and the combined boolean representations with equal weights performed worse than the best single run. Not until weighting the natural language run four times more

Table 7: Comparison of combination runs and the five individual runs (Ad-hoc Topics 51-100).

Run	Average Precision				R-Precision			
	AP-1	WSJ-1	AP-2	WSJ-2	AP-1	WSJ-1	AP-2	WSJ-2
SV	0.2387	0.2203	0.2543	0.1503	0.2624	0.2616	0.2649	0.1744
LV	0.2435	0.2414	0.2664	0.1633	0.2672	0.2800	0.2704	0.1860
Pn1.0	0.2810	0.2941	0.3004	0.2206	0.2688	0.3221	0.3165	0.2367
Pn1.5	**0.3122**	0.3199	**0.3332**	**0.2327**	**0.2976**	0.3443	**0.3412**	**0.2511**
Pn2.0	0.3027	**0.3217**	0.3300	0.2325	0.2968	**0.3470**	0.3339	0.2442
CombMAX	0.2856	0.3205	**0.3337**	**0.2343**	**0.3013**	**0.3484**	**0.3431**	0.2449
CombMIN	0.2863	0.1924	0.3047	0.1308	**0.3036**	0.2214	0.2980	0.1395
CombSUM	**0.3493**	**0.3605**	**0.3748**	**0.2752**	**0.3590**	**0.3767**	**0.3732**	**0.2851**
CombANZ	**0.3493**	**0.3367**	**0.3748**	**0.2465**	**0.3590**	**0.3517**	**0.3732**	**0.2590**
CombMNZ	0.3059	0.3368	**0.3516**	**0.2467**	**0.3175**	**0.3517**	**0.3578**	**0.2590**
CombMED	0.2943	0.3204	**0.3335**	0.2328	**0.2977**	0.3444	**0.3414**	**0.2518**

than the combined boolean schemes did they experience improved retrieval performance when combining different query methods. This differs from our results in several ways. Most importantly, the stage at which we combine the different methods differed: Belkin *et al.* combined the query representations before performing the actual retrieval, while we combined the similarity values produced from retrieval on each method individually. The difference between the two methodologies can best be demonstrated using the standard vector space model: Belkin *et al.* combined by summing the vector representations of each query, while our method is analogous to summing the cosines of the angles between each vector and a document. It is easily shown that the cosine of the angle between a document vector and a combined query vector, that is the sum of two query vectors as in the Belkin *et al.* approach, is not equal to the sum of the cosines between a document vector and the two separate query vectors. Other differences between the two methodologies include the fact that our P-norm queries performed better on average than our natural language vector queries, with exceptions on a per query basis. We used only one P-norm query and modified the operator weights while Belkin *et al.* used five different boolean queries. Finally, combining with five runs with equal weights actually improved performance over each individual run. However, one common trend emerges from both experiments: the more query representations considered, the better the results.

4.3 Future Exploration

Planned future work includes studying the following:

- Individually weighting various methods' similarity values when performing combination runs.
- Normalization methods to allow combination of runs made with different weighting schemes.
- Extending the analysis to all combinations of three and four retrieval runs.
- Considering more/different query types.

5 Acknowledgements

This research was supported in part by DARPA and by PRC Inc. We also thank Russell Modlin, M. Prabhakar Koushik and Durgesh Rao for their collaboration during TREC-1.

References

[1] Belkin, N.J., Cool, C., Croft, W.B., Callan, J.P. (1993, June). The Effect of Multiple Query Representations on Information Retrieval Performance. *Proc. 16th Int'l Conf. on R&D in IR (SIGIR '93)*, Pittsburgh, 339–346.

[2] Buckley, C. (1985, May) Implementation of the SMART information retrieval system. Technical Report 85-686, Cornell University, Department of Computer Science.

[3] Fox, E.A. (1983, August). Extending the Boolean and Vector Space Models of Information Retrieval with P-Norm Queries and Multiple Concept Types. Cornell University Department of Computer Science dissertation.

[4] Fox, E.A., Koushik, M.P., Shaw, J., Modlin, R., Rao, D. (1993). Combining Evidence from Multiple Searches. In *The First Text REtrieval Conference*

Table 8: Average Precision and Exact R-Precision for CombSUM runs combining two or three individual runs (Ad-hoc Topics 51-100).

Run	Average Precision				R-Precision			
	AP-1	WSJ-1	AP-2	WSJ-2	AP-1	WSJ-1	AP-2	WSJ-2
SV	0.2387	0.2203	0.2543	0.1503	0.2624	0.2616	0.2649	0.1744
LV	0.2435	0.2414	0.2664	0.1633	0.2672	0.2800	0.2704	0.1860
Pn1.0	0.2810	0.2941	0.3004	0.2206	0.2688	0.3221	0.3165	0.2367
Pn1.5	**0.3122**	0.3199	**0.3332**	**0.2327**	**0.2976**	0.3443	**0.3412**	**0.2511**
Pn2.0	0.3027	**0.3217**	0.3300	0.2325	0.2968	**0.3470**	0.3339	0.2442
SV and LV	0.1457	0.1657	0.1611	0.1100	0.1492	0.1887	0.1524	0.1124
SV and Pn1.0	0.2774	0.3111	0.3257	**0.2362**	0.2874	0.3332	0.3360	**0.2554**
SV and Pn1.5	0.3117	**0.3389**	**0.3614**	**0.2463**	**0.3198**	**0.3575**	**0.3675**	**0.2637**
SV and Pn2.0	0.3012	**0.3395**	**0.3584**	**0.2467**	**0.3153**	**0.3580**	**0.3636**	**0.2551**
LV and Pn1.0	0.2744	0.3136	0.3197	**0.2353**	0.2867	0.3357	0.3269	**0.2518**
LV and Pn1.5	0.3057	**0.3413**	**0.3536**	**0.2442**	**0.3181**	**0.3596**	**0.3624**	**0.2568**
LV and Pn2.0	0.2950	**0.3408**	**0.3518**	**0.2458**	**0.3141**	**0.3608**	**0.3596**	**0.2615**
Pn1.0 and Pn1.5	0.2817	0.3109	0.3243	0.2324	0.2898	0.3412	0.3395	0.2476
Pn1.0 and Pn2.0	0.2935	0.3191	0.3330	**0.2367**	0.2944	0.3426	0.3397	0.2507
Pn1.5 and Pn2.0	0.2928	**0.3233**	**0.3336**	**0.2328**	0.2935	**0.3478**	0.3351	0.2489
Pn1.0, Pn1.5 and Pn2.0	0.2943	0.3192	**0.3345**	**0.2339**	0.2953	0.3421	0.3386	0.2497

(TREC-1), D.K. Harmon (Ed.), National Institute of Standards and Technology Special Publication 500-207, Gaithersburg, MD.

[5] Katzer, J., McGill, M.J., Tessier, J.A., Frakes, W., Dasgupta, P. (1982). A Study of the Overlap among Document Representations. *Information Technology: Reseach and Development*, 1(2):261-274.

[6] Thompson, P. (1990) A Combination of Expert Opinion approach to probabilistic information retrieval, Part 1: The conceptual model. *Information Processing & Management*, 26(3):371–382, 1990.

[7] Thompson, P. (1990). A Combination of Expert Opinion approach to probabilistic information retrieval, Part 2: Mathematical treatment of CEO Model 3. *Information Processing & Management*, 26(3):383–394.

Table 9: Average Precision and Exact R-Precision for the five individual runs compared with the combined CombSUM runs (Routing Topics 51-100).

Average non-interpolated Precision

Run	AP-3	PATN-3	SJM-3	ZF-3	Disk 3
SV	0.1347	0.0189	0.1139	0.0593	0.0589
LV	0.1189	0.0156	0.1056	0.0587	0.0494
Pn1.0	0.2519	**0.0257**	0.2128	0.1141	0.2039
Pn1.5	**0.2869**	0.0239	**0.2411**	0.1189	**0.2279**
Pn2.0	0.2852	0.0221	0.2390	**0.1303**	0.2225
CombSUM	0.3196	0.0260	0.2696	0.1304	0.2681
Chg/Max	11.4%	1.2%	11.8%	0.07%	17.6%

Exact R-Precision

Run	AP-3	PATN-3	SJM-3	ZF-3	Disk 3
SV	0.1703	0.0171	0.1337	0.0595	0.0595
LV	0.1444	0.0156	0.1098	0.0547	0.1002
Pn1.0	0.2790	**0.0325**	0.2185	0.1224	0.2594
Pn1.5	**0.3082**	0.0322	**0.2579**	0.1248	0.2786
Pn2.0	0.3062	0.0310	0.2531	**0.1462**	**0.2809**
CombSUM	0.3319	0.0319	0.2900	0.1260	0.3143
Chg/Max	7.7%	-1.8%	12.4%	-13.8%	11.9%

Table 10: Average Precision and Exact R-Precision for the five individual runs (Ad-hoc Topics 101-150).

Average non-interpolated Precision

	Disk 1					Disk 2				Both
Run	AP	DOE	FR	WSJ	ZF	AP	FR	WSJ	ZF	Disks
SV	0.3237	0.0949	0.0630	0.2740	0.0936	0.3068	0.0650	0.2259	0.1166	0.2035
LV	0.3326	0.0697	0.1018	0.2848	0.0997	0.2981	0.0602	0.2483	0.1045	0.2159
Pn1.0	0.3340	**0.0831**	0.1777	0.3153	0.1292	0.3133	0.1927	0.2838	0.1722	0.2205
Pn1.5	**0.3682**	0.0814	**0.1874**	**0.3332**	**0.1430**	**0.3438**	0.1982	**0.2941**	0.1964	0.2543
Pn2.0	0.3647	0.0750	0.1761	0.3290	0.1307	0.3419	**0.1995**	0.2828	**0.2018**	**0.2573**
CombSUM	0.4153	0.1038	0.2133	0.3778	0.1657	0.3959	0.2000	0.3561	0.2200	0.3206
Chg/Max	12.8%	9.4%	13.8%	13.4%	15.9%	15.1%	0.2%	21.0%	9%	24.6%

Exact R-Precision

	Disk 1					Disk 2				Both
Run	AP	DOE	FR	WSJ	ZF	AP	FR	WSJ	ZF	Disks
SV	0.3351	0.0947	0.0567	0.2881	0.1086	0.3052	0.0718	0.2502	0.1130	0.2649
LV	0.3385	0.0714	0.1006	0.3022	0.1055	0.3019	0.0540	0.2712	0.1036	0.2661
Pn1.0	0.3495	**0.0989**	0.1434	0.3322	0.1190	0.3139	0.1653	0.2934	0.1577	0.2867
Pn1.5	0.3721	0.0899	**0.1582**	**0.3440**	**0.1277**	**0.3465**	**0.1716**	**0.3044**	0.1796	0.3203
Pn2.0	**0.3735**	0.0925	0.1579	0.3389	0.1154	0.3401	0.1640	0.2953	**0.1979**	**0.3233**
CombSUM	0.4137	0.1156	0.1938	0.3757	0.1389	0.3712	0.1741	0.3385	0.1909	0.3711
Chg/Max	10.8%	16.9%	22.5%	9.2%	8.8%	7.1%	1.4%	11.2%	-3.5%	14.8%

Table 11: Per-Topic comparison of Average Precisions (Ad-hoc Topics 51-100).

Topic	#Rel	SV	LV	Pn1.0	Pn1.5	Pn2.0	CombSUM	Chg/Max	Chg/Pn2.0
74	273	0.0000	0.0099	0.0001	0.0002	0.0002	0.0002	-97.98%	0.00%
90	206	0.1248	0.1134	0.0082	0.0036	0.0015	0.0089	-92.87%	493.33%
77	138	0.1100	0.2241	0.0093	0.0196	0.0246	0.0411	-81.66%	67.07%
91	40	0.1648	0.1039	0.0012	0.0038	0.0068	0.0308	-81.31%	352.94%
72	119	0.1099	0.1173	0.0076	0.0116	0.0138	0.0323	-72.46%	134.06%
73	183	0.0231	0.0124	0.0017	0.0052	0.0120	0.0098	-57.58%	-18.33%
94	310	0.0176	0.0185	0.0191	0.1094	0.2100	0.1043	-50.33%	-50.33%
85	894	0.1587	0.1779	0.0239	0.0433	0.0518	0.1018	-42.78%	96.53%
67	92	0.0011	0.0012	0.0435	0.0487	0.0502	0.0290	-42.23%	-42.23%
55	810	0.2362	0.4535	0.1153	0.1266	0.1256	0.2874	-36.63%	128.82%
64	375	0.1493	0.1641	0.0740	0.0673	0.0706	0.1097	-33.15%	55.38%
57	461	0.0282	0.0605	0.5464	0.4518	0.3447	0.3786	-30.71%	9.83%
80	374	0.0218	0.0266	0.1007	0.1125	0.1025	0.0899	-20.09%	-12.29%
82	602	0.2544	0.3139	0.2603	0.1746	0.1278	0.2521	-19.69%	97.26%
97	352	0.0513	0.0513	0.1133	0.1376	0.1405	0.1190	-15.30%	-15.30%
76	294	0.1481	0.1677	0.1221	0.0966	0.0755	0.1440	-14.13%	90.73%
89	174	0.0365	0.0762	0.0481	0.1159	0.1515	0.1306	-13.80%	-13.80%
98	666	0.0822	0.0677	0.0421	0.0579	0.0703	0.0711	-13.50%	1.14%
84	396	0.1659	0.0985	0.0787	0.1251	0.1559	0.1471	-11.33%	-5.64%
99	288	0.4834	0.4796	0.3529	0.3551	0.3224	0.4480	-7.32%	38.96%
58	159	0.0971	0.0620	0.3459	0.4198	0.4072	0.3937	-6.22%	-3.32%
92	88	0.0253	0.0405	0.0130	0.0154	0.0176	0.0381	-5.93%	116.48%
71	380	0.0595	0.0613	0.1396	0.1978	0.2210	0.2156	-2.44%	-2.44%
88	165	0.1813	0.2341	0.3629	0.3635	0.3542	0.3550	-2.34%	0.23%
59	579	0.2368	0.2272	0.1546	0.3827	0.4370	0.4282	-2.01%	-2.01%
93	171	0.0344	0.0385	0.5233	0.4831	0.3779	0.5141	-1.76%	36.04%
95	263	0.0149	0.0228	0.1450	0.2072	0.2392	0.2375	-0.71%	-0.71%
52	535	0.4918	0.4462	0.4896	0.6623	0.6882	0.6864	-0.26%	-0.26%
78	162	0.2202	0.2632	0.7696	0.7484	0.7250	0.7705	0.12%	6.28%
61	206	0.4106	0.5130	0.3646	0.3786	0.3621	0.5167	0.72%	42.70%
51	138	0.2799	0.3957	0.4751	0.5282	0.5428	0.5476	0.88%	0.88%
70	55	0.1156	0.1198	0.7700	0.7919	0.7982	0.8080	1.23%	1.23%
54	171	0.3063	0.3045	0.3950	0.3207	0.2673	0.4048	2.48%	51.44%
62	298	0.1674	0.1907	0.1519	0.2087	0.2121	0.2198	3.63%	3.63%
81	62	0.1759	0.1410	0.2287	0.2370	0.2321	0.2467	4.09%	6.29%
68	195	0.0960	0.0920	0.1040	0.2098	0.2540	0.2651	4.37%	4.37%
69	52	0.0956	0.1629	0.5382	0.5873	0.5833	0.6227	6.03%	6.75%
53	571	0.1821	0.1874	0.1543	0.2928	0.3241	0.3461	6.79%	6.79%
83	633	0.2673	0.2931	0.1753	0.2412	0.2666	0.3317	13.17%	24.42%
56	878	0.3011	0.3277	0.3391	0.3089	0.2691	0.3955	16.63%	46.97%
100	317	0.1904	0.1972	0.1423	0.1815	0.2094	0.2516	20.15%	20.15%
65	386	0.0100	0.0078	0.1111	0.1190	0.1236	0.1529	23.71%	23.71%
86	213	0.5146	0.4242	0.5216	0.5234	0.4891	0.6624	26.56%	35.43%
60	60	0.0547	0.0887	0.0866	0.0960	0.0992	0.1259	26.92%	26.92%
79	232	0.1376	0.1220	0.2057	0.2784	0.2719	0.3690	32.54%	35.71%
75	365	0.0016	0.0037	0.0443	0.0499	0.0481	0.0684	37.07%	42.20%
63	208	0.0032	0.0028	0.0631	0.0597	0.0688	0.0972	41.28%	41.28%
66	197	0.0001	0.0000	0.0320	0.0388	0.0478	0.0695	45.40%	45.40%
87	188	0.0436	0.0559	0.0410	0.0914	0.1256	0.1867	48.65%	48.65%
96	693	0.0095	0.0084	0.0833	0.1177	0.1302	0.2356	80.95%	80.95%
Avg	15667	0.1418	0.1555	0.1988	0.2242	0.225	0.262	16.44%	16.44%

Table 12: Per-Topic comparison of Average Precisions (Ad-hoc Topics 101-150).

Topic	#Rel	SV	LV	Pn1.0	Pn1.5	Pn2.0	CombSUM	Chg/Max	Chg/Pn2.0
103	94	0.2400	0.2379	0.0130	0.0225	0.0309	0.0532	-77.83%	72.17%
122	114	0.1467	0.1059	0.0547	0.0490	0.0544	0.0898	-38.79%	65.07%
101	57	0.2232	0.1932	0.0900	0.1088	0.1175	0.1482	-33.60%	26.13%
140	25	0.0150	0.0520	0.0111	0.0207	0.0315	0.0356	-31.54%	13.02%
135	400	0.4072	0.4449	0.1103	0.1121	0.0835	0.3122	-29.83%	273.89%
104	75	0.1867	0.2214	0.0715	0.0642	0.0828	0.1671	-24.53%	101.81%
121	55	0.0017	0.0028	0.0283	0.0542	0.0628	0.0475	-24.36%	-24.36%
127	223	0.2411	0.2476	0.0543	0.0942	0.1028	0.1911	-22.82%	85.89%
143	397	0.4069	0.3978	0.1628	0.1968	0.2022	0.3257	-19.96%	61.08%
124	173	0.0201	0.0165	0.1711	0.1360	0.1068	0.1501	-12.27%	40.54%
146	358	0.6795	0.7017	0.4394	0.4775	0.4907	0.6255	-10.86%	27.47%
114	138	0.0518	0.0756	0.2381	0.2559	0.2615	0.2525	-3.44%	-3.44%
112	291	0.0041	0.0188	0.3876	0.3515	0.3194	0.3779	-2.50%	18.32%
130	286	0.2301	0.2937	0.4184	0.5477	0.5749	0.5632	-2.04%	-2.04%
102	64	0.2127	0.2308	0.1119	0.1373	0.1505	0.2287	-0.91%	51.96%
128	381	0.0915	0.0751	0.2583	0.3710	0.4190	0.4156	-0.81%	-0.81%
150	458	0.4618	0.4994	0.3362	0.4432	0.4566	0.4985	-0.18%	9.18%
115	165	0.4053	0.4383	0.3262	0.3633	0.3720	0.4407	0.55%	18.47%
113	206	0.0531	0.0768	0.2700	0.3304	0.3030	0.3367	1.91%	11.12%
119	326	0.0644	0.0843	0.2442	0.2252	0.2018	0.2497	2.25%	23.74%
107	98	0.1101	0.1580	0.3156	0.4062	0.4451	0.4592	3.17%	3.17%
116	49	0.2635	0.1886	0.3008	0.2803	0.2487	0.3125	3.89%	25.65%
118	273	0.0259	0.0164	0.1551	0.1736	0.1725	0.1806	4.03%	4.70%
145	162	0.3218	0.2983	0.2169	0.2180	0.1961	0.3359	4.38%	71.29%
138	52	0.0895	0.0992	0.1169	0.1321	0.1322	0.1386	4.84%	4.84%
148	250	0.7256	0.7300	0.7316	0.7904	0.8010	0.8416	5.07%	5.07%
108	294	0.0994	0.1266	0.3089	0.2586	0.1820	0.3264	5.67%	79.34%
136	206	0.1713	0.1976	0.3943	0.5899	0.5894	0.6319	7.12%	7.21%
134	188	0.4033	0.4132	0.5091	0.4897	0.4675	0.5500	8.03%	17.65%
132	201	0.1979	0.3246	0.5671	0.5759	0.5625	0.6222	8.04%	10.61%
133	80	0.2396	0.2793	0.1541	0.2322	0.2425	0.3034	8.63%	25.11%
147	315	0.1371	0.1266	0.2312	0.2820	0.3001	0.3317	10.53%	10.53%
106	201	0.0354	0.0458	0.2784	0.3463	0.3653	0.4039	10.57%	10.57%
126	240	0.2766	0.2990	0.1997	0.3132	0.3548	0.3971	11.92%	11.92%
125	169	0.1632	0.1719	0.2580	0.2224	0.1842	0.2920	13.18%	58.52%
137	158	0.3365	0.4135	0.2339	0.3128	0.3555	0.4715	14.03%	32.63%
131	28	0.0659	0.0648	0.0671	0.0984	0.1061	0.1255	18.28%	18.28%
110	496	0.4909	0.4968	0.4845	0.4768	0.4336	0.6001	20.79%	38.40%
142	660	0.4269	0.4479	0.4060	0.4694	0.4629	0.5721	21.88%	23.59%
117	275	0.1165	0.1030	0.1296	0.1596	0.1637	0.2007	22.60%	22.60%
149	133	0.0309	0.0836	0.0490	0.0732	0.0769	0.1028	22.97%	33.68%
129	207	0.3391	0.2338	0.2402	0.3348	0.3477	0.4369	25.65%	25.65%
144	49	0.2490	0.2275	0.0832	0.1327	0.1958	0.3130	25.70%	59.86%
105	54	0.0656	0.1744	0.1304	0.1439	0.1441	0.2262	29.70%	56.97%
139	55	0.0911	0.1139	0.0592	0.0705	0.0776	0.1481	30.03%	90.85%
111	285	0.3795	0.3540	0.1991	0.3025	0.3756	0.5025	32.41%	33.79%
123	435	0.0697	0.0800	0.2254	0.2252	0.2043	0.3014	33.72%	47.53%
120	83	0.0165	0.0136	0.0490	0.0556	0.0538	0.0746	34.17%	38.66%
109	742	0.0875	0.0879	0.0811	0.1344	0.1500	0.2290	52.67%	52.67%
141	36	0.0084	0.0125	0.0538	0.0517	0.0514	0.0873	62.27%	69.84%
Avg	10760	0.2035	0.2159	0.2205	0.2543	0.2573	0.3206	24.60%	24.60%

Machine Learning for Knowledge-Based Document Routing (A Report on the TREC–2 Experiment)

Richard M. Tong, Lee A. Appelbaum

Advanced Decision Systems
(a division of Booz•Allen & Hamilton, Inc.)
1500 Plymouth Street, Mountain View, CA 94043

1 Introduction

This paper contains a description of the experiments performed by Advanced Decision Systems as part of the Second Text Retrieval Conference (TREC-2).[1]

The overall system we have developed for TREC-2 demonstrates how we can combine statistically-oriented machine learning techniques with a commercially available knowledge-based information retrieval system. As in TREC-1, the tool we using for the fully automatic construction of routing queries is based on the Classification and Regression Trees (CART) algorithm.[2] However, in a departure from TREC-1, we have expanded our definition of what constitutes a document "feature" within the CART algorithm, and also explored how the CART output can be used as the basis of topic definitions that can be interpreted by the TOPIC® retrieval system developed by Verity, Inc. of Mountain View, CA.

Section 2 of the paper contains a review of the CART algorithm itself and the data structures it produces. Section 3 of the paper describes the two basic algorithms we have devised for converting CART output into TOPIC readable files. Section 4 of the paper contains a description of our experimental procedure and an analysis of the official results, as well as data from a series of auxiliary experiments. Section 5 contains some general comments on overall performance. We conclude in Section 6 with a brief discussion of possible future research directions.

1. Requests for further information about the TREC-2 experiments should be directed to the authors at the address above, or electronically to either `rtong@ads.com` or `lee@ads.com`.

2. A comprehensive discussion of the CART algorithm can be found in [1]. Details of previous work on the use of CART for information retrieval are presented in [2] and in [3].

2 The CART Algorithm

CART has been shown to be useful when one has access to datasets describing known classes of observations, and wishes to obtain rules for classifying future observations of unknown class—exactly as in the document routing problem. CART is particularly attractive when the dataset is "messy" (i.e., is noisy and has many missing values) and thus unsuitable for use with more traditional classification techniques. In addition, and particularly important for the document routing application, if it is important to be able to specify both the misclassification costs and the prior probabilities of class membership then CART has a direct way of incorporating such information into the tree building process. Finally, CART can generate auxiliary information, such as the expected misclassification rate for the classifier as a whole and for each terminal node in the tree, that is useful for the document routing problem.

2.1 CART Processing Flow

Figure 1 shows how the CART algorithm is used to construct the optimal classification tree based on the training data provided to it.The diagram shows the four sub-processes used to generate the optimal tree, T*. The "raw" training data (i.e., the original texts of the articles), together with the class specifications (i.e., the training data relevance judgments) and the feature specifications (i.e., the words defined to be features), are input to the Feature Extraction Module. The output is a set of vectors that record the class membership and the features contained in the training data. These vectors, together with class priors and the cost function (these are optional), are input to the Tree Growing Module which then constructs the maximal tree (T_{max}) that characterizes the training data. Since this tree overfits the data, the next step is to construct a series of nested sub-trees by pruning T_{max} to the root. The sequence of sub-trees ($T_{max} > T_1 > \ldots > T_n$) are input to the Tree Selection Module which per-

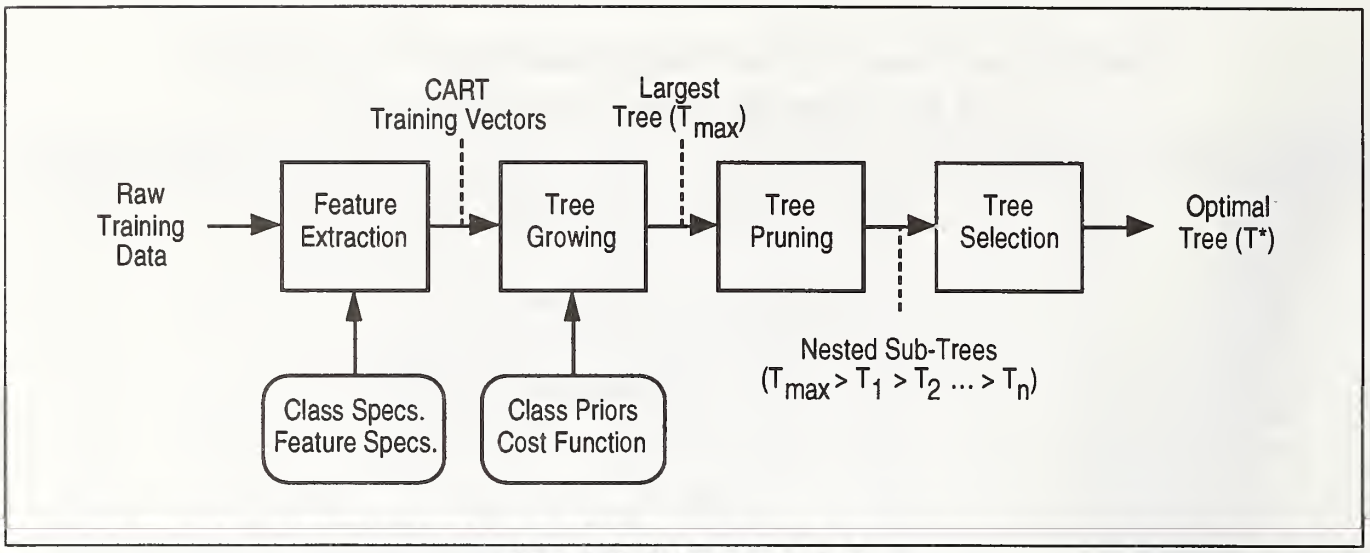


Figure 1: CART Processing Flow

forms a cross-validation analysis on the sequence and selects that tree with the lowest cross-validation error.[3] This is T*.

In our TREC-2 system, we take T* and convert it into a TOPIC outline file as described in Section 3. That is, rather than use CART itself to perform the routing on the unseen documents (as we did in TREC-1), we use the CART trees as skeletons for TOPIC concepts, and then have TOPIC do the routing. An advantage of this is that we can make use of TOPIC's extensively optimized text database capabilities, thus allowing us to easily generate the output files needed for the official scoring program.

2.2 Data Structures Generated by CART

To illustrate the processing that CART performs, we will use an example taken from the TREC-2 corpus. A example query is as follows:

```
<top>
<head> Tipster Topic Description
<num> Number: 097
<dom> Domain: Science and Technology
<title> Topic: Fiber Optics Applications

<desc> Description:
Document must identify instances of fiber
optics technology actually in use.

<narr> Narrative:
To be relevant, a document must describe
actual operational situations in which
fiber optics are being employed, or will
be employed. A document describing future
fiber optics use will be relevant only if
contracts have been signed concerning the
future application.

<con> Concept(s):
1. fiber optic, light
2. telephone, LAN, television

<fac> Factor(s):

<def> Definition(s):
1. Fiber optics refers to technology in
which information is passed via laser
light transmitted through glass or plastic
fibers.
<\top>
```

This is a very comprehensive statement of information need and provides a rich set of features that we can use for CART.[4] Our basic procedure is to extract from the information need statement all the unique content words and then stem them, which gives the following list:

```
ACT APPL CONCERN CONTRACT DESCRIB DOCU
EMPLOY FIB FUT GLASS INFORM LAN LAS LIGHT
OPER OPT PAS PLAST REF RELEV SIGN SITU
TECHNOLOG TELEPHON TELEV TRANSMIT VIA[5]
```

3. The algorithm actually minimizes with respect to both the cross-validation error and the tree complexity. So that if two trees have statistically indistinguishable error rates, then the smaller of the two trees will be selected as optimal.

4. In general, determining what set of features to use for CART is a matter of experience and judgement. For the TREC-2 corpus we have made use of the information need statements, but other approaches, such as using all the unique words in the training set, are equally valid. In fact, it is this freedom of choice of features that gives CART a great deal of its flexibility.

Given this list of features we can generate the CART training vectors by testing all the documents in the training set for the presence of absence of the feature.[6] The resulting vectors, together with the ground truth classification, are the input to CART. The output from CART contains information about the sequence of nested decision trees and a specification of the optimal tree.

For our example, the maximal tree is shown in Figure 2. This tree is to be read from left to right. Thus the first test is on the presence of the stem CONTRACT. If the stem is present (i.e., if the test CONTRACT<=0.5 fails) then the tree branches downwards (this is leftward in CART jargon); if the stem is not present (i.e., the test succeeds) then the tree branches upwards (rightward).

Associated with each decision node is information that tells:

- what level in the tree the test occurs; the k value — in all cases k ranges from k=1, which corresponds to the maximal tree, to $k=k_{max}$, which corresponds to the null tree and where k_{max} is dependent on the actual tree grown by CART;
- the class to be assigned to this node if the tree were pruned here; the ac value — in the present case ac=1 for the class that corresponds to a relevant document and ac=0 for the class that corresponds to a non-relevant document; and
- the error rate of the node; the Rt value — this indicates the resubstitution error rate for the specific node.

The terminal nodes in the tree have similar information associated with them. Notice though that terminal nodes, by definition, specify to which class the node corresponds.

```
                                    class 0 (k=2,Rt=0.230318)
                               CONCERN<=0.50 (k=2,ac=0,Rt=0.251256)
                                    class 1 (k=2,Rt=0.019586)
                                TRANSMIT<=0.50 (k=2,ac=0,Rt=0.020938)
                                    class 0 (k=2,Rt=0.000000)
                           GLASS<=0.50 (k=5,ac=0,Rt=0.261725)
                                    class 1 (k=2,Rt=0.007834)
                                TRANSMIT<=0.50 (k=2,ac=0,Rt=0.010469)
                                    class 0 (k=2,Rt=0.000000)
                               CONCERN<=0.50 (k=2,ac=0,Rt=0.010469)
                                    class 0 (k=2,Rt=0.000000)
                      VIA<=0.50 (k=5,ac=0,Rt=0.303601)
                               class 1 (k=4,Rt=0.003917)
                           TRANSMIT<=0.50 (k=5,ac=1,Rt=0.007834)
                               class 0 (k=4,Rt=0.000000)
                SIGN<=0.50 (k=5,ac=0,Rt=0.345477)
                      class 1 (k=5,Rt=0.003917)
           LIGHT<=0.50 (k=6,ac=0,Rt=0.376884)
                      class 0 (k=3,Rt=0.000000)
                GLASS<=0.50 (k=5,ac=0,Rt=0.031407)
                           class 0 (k=3,Rt=0.010469)
                      TRANSMIT<=0.50 (k=3,ac=0,Rt=0.031407)
                               class 0 (k=3,Rt=0.010469)
                           VIA<=0.50 (k=3,ac=1,Rt=0.019586)
                               class 1 (k=3,Rt=0.003917)
CONTRACT<=0.50 (k=7,ac=1,Rt=0.497487)
                      class 1 (k=4,Rt=0.019586)
                SIGN<=0.50 (k=4,ac=1,Rt=0.023503)
                      class 0 (k=4,Rt=0.000000)
           LIGHT<=0.50 (k=6,ac=1,Rt=0.027421)
                class 0 (k=4,Rt=0.000000)
```

Figure 2: Maximal Tree

5. We only use the <narr>, <con> and <def> fields, and the stop word list is taken from [4]. The stemming algorithm is taken from [5].

6. In our TREC-1 experiments we also made use of the frequency of occurrence of the features in the documents. Since our ultimate goal here is to generate TOPIC trees we restrict ourselves to just testing for the presence or absence of the feature — this allows us to perform a straightforward conversion between CART and TOPIC.

CART also generates the following table of error estimates that are used in selecting the optimal tree. Here k is the k-value, |T| is the size of the tree (measured in terms of the number of terminals), R(T) is the resubstitution rate for the overall tree, and Rcv(T) is the cross-validation rate for the overall tree. Recall that CART minimizes with respect to both size and cross-validation rate, so that for our example, the optimal tree is when k=4.

k	|T|	R(T)	Rcv(T)
1	16	0.3100	0.3037
2	11	0.3140	0.2866
3	8	0.3206	0.2353
*4	5	0.3323	0.2296
5	2	0.4043	0.3891
6	1	0.4975	0.6285

So, pruning the maximal tree so that all nodes have k>4 gives us the following optimal tree shown in Figure 3.

3	20	0.1313	0.2935
4	10	0.1761	0.2758
5	7	0.1971	0.2815
*6	6	0.2050	0.2929
7	5	0.2154	0.3101
8	4	0.2273	0.3101
9	2	0.2681	0.3328
10	1	0.4975	0.6264

so that the optimal tree is as shown in Figure 4.

3 Transforming CART Trees into TOPIC Outline Specifications

The trees produced by CART are not directly usable by TOPIC because they represent the information we need (i.e., the decision function and the associated decision variables) in a form that is incompatible with the TOPIC

```
                                    class 0 (k=5,Rt=0.261725)
                                VIA<=0.50 (k=5,ac=0,Rt=0.303601)
                                    class 1 (k=5,Rt=0.007834)
                            SIGN<=0.50 (k=5,ac=0,Rt=0.345477)
                                    class 1 (k=5,Rt=0.003917)
                        LIGHT<=0.50 (k=6,ac=0,Rt=0.376884)
                                class 0 (k=5,Rt=0.031407)
                    CONTRACT<=0.50 (k=7,ac=1,Rt=0.497487)
                        class 1 (k=6,Rt=0.027421)
```

Figure 3: Optimal Tree Using Stemmed Features

Notice that by pruning away the deeper nodes in the tree we are left with a tree that tests on just five of the original 26 stems, and has three paths that lead to `class-1` nodes (i.e., a decision that the document is relevant).

The unstemmed version of our procedure is identical except that we do no stemming of the unique words extracted from the information need statement. For the example, this produces the following list of features:

```
ACTUAL APPLICATION CONCERNING CONTRACTS
DESCRIBE DESCRIBING DOCUMENT EMPLOYED
FIBER FIBERS FUTURE GLASS INFORMATION LAN
LASER LIGHT OPERATIONAL OPTIC OPTICS
PASSED PLASTIC REFERS RELEVANT SIGNED
SITUATIONS TECHNOLOGY TELEPHONE TELEVISION
TRANSMITTED VIA
```

and the following error table:

k	|T|	R(T)	Rcv(T)
1	36	0.0932	0.3168
2	33	0.0972	0.2763

knowledge-representation. The main problem then is to define a transformation from the CART representation to the TOPIC representation that at least preserves the decision information and perhaps augments it so that we get improved routing performance.

In this section we explore two possible strategies for constructing these transformations. The first is a strict re-coding of the information in the CART tree. The second generalizes the intent of the CART tree but adds no new information. Both of these techniques are "automatic," in the sense that, once various parameters have been chosen, the algorithms work without human intervention.

3.1 First Canonical Form

The first canonical from completely preserves the decision function generated by CART and involves a simple mapping of the CART tree into TOPIC outline file format.[7] To do this, we begin by observing that each path in the CART tree from the root to a `class-1` leaf node constitutes a conjunction of tests of decision variables. Since we

```
          class 0 (k=10,Rt=0.062814)
TELEPHONE<=0.50 (k=11,ac=1,Rt=0.497487)
               class 1 (k=8,Rt=0.142139)
     TRANSMITTED<=0.50 (k=9,ac=1,Rt=0.153984)
               class 0 (k=8,Rt=0.000000)
LIGHT<=0.50 (k=10,ac=1,Rt=0.205312)
                    class 0 (k=7,Rt=0.000000)
               TRANSMITTED<=0.50 (k=9,ac=0,Rt=0.010469)
                    class 1 (k=7,Rt=0.000000)
          ACTUAL<=0.50 (k=9,ac=0,Rt=0.031407)
               class 1 (k=9,Rt=0.000000)
```

Figure 4: Optimal Tree Using Unstemmed Features

have constrained the tests to be of the form "Is word X present or not?" we can easily model this conjunction in TOPIC using AND and NOT operators. Since there will generally be multiple paths to `class-1` leaf nodes, the algorithm combines the separate paths in the TOPIC definition using the OR operator.

To illustrate, the optimal tree for our example contains the following conjunctive descriptions of class-1 leaf nodes:

```
1. CONTRACT
2. SIGN and not LIGHT and not CONTRACT
3. VIA and not SIGN and not LIGHT and not
CONTRACT
```

which leads directly to a TOPIC outline of the form:

```
Topic_097 <Or>
*  0.77 TopicStyle_097_1 <Or>
**  0.99 TopicPath_097_1-1 <And>
***  ~ 'CONTRACT'
***  ~ 'LIGHT'
***  ~ 'SIGN'
***  'VIA'
**  1.00 TopicPath_097_1-2 <And>
***  ~ 'CONTRACT'
***  ~ 'LIGHT'
***  'SIGN'
**  0.97 TopicPath_097_1-3 <And>
***  'CONTRACT'
```

Here we use a notation for topic names that ensures uniqueness. Notice that in this model we use the resubstitution rates (actually $1-R_t$) to define a weight for the individual conjuncts and the overall cross-validation rate (actually $1-R_{cv}$) as the weight for the disjuncts. Note also that since TOPIC only uses weights defined to two decimal places we have rounded the weights derived from the CART tree.

7. TOPIC uses a representation for concepts that can be recorded in so-called outline format. A collection of such specifications is used by TOPIC to built the concept trees used for retrieval.

3.2 Second Canonical Form

The second canonical form makes use of the decision variables chosen by CART but not the actual decision function. This model is based on two observations:

- first, that the set of variables used by CART are, when taken as a whole, indicative of the general topic of the information need statement, and
- second, that every variable used in the tree is on the path to at least one `class-0` node and at least one `class-1` node.

Thus, from an information retrieval perspective, all the decision variables have some contribution to make to an assessment of the relevance of a document. So rather than use the specific decision function constructed by CART we can replace this with one that can be thought of as "The more features present in a document the better." This is modelled straightforwardly in TOPIC using the ACCRUE operator.

To illustrate, the maximal tree for our example contains the following set of decision variables:

```
CONCERN CONTRACT GLASS LIGHT SIGN TRANSMIT
VIA
```

which leads directly to a TOPIC outline of the form:

```
Topic_097 <Or>
*  0.70 TopicStyle_097_2 <Accrue>
**  0.25 'CONCERN'
**  0.75 'CONTRACT'
**  0.75 'GLASS'
**  0.75 'LIGHT'
**  0.75 'SIGN'
**  0.75 'TRANSMIT'
**  0.75 'VIA'
```

The weighting scheme we have used gives higher values to variables (features) that are in the optimal tree, intermediate values to variables on the fringe of the optimal tree (there

are none of these in the current example), and lower values to features outside the optimal tree.[8] At this point the specific values chosen represent our "best guess" at a weighting scheme, further experimentation will undoubtedly reveal a better strategy. As in the first canonical form, the overall weight for the TOPIC tree is based on the cross-validation rate for the maximal tree.

4 The TREC-2 Experiments

For TREC-2 we again focused only on the document routing problem. Since our technique requires training data it does not easily lend itself to the *ad hoc* retrieval problem and so rather than "force-fit" it we chose to generate four sets of results for the routing queries (topics 51-100). Each set of results was generated totally automatically. The results sets are labelled ads1, ads2, ads3, and ads4, and the table below shows to which combinations of features and TOPIC models they correspond.

Table 1: Results Identification

Result Set	Word Features	TOPIC Model
ads1	stemmed	model-1
ads2	unstemmed	model-1
ads3	stemmed	model-2
ads4	unstemmed	model-2

Although we generated four sets of results, the resource constraints at NIST resulted in only ads1 and ads2 being officially scored. Reference in the remainder of the paper to scores associated with ads3 and ads4 are to the unofficial score generated by us using the TREC-2 scoring program and the published qrels for the routing topics.

4.1 The Experimental Procedure

The experimental procedure for TREC-2 consists of five basic steps. We briefly describe each of these:

- First, we generated the CART training data from the information need statements and the ground truth files (i.e., the qrels) provided by NIST. This produced two feature sets for each topic (corresponding to the stemmed and unstemmed versions of the features), and two sets of training vectors labelled with the ground truth information.[9] Since CART is a statistically-oriented classifier, we decided to minimize the "noise" in the training sets by using only the Wall Street Journal articles identified in the qrel files. Further, for all but topics 80 and 81, we used just the Wall Street Journal articles on Disk 2.
- Second, we grew the CART trees from this training data. Since we had two sets of training data for each topic, we grew two trees for each topic.
- Third, we used the algorithms described in Section 3 to convert the CART trees into a TOPIC readable form. This produced four TOPIC definitions for each of the information need statements. Table 1 above shows the various combinations.
- Fourth, we ran the TOPIC definitions against the indexed unseen data.[10] Again, to minimize noise effects, we used only the Associated Press articles on Disk 3 to generate our official results.
- Fifth, we sorted and merged the results generated by TOPIC and converted them into the TREC format for scoring by NIST.

4.2 Discussion of Official Results

The official results for ads1 and ads2, together with the unofficial results for ads3 and ads4, are shown in Table 2.

Table 2: TREC-2 Results (AP Only)

Run ID	No. Retr.	No. Rel.	Rel. Ret.	Av. Prec.	Exact Prec.
ads1	40,423	5,677	822	0.0195	0.0390
ads2	33,034	5,677	1,468	0.0821	0.1092
ads3	49,006	5,677	1,182	0.0168	0.0374
ads4	50,000	5,677	1,847	0.0630	0.0868

The first observation is that the trees built using exact words as features (i.e., results ads2 and ads4) had higher precision than those built using word stems. We

8. A variable is in the optimal tree if its k-value is greater than k*; is on the fringe if k=k*; and outside the optimal tree if k<k*. Note that in general the individual features appear at multiple locations in the tree. Our strategy is to remove duplicates by retaining the instance with the highest k-value.

9. The feature specification and extraction procedure we used is identical to that used in TREC-1 and is described in detail in the TREC-1 proceedings. The only differences are the addition of a stemmed version of the features and the fact that we do not make use of the feature count information.

10. We are grateful to Verity Inc. for allowing us to have access to their computer systems and databases.

believe that the explanation for this is that by using stemmed versions of the features we added a significant amount of "noise" to the sample space. That is, given the relatively small size of the training sets, using stems tends to reduce the discriminating power of any given feature with respect to the training sets. This manifests itself indirectly in two ways. First, the optimal trees built with stems are generally smaller than those built using exact words; and, second, optimal trees built with stems have higher cross-validation error rates than those built using exact words.

The second observation is that the TOPIC trees built using model-2 (i.e., results ads3 and ads4) had better recall than those built using model-1. The explanation for this is straightforward. Since the model-2 trees essentially use all the features extracted from the information need statement in a generalized disjunction, they provide much broader coverage than the model-1 trees which often use just one or two of the features.

The third observation is, of course, that these are not strong results since on average all the models performed in the low end of the scores reported by NIST in the summary routing table. This is somewhat disappointing since our results in TREC-1 led us to believe that we might be able to do significantly better.[11]

Notwithstanding the fact that we did not explore some of the ideas discussed in the TREC-1 paper (e.g., the use of concepts rather than words as features, and the use of surrogate split information), we are now inclined to the view that the output from tools like CART are best used as the basis for manually constructed routing topics. To begin to explore this idea, we performed a number of auxiliary tests that we report in the following section.

4.3 Auxiliary Experiments

To explore the idea of using the CART output as the "skeleton" for a manually constructed routing query, we selected two model-2 trees to determine whether a minimal set of "edits" could significantly improve their performance. We selected Topics 52 and 54 since they represented one topic for which the automatically generated tree did well (Topic 52) and one for which the automatically generated tree did poorly (Topic 54).

The scores for Topic 52 (for the AP corpus only) are shown below:

```
Queryid (Num):       52
Total number of documents over all queries
    Retrieved:     1000
    Relevant:       345
    Rel_ret:        328
Interpolated Recall - Precision Averages:
    at 0.00       0.6667
    at 0.10       0.4684
    at 0.20       0.4032
    at 0.30       0.4032
    at 0.40       0.4032
    at 0.50       0.4032
    at 0.60       0.4013
    at 0.70       0.4013
    at 0.80       0.4013
    at 0.90       0.4003
    at 1.00       0.0000
Average precision (non-interpolated) over
all rel docs:
                  0.3828
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.3000
  At   15 docs:   0.4667
  At   20 docs:   0.6000
  At   30 docs:   0.6333
  At  100 docs:   0.4100
  At  200 docs:   0.3550
  At  500 docs:   0.3820
  At 1000 docs:   0.3280
R-Precision (precision after R (= num_rel
for a query) docs retrieved):
    Exact:        0.3739
```

Here we see that this topic tree does well — recall is excellent and precision is sustained even at high recall levels.

In contrast, the topic tree for Topic 54 produces the following results:

```
Queryid (Num):       54
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        65
    Rel_ret:         64
Interpolated Recall - Precision Averages:
    at 0.00       0.1323
    at 0.10       0.1323
    at 0.20       0.1323
    at 0.30       0.1323
    at 0.40       0.1323
    at 0.50       0.1323
    at 0.60       0.1323
    at 0.70       0.1102
    at 0.80       0.1102
    at 0.90       0.1097
    at 1.00       0.0000
Average precision (non-interpolated) over
all rel docs:
                  0.0907
```

11. We do note, however, that for a number of topics we did rather well in comparison with other systems (i.e., Topics 51 and 75), and that in absolute terms we produced a number of trees that had greater than 30% R-precision (i.e., for Topics 52, 58, 78 and 93).

```
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0500
  At  200 docs:   0.0600
  At  500 docs:   0.1080
  At 1000 docs:   0.0640
R-Precision (precision after R (= num_rel
for a query) docs retrieved):
    Exact:        0.0308
```

Thus although recall is very good, precision is completely unsatisfactory.

Our conjecture is that the automatically constructed model-2 trees while generally giving good recall give poor precision because they contain many extraneous features, or features that should be combined. To illustrate this, we considered the model-2 tree for Topic 52, as a starting point for a manually constructed tree. The initial model-2 tree is:

```
Topic_52 <Or>
*  0.86 Topic_Style_52_2 <Accrue>
**  0.75 "AFRICA"
**  0.25 "AFRICAN"
**  0.75 "APARTHEID"
**  0.25 "ARMS"
**  0.25 "BAN"
**  0.25 "BLACK"
**  0.25 "COMPANY"
**  0.25 "COMPLIANCE"
**  0.25 "CONTRACTS"
**  0.25 "CORPORATE"
**  0.25 "DISCUSS"
**  0.25 "DOCUMENT"
**  0.50 "DOMINATION"
**  0.25 "GOVERNMENT"
**  0.25 "INTERNATIONAL"
**  0.25 "INVESTMENT"
**  0.25 "ORGANIZATION"
**  0.25 "PRESSURE"
**  0.75 "PRETORIA"
**  0.25 "REDUCTION"
**  0.25 "RESPONSE"
**  0.75 "SANCTIONS"
**  0.75 "SOUTH"
**  0.25 "TIES"
**  0.25 "TRADE"
**  0.25 "UNITED"
```

Obviously there are a number of features here that are basically "noise" — for example the words "COMPANY" and "RESPONSE"; and other words are clearly elements of a larger phrase — for example the words "SOUTH" and "AFRICA". Notice that, in general, words with lower scores are always candidates for elimination.

The result of this pruning exercise was the following revised definition for Topic 52:

```
Topic_52 <Or>
*  0.86 TopicStyle_52-2 <Accrue>
**  0.50 S_Africa <Accrue>
***  0.50 'SOUTH AFRICA'
***  0.50 "PRETORIA"
**  0.50 'SANCTIONS'
**  0.20 Topic_52_Support <Accrue>
***  0.50 "APARTHEID"
***  0.50 <Near>
****          'BAN'
****          'TRADE'
***  0.50 <Near>
****          'BAN'
****          'INVESTMENT'
```

So although we have added no new features, we have combined "SOUTH" and "AFRICA" and used this together with "PRETORIA" to define a concept called S_Africa. We have also used "APARTHEID"; and "BAN" with "TRADE" and "INVESTMENT" to define another concept called Topic_52_Support. Finally we adjusted the weights to give more prominence to S_Africa than Topic_52_Support.

The results for this modified topic description are:

```
Queryid (Num):       52
Total number of documents over all queries
    Retrieved:     1000
    Relevant:       345
    Rel_ret:        312
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       0.9780
    at 0.30       0.9766
    at 0.40       0.9603
    at 0.50       0.9067
    at 0.60       0.8620
    at 0.70       0.8620
    at 0.80       0.8620
    at 0.90       0.7422
    at 1.00       0.0000
Average precision (non-interpolated) over
all rel docs:
                  0.8305
Precision:
  At    5 docs:   1.0000
  At   10 docs:   1.0000
  At   15 docs:   1.0000
  At   20 docs:   1.0000
  At   30 docs:   1.0000
  At  100 docs:   0.9700
```

```
  At  200 docs:   0.8900
  At  500 docs:   0.6220
  At 1000 docs:   0.3120
R-Precision (precision after R (= num_rel
for a query) docs retrieved):
    Exact:        0.8493
```

So that although recall decreased slightly (we now retrieve 312 rather than 328 of the 345 relevant documents), the precision is improved by nearly 50 percentage points. Obviously, this is a significant improvement and was achieved with minimal manual input. The time required to make these changes was only of the order of 10 minutes.

We repeated this exercise with Topic 54, the initial model-2 tree for which is:

```
Topic_54 <Or>
*  0.93 Topic_Path_54_2 <Accrue>
**  0.75 "AGREEMENT"
**  0.75 "ARIANE"
**  0.75 "ARIANESPACE"
**  0.75 "ATLAS"
**  0.75 "COMMERCIAL"
**  0.75 "CONTRACT"
**  0.75 "DELTA"
**  0.75 "DOCUMENT"
**  0.75 "DOUGLAS"
**  0.75 "DYNAMICS"
**  0.75 "INDUSTRY"
**  0.75 "LAUNCH"
**  0.75 "MARTIN"
**  0.75 "MENTION"
**  0.75 "PAYLOAD"
**  0.75 "PRELIMINARY"
**  0.75 "RELEVANT"
**  0.75 "RESERVATION"
**  0.75 "ROCKET"
**  0.75 "SATELLITE"
**  0.75 "SERVICES"
**  0.75 "SPACE"
**  0.75 "TENTATIVE"
**  0.50 "TITAN"
```

Using the same kinds of procedures (i.e., removing extraneous words and combining words into phrases) we constructed the following modified tree:

```
Topic_54 <Or>
*  0.93 Topic_Path_54_2 <Accrue>
**  0.10 'AGREEMENT'
**  0.75 "ARIANE"
**  0.75 "ARIANESPACE"
**  0.75 Atlas_Rocket <Sentence>
***         "ATLAS"
***         "ROCKET"
**  0.75 Commercial_Satellite <Sentence>
***         "COMMERCIAL"
***         "SATELLITE"
**  0.75 "DELTA II"
**  0.75 "MCDONNELL DOUGLAS"
**  0.75 "GENERAL DYNAMICS"
**  0.10 "LAUNCH"
**  0.75 "MARTIN MARIETTA"
**  0.75 "PAYLOAD"
**  0.75 "ROCKET"
**  0.75 "SATELLITE"
**  0.75 "SPACE"
**  0.50 "TITAN"
**  0.75 'CONTRACT'
**  0.75 'LAUNCH SERVICE'
```

Notice that here we actually added words that were part of obvious proper names (i.e., the "GENERAL" of "GENERAL DYNAMICS", the "MARIETTA" of "MARTIN MARIETTA", and the "MCDONNELL" of "MCDONNELL DOUGLAS"), but otherwise nothing was added. We also adjusted the weights on 'AGREEMENT' and "LAUNCH" to de-emphasize their importance.

The results of running this modified query are:

```
Queryid (Num):       54
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        65
    Rel_ret:         65
Interpolated Recall - Precision Averages:
    at 0.00       0.5800
    at 0.10       0.5800
    at 0.20       0.5800
    at 0.30       0.5800
    at 0.40       0.5800
    at 0.50       0.4592
    at 0.60       0.4592
    at 0.70       0.4324
    at 0.80       0.3355
    at 0.90       0.1474
    at 1.00       0.0657
Average precision (non-interpolated) over
all rel docs:
                  0.3889
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.1000
  At   15 docs:   0.3333
  At   20 docs:   0.4000
  At   30 docs:   0.4667
  At  100 docs:   0.4500
  At  200 docs:   0.2700
  At  500 docs:   0.1260
  At 1000 docs:   0.0650
R-Precision (precision after R (= num_rel
for a query) docs retrieved):
    Exact:        0.4615
```

So that, again for very little manual input, we achieved a significant improvement in precision performance; and this time at no cost to recall.

It is interesting to compare our results with Verity's scores for these two topics. To do this we re-scored Verity's TOPIC2 results on the AP corpus alone.[12] For Topic 52, Verity's results were:

```
Queryid (Num):       52
Total number of documents over all queries
    Retrieved:     1000
    Relevant:       345
    Rel_ret:        317
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       0.9833
    at 0.20       0.9342
    at 0.30       0.8607
    at 0.40       0.8314
    at 0.50       0.7425
    at 0.60       0.7125
    at 0.70       0.6704
    at 0.80       0.6161
    at 0.90       0.3952
    at 1.00       0.0000
Average precision (non-interpolated) over
all rel docs:
                  0.7159
Precision:
  At    5 docs:   1.0000
  At   10 docs:   1.0000
  At   15 docs:   1.0000
  At   20 docs:   1.0000
  At   30 docs:   1.0000
  At  100 docs:   0.9000
  At  200 docs:   0.7900
  At  500 docs:   0.5820
  At 1000 docs:   0.3170
R-Precision (precision after R (= num_rel
for a query) docs retrieved):
    Exact:        0.6812
```

Here we see better recall (317 of the 345 relevant documents retrieved) but with slightly lower precision. The TOPIC2 tree for this topic is much more complex than the one we developed, which explains the better recall. Notice however that both trees gave perfect precision for the first 30 documents.

For Topic 54, Verity's TOPIC2 results were:

```
Queryid (Num):       54
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        65
    Rel_ret:         65
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       0.9130
    at 0.20       0.9130
    at 0.30       0.9130
    at 0.40       0.9000
    at 0.50       0.7609
    at 0.60       0.5942
    at 0.70       0.5679
    at 0.80       0.5049
    at 0.90       0.2027
    at 1.00       0.0927
Average precision (non-interpolated) over
all rel docs:
                  0.6838
Precision:
  At    5 docs:   1.0000
  At   10 docs:   0.8000
  At   15 docs:   0.8667
  At   20 docs:   0.9000
  At   30 docs:   0.9000
  At  100 docs:   0.5100
  At  200 docs:   0.2900
  At  500 docs:   0.1280
  At 1000 docs:   0.0650
R-Precision (precision after R (= num_rel
for a query) docs retrieved):
    Exact:        0.5846
```

This shows the same recall performance (i.e., all 65 relevant documents were retrieved) but substantially better precision performance. Through the first 30 documents TOPIC2 gave excellent results, whereas our modified model-2 result was only half as good. Again however, the TOPIC2 tree is much more complex, and required more effort to develop.[13]

Overall, we are impressed by the improved performance we were able to achieve with minimal manual effort. These auxiliary experiments provide at least suggestive evidence of the value of automatic generation of initial trees. The extent to which this is consistently achievable will require further investigation, and we hope to report on this in TREC-3.

5 Commentary

The official results of our TREC-2 experiments demonstrate that automatic construction of routing queries from training documents is indeed feasible. The queries produced are in fact binary classification trees that are optimal with respect to size (measured in terms of the number of terminals in the tree) and the estimated error rate of the tree. Unfortunately, however, these trees generally appear to have poor performance. In a few cases the trees were comparable with the results from other sites, but they mostly

12. We are grateful to Verity for allowing us to examine their TREC-2 results in detail.

13. We do not have precise figures for the amount of effort needed to build the Verity TOPIC2 trees, but in general each topic required several hours of effort.

seemed unsatisfactory. We have not been able to identify any specific conditions under which we could expect the CART trees to perform well.

We believe, nevertheless, that further experiments to assess the utility of using a variety of extensions to the basic CART technique would still be of interest. In particular, the use of "low-level" concepts as features, surrogate split information, larger training sets, and features drawn from the complete corpus rather than the information need statements, are all likely to assist in the construction of more effective trees.

The main focus of our TREC-2 effort has been to explore the idea that the CART trees can form the basis of a semi-automated approach to building knowledge-based descriptions of routing topics. To that end, we developed two techniques for converting CART output into a form that can be used by the TOPIC retrieval engine from Verity, Inc. In the first technique we perform a "lossless" transformation of the CART tree into a TOPIC tree. In the second we generalize the CART tree, although add no new features.

As the examples contained in the paper show, the TOPIC trees generated in the first canonical form are "skeletal." This is just as we would expect. Since CART is a parsimonious classifier, rather than a broad-based information retrieval tool, it produces minimally complex decision trees with respect to the expected misclassification rate.

From an information retrieval perspective, the second canonical form seems more like the ones we might have built by hand. It uses a range of features and gives them weights based on their ability to discriminate among the training data. In effect, we have used CART to indicate which of the features are of use in defining the topic and then generalized the CART decision function using our (external) knowledge of the information retrieval problem.

Using these automatically constructed TOPIC trees as a starting point, we conducted a limited series of tests to assess the impact of performing minimal "editing" of these trees. For the two topics selected, we were able to produce significant performance gains with edits that added no new information (at least at the level of the features used) and that took of the order of only a few minutes to implement.

We are encouraged by these results, and, while the generalization of them will require a more carefully controlled series of experiments, we are now of the opinion that the most effective role for machine learning techniques in information retrieval is as a tool for producing candidate descriptions of information need. These candidates can then be reviewed by end-users who can easily make obvious corrections and modifications. We intend to explore this idea in more detail and report on our results in TREC-3.

6 Future Research

There a number of directions in which we might develop the basic research ideas presented in this paper. We briefly consider a number of them here.

We currently use just two classes (relevant and not relevant), but nothing in CART prevents it working with multiple classes. For the document routing problem, there is a case to be made for adding a third class — unknown relevance. Adding such a class might allow us to make use of larger training sets without the costs associated with developing large ground truths.

One way of extending the skeleton TOPIC trees produced by our tool is to make use of external lexical resources. For example, we might investigate the use of WordNet as a way to expand each of the classification features into a set of related words. Similarly, we might investigate the use of TOPIC's own lexical resources (e.g., the thesaurus and Soundex tools) by replacing the unstemmed words in the topic outline files with the appropriate TOPIC operator (e.g., `<SOUNDEX> word`, or `<THESARUS> word`).

Although we have used CART as the module for building the initial classification tree, we might be interested in exploring other tree building tools that have been used in the machine learning community. For example, the C4.5 algorithm by Quinlan [6], or various algorithms based on Bayesian methods such as Minimum Message Length models and decision graphs [7]. All of these tools generate decision trees from training data but offer different mathematical philosophies to justify their approaches.

Finally, as in all machine learning problems, the initial choice of features over which to learn is extremely critical to the overall success of the process. An investigation of various extended feature definition tools (e.g., recognizing key phrase and proper names), as well as exploring the impact of making different assumptions from TOPIC about how the lexical tokens in the texts are to be treated, would almost certainly yield important insights.

References

1. Breiman, L.; Friedman, J. H.; Olshen, R. A.; Stone, C. J. Classification and regression trees. Pacific Grove, CA: Wadsworth & Brooks; 1984.

2. Crawford, S. L.; Fung, R. M.; Appelbaum, L.A.; Tong, R. M. Classification trees for information retrieval. In: Proceedings of the eighth international workshop on machine learning (ML91). San Mateo, CA: Morgan Kaufmann; 1991.

3. Tong, R. M.; Winkler, A.; Gage, P. Classification trees for document routing. In: Harman, D. K., ed. The first text retrieval conference (TREC-1). NIST Special Publication SP-500-207, Gaithersburg, MD: March 1993.

4. Fox, C. A stop list for general text. SIGIR Forum, 24(1,2):19–35; Fall/Winter 1989/90.

5. Paice, C. D. Another stemmer. SIGIR Forum, 24(3):56–61; Fall 1990.

6. Quinlan, J. R. C4.5: Programs for machine learning. San Mateo, CA: Morgan Kaufmann; 1992.

7. Oliver, J. J.; Dowe, D. L.; Wallace, C. S. Inferring decision graphs using the minimum message length principle. In: Proceedings 5th Australian joint conf. on AI. 1992.

The ConQuest System

Paul E. Nelson
VP of Research & Development

ConQuest™ Software

9705 Patuxent Woods Drive, Columbia, Maryland 21046
(410)-290-6290

Introduction

ConQuest software has a commercially available text search and retrieval system* called "ConQuest" (for Concept Quest). ConQuest is primarily an advanced statistical based search system, with processing enhancements drawn from the field of Natural Language Processing (NLP).

ConQuest participated in Category A of TREC, and so produced results for 50 test queries over the entire 2.3 Gigabyte database. In this category, we constructed queries and submitted results for two different ranking functions. These two functions tested the difference between local and global document relevancy, and are fully described later.

In TREC-2, ConQuest had a very strong showing. Our recall scores in particular improved by about 18 percentage points over the adjusted TREC-1 scores. Our precision scores were also very competitive.

The purpose of this paper is to discuss how we prepared for TREC-2: how queries were performed, what initial judgments were made and why, and interpretation of the results. Then, I will cover the tests which were performed after TREC-2, and how these tests clearly identify the areas where ConQuest could most effectively be improved.

System Architecture

For a complete discussion of the system architecture of ConQuest, see the TREC-1 conference proceedings, or call the author. The following overview is meant as a brief refresher.

ConQuest uses pre-built indexes to perform text database searches at fast speeds. In such a system, all text to be searched must first be indexed. These indexes are then used for all searching; the original document data is not required.

ConQuest uses a dictionary augmented with a semantic network for both indexing and queries. The dictionary is a list of words where each word contains multiple meanings. Each meaning contains syntactic information (part-of-speech, feature values), and a dictionary definition.

The semantic network contains nodes which correspond to meanings of words. These nodes are linked to other related nodes. Relationships between nodes are extracted from machine readable dictionaries. Some example relationship types include synonym, antonym, child-of, parent-of, related-to, part-of, substance-of, contrasting, and similar-to.

The ConQuest dictionary was generated automatically from several Machine Readable Dictionary (MRDs) sources, commercially available. This gives ConQuest the most robust and thorough coverage of English available. It is the completeness of coverage that drives performance gains in recall and precision.

Since ConQuest is a commercially available product, many additional components, not required for TREC-2, are also available, such as true client/server, graphical user interfaces, routing and dissemination, and sophisticated application program interfaces.

Query

Generally speaking, ConQuest attempts to refine and enhance the user's query. The result is then matched against the indexes to look for documents which contain similar concepts.

Queries are not "understood" in the traditional sense of natural language processing. ConQuest makes no attempt to deeply understand the objects in the query, their interaction. or the user's intent. Rather, ConQuest attempts to understand the meaning of each individual word and the importance of the word. It then uses the set of meanings and their related terms (retrieved from the semantic networks) as a statistical set which is matched against document information stored in the indexes.

* For additional information on ConQuest, please contact the author.

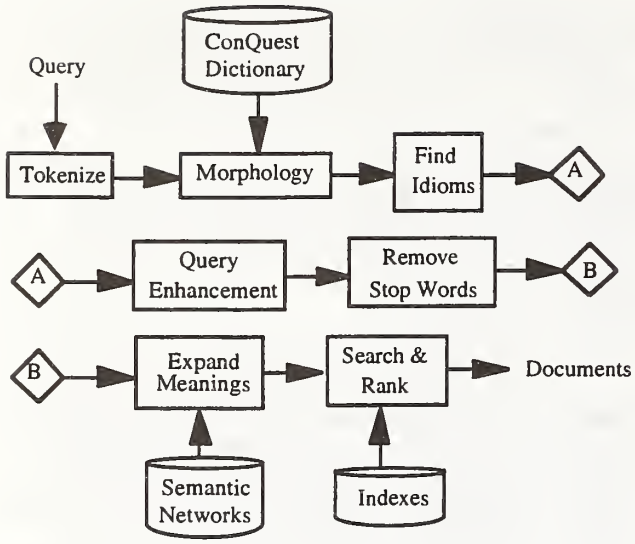


Figure 1 The Query Process

The following is a description of the modules used for query:

- *Tokenize:* Divides a string of characters into words.
- *Morphology:* An advanced form of stemming; attempts to remove suffixes and perform spelling changes to reduce words to simpler forms which are found in the dictionary. For example, one morphology rule will take "babies," strip the "ies," add "y," and produce "baby," which is found in the dictionary.
- *Find Idioms:* This module finds idioms in the text and indexes the idiom as a single unit. This prevents idioms such as "Dow Jones Industrial Average" from getting confused with queries on "industrial history." Words inside of idioms can still be located individually, if desired.
- *Query Enhancement:* The user is given the opportunity to enhance the query for additional improvement in precision and recall. There are many options available here, but the two most important are to choose meanings and weight query terms.

  Choosing a meaning of a word will restrict the expansion of words to only related terms which are relevant to the chosen meanings. This reduces noise in the query. When running in automatic mode, ConQuest expands all meanings of all words.

  Weighting query terms identifies the importance of the various words in the query. These weights are used by the search engine when ranking documents and computing document relevance factors.
- *Remove Stop Words:* Small function words—such as determiners, conjunctions, auxiliary verbs, and small adverbs—are removed from the query.
- *Expand Meanings:* Words in the query are expanded to include related terms.
- *Search and Rank:* ConQuest uses an integrated search and rank algorithm (described in the next section) which considers the relevance rankings of documents throughout the search process. Since ranking and search are integrated, the search engine automatically produces the most relevant documents right away.

Queries can be expanded to a very large number of terms, if desired. If the user wishes for the greatest amount of recall, a 5 word query can be expanded to 200 or 300 related terms.

Many other query features are also available in ConQuest, including wildcards, fuzzy spelling expansion, numeric and date range searching, boolean, mixed boolean and statistical, fielded searching (a variety of types), and searching over document categories.

Ranking Factors

Ranking and retrieval with ConQuest uses a variety of statistics and criteria, which are flexible and can be modified to handle varying requirements. The following are some of the factors used in ranking:

Completeness: A good document should contain at least one term or related term for each word in the original query.

Contextual Evidence: Words are supported by their related terms. If a document contains a word and its related terms, then the word is given a higher weight because it is surrounded by supporting evidence.

Semantic Distance: The semantic network contains information on how closely two terms are related.

Proximity: A document is considered to be more relevant if it contains matching terms which occur close together, preferably in the same paragraph or sentence.

Quantity: The absolute quantity of hits in the document is also included, but is not as strong a discriminator of relevance as the other factors.

ConQuest is the first truly "concept-based" search system to operate over unrestricted domains. If a document contains the word and some of its related terms, the word is more likely to be used in the correct context, using the "contextual evidence" factor above. In this way, ConQuest can determine word meanings at query time.

Coarse and Fine Grain Ranking

To further improve retrieval speed, ConQuest performs the search in two phases. The first is "coarse-grain." This phase is integrated with the document search process. Documents are output from the ConQuest search engine in descending coarse-grain rank order.

To compute the coarse-grain rank for a document, the statistics for the words contained in the document are combined using the coarse-grain ranking function. The inputs to this function include the semantic network

strength of each word, frequency in query, expansion terms, inverse document frequency, and query structure.

Once a document is found using coarse-grain rank, a second phase of relevancy ranking is applied, called "fine-grain" rank. This second phase uses a different ranking function which has access to more local information within the document. The inputs to this function include all of the inputs used in coarse-grain ranking, plus word location, proximity, frequency in document, and document structure.

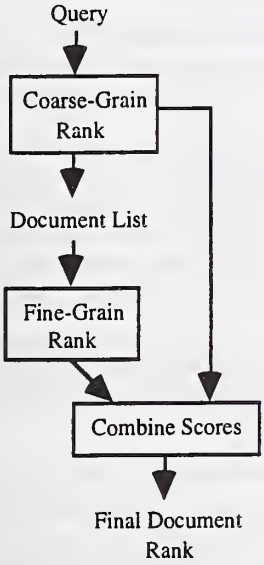


Figure 2 Fine and Coarse Grain Ranking

In general, the coarse-grain rank of a document represents global information on the document. It is a score that applies more to the document as a whole. The coarse-grain rank will be high for a document if it contains a large number of query words and related terms, ignoring the position of those terms in the document.

The fine-grain rank, on the other hand, represents local information, because the proximity (physical closeness) of the terms is the strongest contributor. The fine-grain rank of a document will be high if there is a single strong reference contained in the document.

As shown in Figure 2 above, the final document score is computed as a combination of the coarse- and fine-grain scores.

Pre-TREC Experiments

In preparation for TREC-2, ConQuest performed numerous experiments to improve the coarse-grain ranking algorithms and data. These experiments included the following:

1. Statistical word studies (statistical regressions to predict the probability that a document containing a word is relevant)
2. Statistical word-pair studies
3. Various weighting formulae
4. Various query structuring techniques

These studies were all performed under the assumption that the coarse-grain ranking formula used for TREC was weaker than the fine-grain ranking formula. The concern was that coarse-grain ranking did not retrieve a large enough percentage of relevant documents in the initial retrieval set. It was thought that once these documents were retrieved, the fine-grain algorithms would effectively use proximity and term frequency information to sort the documents and put all of the truly relevant ones at the top of the list.

Unlike other systems, ConQuest did not have funding for these TREC studies. This put the TREC studies in direct conflict with other more pressing concerns, such as supporting customers, or providing new functionality such as client/server.

As a result, the testing from these early studies proved ambiguous and unreliable. We believe that this was due to the following:

- Since time and resources were limited, tests were performed on only a small number of queries (5-10). This did not provide a large enough sample set of queries to produce reliable test results.
- ConQuest never tested the original assumption that coarse-grain was the limiting step in improving accuracy.
- The queries for this testing were taken from the TREC-1 final test queries. However, many of these queries were hastily constructed and thus added noise to the test results.

Just before the TREC-2 results were due, ConQuest decided to concentrate most of its effort on improving the tools used to generate queries. The tools and processes created are described in the next section.

Generating Queries for TREC-2

Generating queries was primarily an automatic process, based on the initial TREC-2 topic descriptions. Manual input was used primarily to remove things: Words, word meanings, and expansions. This produced queries with only the terms that are relevant. If needed, a user can also set weights for query terms.

Note that all manual steps were performed for all queries before any documents were retrieved. In other words, no feedback information was used in generating the queries. This makes ConQuest fully compliant with the rules for ad-hoc queries in TREC-2.

Automatic Query Generation Steps

A special program was created to convert TREC-2 topic descriptions into ConQuest query log files. The architecture of this program is show in Figure 3.

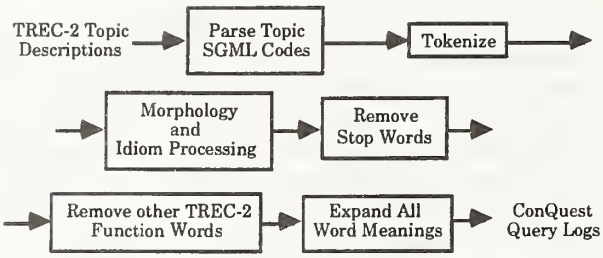


Figure 3 Program to Automatically Generate Query Log Files

The modules in the program are as follows:

Parse Topic - Reads through the topic looking for the SGML codes (such as <description>). The location within the topic for all words in the query are preserved in the final query log files.

Tokenize - Divides up strings into tokens.

Morphology - Locates all words in the dictionary and reduces them to root words if possible.

Idiom Processing - Collects idioms together as single terms, such as "United States."

Remove Stop Words - Removes conjunctions, determiners, auxiliary verbs, prepositions, etc.

Remove Function Words - Removes words such as "document," "relevant," and "retrieve" which are used often in TREC-2 narratives but do not help retrieval.

Expand Word Meanings - All word meanings are expanded using the ConQuest semantic network and all expansions are added to the query.

Note that all of these steps occur automatically with no manual input. The program also generates other statistics, such as the count of each term in the query, a count for each term for each section of the query (sections being the topic, description, narrative, concepts, and factors), and the total number of words in the query.

Manual Query Generation Steps

There were two manual steps used to generate queries:

1. Remove words, word meanings, and/or expansions
2. Set term weights (if necessary)

Fortunately, ConQuest has graphical user interfaces (GUIs) for removing words, word meanings, and expansions from the queries automatically generated. A user merely brings up the query and uses the mouse to select items to be deleted.

In TREC-2, terms were not weighted in the traditional sense, but rather were categorized into three sets:

1. Terms that embody the entire query, which would make good search terms if used by themselves
2. Terms which embody a necessary portion of the query. but not the entire concept
3. All other related terms

These categories provide simple guidelines for setting term weights, which make it much easier to generate queries. Evaluations using the TREC-2 test topics determined the functions for the actual term weights.

To emphasize once more, no document feedback was used for these manual steps. All query adjustments were performed without executing any query. Only after all queries were generated were the final results generated.

The TREC-2 Results

ConQuest scored very well in TREC-2. In particular, our recall percentages were quite high. Our average precision scores were not as good, but still competitive.

ConQuest submitted two sets of results for TREC-2, CnQst1 and CnQst2. Both sets used the same coarse-grain algorithm which retrieved the best 5000 documents from the database. The difference between the two results was how these 5000 documents were sorted to derive the top 1000 documents which were used for the official results.

The first set (CnQst1) used fine-grain as the only sorting algorithm. This algorithm primarily depends on local proximity information, although word statistics and query structure are also incorporated.

The second set of results (CnQst2) was a weighted average of the fine-grain and coarse-grain statistics for each document. As it turned out, this combination of local (fine-grain) and global (coarse-grain) statistics provided significantly better statistics.

The relatively modest addition of global information improved the results more than expected. Previous experience had always indicated that fine-grain information, especially the proximity test, was the strongest contributor to document relevancy.

Some additional insights can be extracted from topic analyses presented at the TREC-2 conference. Specifically, the topics where ConQuest excelled over other systems were also those which tended to have fewer relevant documents in the database. This indicates that local proximity statistics (used by ConQuest) are more important for these queries, since most other systems in TREC-2 are heavily weighted towards global document statistics. In other words, ConQuest appears to perform better for queries where one needs to find the "needle in the haystack."

Post TREC Analysis

After TREC-2, we had the chance to clean up our initial tests, gather new statistics, and perform some additional analysis.

The first step in this process was to prove the accuracy of the coarse-grain algorithm. Remember that initial tests attempted to improve the coarse-grain algorithm. But did the coarse-grain algorithm really need improvement? One indication that coarse-grain was accurate was provided by the CnQst2 run, which performed better than expected.

To check out the coarse-grain rank, we constructed graphs which more clearly shows its performance. Since fine-grain can only work on the results of the coarse-grain algorithm, what is the loss in recall for coarse-grain?

The following graph shows the cumulative recall percentage as documents are retrieved from coarse-grain rank. Every time a relevant document is retrieved, the recall percentage gradually inches up towards 100%. Note: these tests were run on just the Category B data.

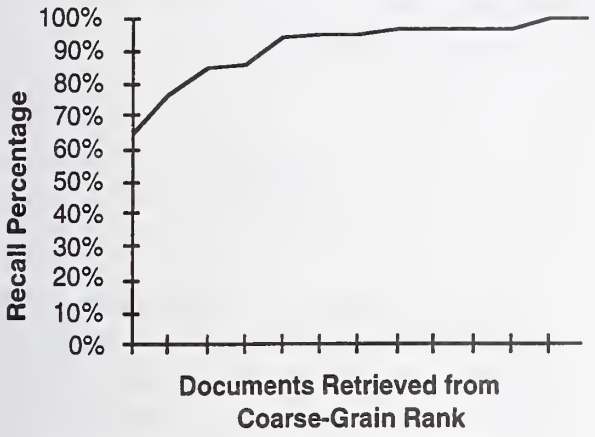


Figure 4 Cumulative Recall Percentage for Query #110

Figure 4 shows two exciting discoveries. The first is that the coarse-grain performance achieves over 95% recall. This strongly contradicts our initial fears that coarse-grain was not retrieving enough relevant documents.

The second discovery is that the high recall figures are achieved quickly. This implies that ConQuest can retrieve fewer documents (greatly improving speed) and still achieve high accuracy.

To further establish these claims, we repeated the analysis on all queries in the TREC-2 topic set, then averaged the results together, as shown in the next graph:

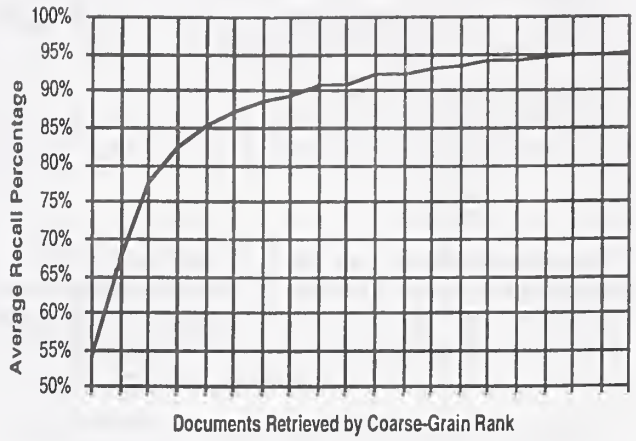


Figure 5 Cumulative Recall as Documents are Retrieved using Coarse-Grain Rank

This figure is an average over all queries. The average strongly correlates with the results from query #110. This verifies the two discoveries identified above.

Some initial studies also more clearly show the difference between fine-grain and coarse-grain sorting of documents. The following figure shows both graphs superimposed:

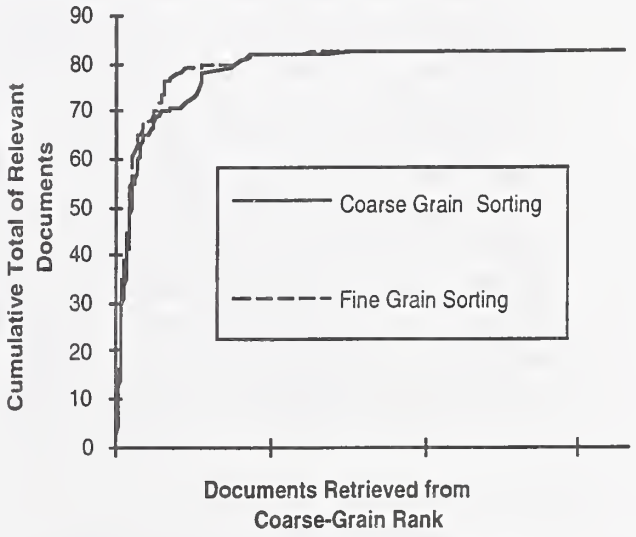


Figure 6 Coarse-Grain Sorting vs. Fine-Grain Sorting for TREC-2 Topic #135

In this diagram, we see that fine-grain sorting is in fact better than coarse-grain. In other queries, the results are more mixed. Clearly, the difference is not as great as was initially assumed.

This suggests that the area where ConQuest can most improve is not in the coarse-grain ranking algorithm, but rather in improving the fine-grain algorithm, or providing a better combination of the two.

Upon further study, we believe we now know why. When the fine-grain algorithm was developed, the programmers assumed an average query length of about 5 words. Studies of typical users indicate that their preferred query type is a

simple two or three word phrase. Therefore, a fine-grain rank tuned for short queries has provided the best and most accurate system for our commercial users.

In TREC-2, however, the queries are often 40-50 words long. These analyses have shown us that our initial fine-grain algorithms are not as accurate for such long queries.

Future Analysis

Our tests indicate that more study of other fine-grain algorithms is where ConQuest can most likely improve its scores for TREC-3. New ways of looking at proximity and positional information in the document will be explored and compared against the existing coarse-grain ranking results.

We still feel that our existing fine-grain algorithm is best for the typical commercial user, and we are looking for ways to fully test this hypothesis.

Finally, we are now much more sensitive to the effects of query size on fine-grain algorithms and are looking more closely at ways to desensitize our fine-grain algorithms, or to adapt them easily to different query lengths.

Description of the PRC CEO Algorithm for TREC-2

Paul Thompson
PRC Inc., Mail Stop 5S3
1500 PRC Drive
McLean, VA 22102
Phone: 612/687-4650
E-mail: thompson@research.westlaw.com
(current affiliation: West Publishing Company)

Abstract

This paper describes an application of the Combination of Expert Opinion technique to combine the results of multiple retrieval methods used on the TREC-2 collection. The methods being combined were weighted by their TREC-1 performance.

1. Introduction

This paper describes work done on the TREC-2 project at PRC Inc. in collaboration with Professor Edward Fox and his colleagues at Virginia Polytechnic Institute and State University (VPI&SU). The reader should refer to the description of their system included in these working notes for further details on the common processing of the TREC-2 data shared by PRC and VPI&SU (Fox et al. 1993). PRC used its algorithm, the Combination of Expert Opinion (CEO), to combine the results of VPI&SU's runs. VPI&SU used a different combination technique for their final results. Originally the intent was that the CEO algorithm would be integrated with the SMART system used by VPI&SU. Both upper and lower level combination of results would take place, i.e., at the lower level of individual document features within a particular retrieval method and the upper level of combination of the output of the individual methods themselves, i.e., the various cosine and p-norm methods used by VPI&SU. For TREC-1 we were not able to train the CEO algorithm, so that the weighting of the various methods would be optimized based on relevance judgments. For TREC-2 we used the 11 point average scores obtained by the various methods for TREC-1 for weighting. Again we only used the upper level of CEO. For TREC-1 we found that combining all methods resulted in lower performance than using the single best method. This year our first version combined the top two methods, based on TREC-1, while the second version used the top five methods.

2. Combination of Expert Opinion

The statistical technique of CEO provides a solution to the problem of combining different probabilistic models of document retrieval. This technique is expected to result in improved precision and recall over that provided by any one model, or method, since research has shown that various retrieval models retrieve different sets of more or less equally relevant documents (Katzer et al. 1982, Fox et al. 1988). In the

Bayesian formulation of the CEO problem (Lindley 1983) a decision maker is interested in some parameter or event for which he/she has a prior, or initial, distribution or probability. The decision maker revises the distribution upon consulting several experts, each with his/her own distribution or probability for the parameter or event. To effect this revision, the decision maker must assess the relative expertise of the experts and their interdependence, both with each other and the decision maker. The experts' distributions are considered as data by the decision maker, which is used to update the prior distribution.

For automatic document retrieval, the retrieval system is the decision maker, and different retrieval algorithms, or models, are the experts (Thompson 1990a,b, 1991). This is referred to as the upper level CEO. At the lower level the probabilities of individual features, e.g., terms, within a particular retrieval model can be combined using CEO. In lower level CEO the retrieval model is the decision maker and the term probabilities are viewed as lower level experts. The probability distributions supplied by these lower level experts can be updated, according to Bayes theorem, by user relevance judgments for retrieved documents. These same relevance judgments also give the system a way to evaluate the performance of each model, both in the context of a single search of several iterations and over all searches to date. These results can be used in a statistically sound way to weight the contributions of the models in the combined probability distribution used to rank the retrieved documents. Since various algorithms, such as p-norm, are expressed in terms of correlations rather than probability distributions, it was necessary to extend the CEO algorithm to handle correlations. So far this extension has been handled in a heuristic fashion. If a retrieval method, e.g., one of the cosine methods, returned a value between 0 and 1 as a retrieval status value; the logistic transformation of this weight was interpreted as an estimate of the mean of a logistically transformed beta distribution which was provided as evidence to the decision maker. Since there was no basis with which to assign a standard deviation to this distribution, as called for by the CEO methodology, an assumption was made that all standard deviations were .4045, a value corresponding to a standard deviation of .1 in terms of probabilities. The CEO code was written in g++.

For TREC-1 we used the CEO algorithm to combine all of the VPI&SU retrieval methods except for the Boolean, i.e., weighted and unweighted cosine and inner product measures as well as p-norm measures of 1.0, 1.5, and 2.0. For measures, such as the inner product and some of the p-norm results not giving a retrieval status value in the 0 to 1 range, the result was mapped to this interval by scaling the highest score of the method in question for a given topic to the highest score given by one of the cosine measures. Default scores half way between 0 and the lowest score achieved by a particular method were used for documents not retrieved in the top 200 in response to a given topic, since the actual score of these documents was unknown. For TREC-2 we followed the same approach except that only the results of methods with better TREC-1 performance were combined. Our first version used Cosine.atn and Cosine.nnn, the two best VPI&SU methods from TREC-1, weighted by their performance on TREC-1. The second version used these two methods and the next best three, Inner.atn, Inner.nnn, and Pnorm 1.0, also weighted by their TREC-1 performance (see VPI&SU report for details on these methods). Figures 1 and 2 show

our summary official TREC-2 results for versions 1 and 2 respectively.

Queryid (Num): all prceo1
Total number of documents over all queries
Retrieved: 50000
Relevant: 10785
Rel_ret: 3561
Interpolated Recall - Precision Averages:
at 0.00 0.4768
at 0.10 0.2613
at 0.20 0.1807
at 0.30 0.1202
at 0.40 0.0875
at 0.50 0.0504
at 0.60 0.0328
at 0.70 0.0169
at 0.80 0.0142
at 0.90 0.0077
at 1.00 0.0031
Average precision (non-interpolated) over all rel docs
0.0904
Precision:
At 5 docs: 0.2120
At 10 docs: 0.2480
At 15 docs: 0.2707
At 20 docs: 0.2760
At 30 docs: 0.2753
At 100 docs: 0.2418
At 200 docs: 0.1756
At 500 docs: 0.1086
At 1000 docs: 0.0712
R-Precision (precision after R (= num_rel for a query) docs retrieved):
Exact: 0.1703

Figure 1: Summary scores for PRC version 1 using two best experts

Queryid (Num): all prceo1
Total number of documents over all queries
Retrieved: 50000
Relevant: 10785
Rel_ret: 3323
Interpolated Recall - Precision Averages:
at 0.00 0.5963
at 0.10 0.3270
at 0.20 0.2306
at 0.30 0.1430
at 0.40 0.0866
at 0.50 0.0425
at 0.60 0.0323
at 0.70 0.0246
at 0.80 0.0209
at 0.90 0.0131
at 1.00 0.0041
Average precision (non-interpolated) over all rel docs
0.1120
Precision:
At 5 docs: 0.4120
At 10 docs: 0.4000
At 15 docs: 0.3920
At 20 docs: 0.3740
At 30 docs: 0.3527
At 100 docs: 0.2722
At 200 docs: 0.1974
At 500 docs: 0.1090
At 1000 docs: 0.0665
R-Precision (precision after R (= num_rel for a query) docs retrieved):
Exact: 0.1809

Figure 2: Summary scores for PRC version 2 using five best experts

3. Conclusion

Although selecting the best individual TREC-1 VPI&SU retrieval methods for combination and weighting them by their TREC-1 performance seemed to be a reasonable strategy which would yield better retrieval results than unweighted combination of all methods, in fact CEO performance was worse on TREC-2. This was due, in part, to changes made in the individual methods which made methods that had been best in TREC-1 less effective than other methods for TREC-2. These results suggest that selection and weighting of methods for combination based on performance of earlier versions of the methods is unwarranted.

References

Fox, E.A.; Koushik, P.; Shaw, J.; Modlin, R.; Rao, D. 1993. "Combining Evidence from Multiple Searches" this proceedings.

Fox, E.A.; Nunn, G.L.; Lee, W.C. 1988. "Coefficients for Combining Concept Classes in a Collection" *Proceedings of the 11th. International Conference on Research and Development in Information Retrieval*, June 13-15, Grenoble, France, p. 291-307.

Katzer, J.; McGill, M.J.; Tessier, J.A.; Frakes, W.; DasGupta, P. 1982. "A study of the overlap among document representations" *Information Technology: Research and Development*, vol. 2, p. 261-274.

Lindley, D.V. 1983. "Reconciliation of probability distributions" *Operations Research*, v. 31, no. 5, p. 866-880.

Thompson, P. 1990a. "A Combination of Expert Opinion Approach to Probabilistic Information Retrieval, Part 1: The Conceptual Model" *Information Processing and Management*, vol. 26, no. 3, p. 371-382.

Thompson, P. 1990b. "A Combination of Expert Opinion Approach to Probabilistic Information Retrieval, Part 2: Mathematical Treatment of CEO Model 3" *Information Processing and Management*, vol. 26, no. 3, p. 383-394.

Thompson, P. 1991. "Machine Learning in the Combination of Expert Opinion Approach to IR" In Birnbaum, L. and Collins, G. (eds.) *Machine Learning: Proceedings of the Eighth International Workshop (ML91)*, San Mateo: Morgan Kaufmann, p.270-274.

Report of Progress for TREC II

William Kelleher

Systems Environments Corporation

November 1, 1993

1.0 Introduction

Systems Environments is a commercial category B participant in TREC. For the past two years we have been developing a software system starting on a DOS based PC and recently moving to Windows NT on the same 486/25 PC. Much development time has been diverted toward getting around bugs and trying to fit the system on small (200 Mb disks and within the memory constraints of the DOCS system). At this time the system is still under development.

2.0 System Architecture

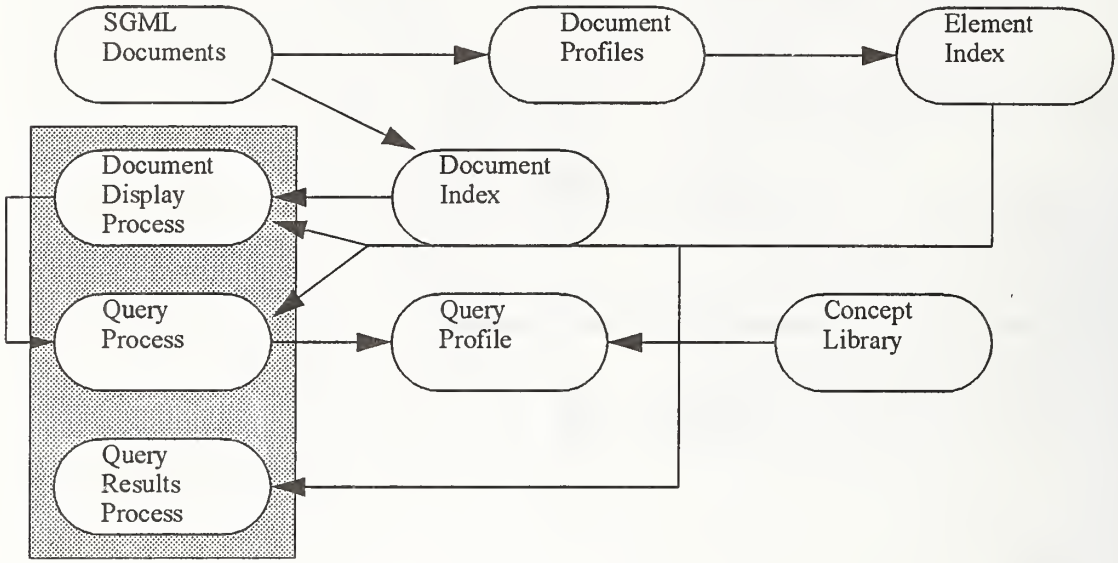
FORMS (Feedback, Object-oriented Retrieval Methods) is an object oriented, concept based information retrieval system.

FORMS is concept based because it creates a profile of the users information need and matches that profile to the document profiles. Profile can be stored in a library of concepts which is retained from one use of the system to another. Relevance feedback is an essential ingredient in the design of FORMS although so far it has not been used in TREC.

FORMS is designed and implemented using the object oriented approach. Object oriented methods were chosen to create a system architecture that can be installed in a variety of different environments using different architectures. In its smallest implementation FORMS can function as a personal information system on a single PC, or it can be installed as a organization wide system using client server approach. The basic objects and their relationships are shown in the following diagram and described in the following sections.

2.1 SGML Documents

A document arrives and first must be translated into an SGML (Standard Generalized Markup Language) Document.



2.2 Document Profiles

Elements are the things found in text documents that capture the meaning imbedded in the documents. The basic element in text is obviously the words. But there are other elements that can be very helpful. Identifying certain kinds of proper nouns can be crucial to determining the relevance of text to an information need. Important proper nouns might be persons, places, and some kinds of things, e.g. companies.

There are two distinct types of elements used in FORMS, simple and complex. A simple element consists of a character string, e.g. a word, and an element type, e. g. a noun or a noun phrase. Complex elements are various combinations of simple elements. For example a boolean 'and' of several simple elements occurring in the same sentence, paragraph or document would be a complex element.

In general simple elements are only used to create document profiles elements. A document profile element IS-A simple element that includes the document id and the number of occurrences of Th. element in the document.

2.3 Indexes

After the document profiles are created, those elements that occur in more than one document are stored in the element Index. An second index from the document ID to the full text of the document is also created so that the user can retrieve a document for inspection.

3.0 User Interface

The user interface objects (grey area in the diagram) provide object to display documents, retrieve documents using a natural language query as well as stored concepts, and a window that displays results.

3.1 Document Display

The document display window permits a user to enter a document ID and the system will simply retrieve the document from one of the text files and display it on the screen. When a document is displayed, the user has the option to indicate that the document is or is not relevant to the current query (or concept). When an indication of relevance is given the system modifies the concept/query profile and provides a new rank list of retrieve documents.

3.2 Query Window

The query window actually consists several parts. A natural language part for entering a query, and a query profile part that displays the elements of the query/concept along with various statistics about each element. Most important

is the probability that a document is relevant if it contains a particular element. Each element is treated as being independent of every other element.

3.3 Results Window

The results window contains the list of all documents with a probability of relevance to the current query/concept. Documents are ranked by probability of relevance. For each document, the elements found in the document are listed as well as any previous relevance information. The document ID selected from the results window can be used in the document window to examine the document.

4.0 Basic Assumptions

The approach being taken by FORMS is based a number of critical assumptions.

- Using the axioms of probabilities as a foundation for determining the relevance of documents to a query
- Elements besides words, such as phrases, can be found to make a significant contribution to retrieval accuracy
- A number of different approaches to identifying relevant elements are worth pursuing. These include proper noun identification, part of speech tagging, noun phrase tagging, and as yet undetermined relations that can be extracted from natural language.

5.0 Progress to Date

Frankly, there has been little progress to date beyond basic system development. In both TREC I and II, we have performed routing queries with very poor results. One obvious reason is that we have had to perform the analysis by breaking the texts into small sections and doing even class B in sections.

6.0 Future Work

FORMS is designed to provide information on the effectiveness of different approaches.

- To examine different approaches to incorporating relevance feedback into evaluating relevance of other documents.
- To examine whether queries can be analyzed and classified so that the system can determine which approach is most likely to be successful for a particular query or concept.
- To examine whether certain kinds of elements (where an element is a word, a phrase, a proper noun, a verb, a cooccurrence etc.) can be predetermined to be helpful in certain types of queries.

7.0 Results & Conclusion

At this time no results are supported by work performed with FORMS. However, it is worth emphasizing that the results in TREC seem to support the view that there is no specific approach that is going to revolutionize information retrieval. Rather, it seems that improvements are going to come from attention to details and finding the right element to use at the right time.

UCLA–Okapi at TREC–2: Query Expansion Experiments

Efthimis N. Efthimiadis* and **Paul V. Biron**

Graduate School of Library and Information Science
University of California at Los Angeles

1 Introduction

This is the first participation of the Graduate School of Library and Information Science, University of California at Los Angeles in the TREC Conference. For TREC–2, Category B, UCLA used a version of the Okapi text retrieval system that was made available to UCLA by City University, London, UK. OKAPI has been described in TREC-1 (Robertson, Walker, Hancock-Beaulieu, Gull & Lau, 1993a) as well as in this conference (Robertson, Walker, Jones, Hancock-Beaulieu, & Gatford, 1994). Okapi is a simple set-oriented system based on a generalized probabilistic model with facilities for relevance feedback. In addition OKAPI supports a full range of deterministic Boolean and quasi-Boolean operations.

1.1 Objectives

The main research objective of the UCLA participation in TREC–2 was to investigate query expansion within the framework as provided by Okapi. More specifically, the objectives were to:

- use an enhanced version of the Go-See-List (GSL) and evaluate its effect on retrieval performance.
- investigate the performance of query expansion with and without relevance information by varying the number of documents that are treated as relevant and the number of terms that are included in the expansion.
- compare the performance of different ranking algorithms for the ranking of terms for term selection during query expansion.
- compare the effectiveness in retrieval of user assigned relevance judgements against hypothetically assumed relevance judgements based on the top X documents.

*To whom all correspondence should be addressed. Graduate School of Library and Information Science, University of California at Los Angeles, 405 Hilgard Avenue, Los Angeles, CA 90024-1520, e-mail: iacxene@mvs.oac.ucla.edu

1.2 The Okapi version at UCLA and the WSJ database

The Okapi system consists of a low level search engine or basic search system (BSS), a user interface for the manual search experiments and data conversion and inversion utilities.

The UCLA hardware consisted of Sun SPARC-2 machine with 32 MB of memory, and 1 GB of disk storage.

The Wall Street Journal (WSJ) database was used for both the routing and ad-hoc searches. Because of the lack of adequate disk space on the UCLA machine the database was indexed at City University by Stephen Walker and it was then transferred (FTP-ed) to UCLA.

For TREC–2 the Okapi databases were built by indexing mainly the DOCNO and TEXT fields of the records. Inverted indexes included complete within-document positional information, enabling term frequency and term proximity to be used. Okapi's typical index size overhead is around 80% of the textfile size. The elapsed time for inversion of the WSJ database was about 12 hours.

At this point it is worth noting of (a) the nature of the WSJ records, and (b) a limitation of Okapi's due to indexing.

(a) The WSJ records consist of documents that do not have the same kind of structure found in bibliographic databases, such as INSPEC or ERIC. The records contain the full-text of stories and have varied length, mostly longer than the length of an average abstract of a bibliographic database. In addition, the language and the style is mostly 'journalistic' as opposed to 'scientific', i.e. less structured. One important issue is that some WSJ records often contain short multi-story articles which are completely unrelated one from the other. This type of record is usually a compilation of a number of one- or two-paragraph long news stories. The stories share no content relation between them, the only common feature is their co-existence in the same record. This has implications in retrieval effectiveness, especially when such records are included in the pool

of documents that provide terms for query expansion, because of the noise introduced by the terms taken from the irrelevant stories.

(b) This last issue relates to a limitation of Okapi. The version of Okapi used at UCLA retrieves documents at the record level only. Retrieval at the paragraph level, which would have facilitated a better handling of some issues like the above, is not currently available.

2 The weighting functions

The weighting of search terms can be said to involve two levels:

level 1: A weighting function is used to weight the terms for the initial query as well as the terms for subsequent search iterations of the same query or some modified version of the query.

level 2: A weighting function is used for the weighting of candidate terms for query expansion.

Sections 2.1 and 2.2 discuss functions used in level 1 and level 2 respectively.

2.1 Search term weighting

The theory of relevance weights (Robertson & Sparck Jones, 1976) provides the basic probabilistic model. The binary independence or relevance weight model assigns a weight to each term and the matching function for each document is given by the *'simple sum-of-weights'* over all of the terms in the query.

The weight of a term is calculated by following function which is also known as the *f4* point-5 formula:

$$w_{f4} = \log \frac{(r+.5)(N-n-R+r+.5)}{(n-r+.5)(R-r+.5)} \qquad (1)$$

where,

N is the total number of documents in the collection;
R is the sample of relevant documents as defined by the user's feedback;
n is the number of documents indexed by term t;
r is the number of relevant documents (from the sample R) assigned to term t.

When relevance information is not available the above weight reduces to approximately the inverse document frequency (IDF).

For calculating the total weight of a document the following function was used which is based on the binary independence model, and takes into consideration the 2-Poisson model for within document frequency (tf) and the document length. These are described in detail in Robertson et al (1993b). The purpose of the UCLA Okapi system was to evaluate the existing Okapi models and therefore did not allow for modifications of the existing functions. For compatibility purposes and for comparisons it was decided to use the BM15 (best match) function for the runs. The BM15 best match weigthing function is:

$$docweight_{bm15} = \sum((\frac{tf}{(k_1+tf)}) \times w_{f4}) + k_2 \times nq \times \frac{(avedl-dl)}{(avedl+dl)} \qquad ()$$

where k_1 and k_2 are unknown constants. In the UCLA-Okapi implementation the values for these constants are: $k_1 = 1$ and $k_2 = 1$.

2.2 Query expansion term weighting

The ranking algorithms that were considered for the ranking of terms for query expansion were: *wpq*, *emim*, *porter*, *r_lohi* and *r_hilo*. These algorithms are described briefly below.

2.2.1 The *wpq* algorithm

This algorithm is based on an independence assumption that holds between a query expansion term and the terms in the entire previous search formulation (Robertson, 1990). According to the relevance weighting theory, the inclusion of term t in the search formulation with weight w_t will increase the effectiveness of retrieval by

$$wpq = w_t(p_t - q_t) \qquad (3)$$

where, w_t is a weighting function, which in this case is the w_{f4}; p_t is the probability of term t occurring in a relevant document; and q_t is the probability of a term t occurring in a non-relevant document.

This means that irrespective of the weighting function (w_t) used the rule for deciding the inclusion of a term in a query expansion search should be based on the ranking of *wpq* instead of w_t alone. Substituting the weighting function and the probability of relevance in *wpq* with *r*, *R*, *n*, *N* we get:

$$wpq = \lg \frac{(r+.5)(N-n-R+r+.5)}{(n-r+.5)(R-r+.5)} \cdot (\frac{r}{R} - \frac{n-r}{N-R}) \qquad ()$$

The *wpq* algorithm combines the effects of the relevance weighting theory, as expressed by the w_{f4} component, which assign greater importance to the infrequent terms with the frequency of occurrence of a term in the relevant document set.

2.2.2 The *emim* algorithm

The expected mutual information measure (*emim*) is a term weighting model incorporating relevance information in which it is assumed that index terms may not be distributed independently of each other. (van Rijsbergen, 1977; Harper and van Rijsbergen, 1978; van Rijsbergen, Harper & Porter, 1981)

The *emim* weight reduces to the *f4* weight when the "degree of involvement", i.e. the joint probabilities, are all unity. Assuming the same definitions for *n, N, r, R,* as those already used earlier, the *emim* weight of a term is calculated as follows:

$$\begin{aligned} E_{iq} &= p_{11}i_{11} - p_{12}i_{12} - p_{21}i_{21} + p_{22}i_{22} \\ &= \log \frac{rN}{Rn} \cdot r \\ &\quad - \log \frac{(n-r)N}{(N-R)n} \cdot (n-r) \\ &\quad - \log \frac{(R-r)N}{(N-n)R} \cdot (R-r) \\ &\quad + \log \frac{(N-n-R+r)N}{(N-n)(N-R)} \cdot (N-n-R+r) \end{aligned}$$

2.2.3 The *porter* algorithm

Porter and Galpin (1988) describe a ranking formula used in the MUSCAT online catalogue:

$$porter = \frac{r}{R} - \frac{n}{N} \qquad (5)$$

where *r, R, n, N* are defined as in the *f4* weight (eq. 1).

2.2.4 The *r_lohi* algorithm

The *r_lohi* algorithm has been proposed by Efthimiadis (1993a) as the result of the observation of the ranking behavior of six algorithms used for ranking terms for query expansion.

The *r_lohi* ranking algorithm:

- ranks terms according to r, i.e. their frequency of occurrence in the relevant document set, in descending order and
- resolves ties according to their term frequency, n, from low-to-high frequency.

It was hypothesized that the *r_lohi* algorithm would have an almost identical ranking to *porter* and a performance approaching that of *wpq* and *emim*. More differences between the algorithms may occur if the size of the set of relevant documents (R) gets larger. Conclusions about the algorithm however could not be drawn before it was evaluated against the other algorithms. The results of that evaluation are reported in Efthimiadis (in press) where the *r_lohi* algorithm demonstrated better performance when compared to the other algorithms.

2.2.5 The *r_hilo* sort

A variant of the *r_lohi* algorithm is to rank candidate terms for query expansion using the *r_hilo* rank which:

- ranks terms according to r, i.e. their frequency of occurrence in the relevant document set, and
- resolves ties according to their term frequency, n, from high-to-low frequency.

Since the *r_hilo* algorithm will result in sorting terms in exactly the opposite way of the *r_lohi* algorithm it was included as a control for the study.

3 Methodology

3.1 Runs

Initial tests were performed in topics 1-50 where the dependent variables were the weighting function and the query processing of terms. ¿From the results obtained it was established that the function to use will be BM15 and that the parsing of the Topics would include both single terms and "phrases" as defined by comma delimited text in the Topics.

The table below (Table 3.1 gives all the variables used in constructing the runs. The options available for each variable are also provided.

Weighting Function: Best match function BM15 (see equation 2).

Phrases: Choice of YES, NO, or BOTH. This determines the type of parsing of the "Concepts" and "Title" fields

Table 3.1 Methodology for the Routing Runs on Topics 51-100						
Weighting Function	Phrases	QE	Query Expansion Algorithm	Number of Terms Expanded	No. of Docs used for Auto Rel Fbk	UCLA GSL
bm15	no yes both	no yes	*wpq* *emim* *porter* *r_lohi* *r_hilo*	0 10 20 30	0 5 10 15 20	no yes

from the Topics, which were the source of the search terms. NO means that the terms extracted from the Concepts and Title fields are single terms only. YES means that phrases get extracted as determined by the simple routine, where a phrase is identified by using the punctuation found in the Concepts and Title fields. BOTH is the combination of the two methods and the terms are searched as single terms as well as phrases.

Query Expansion (QE): The choice of query expansion algorithms is one of *wpq*, *emim*, *porter*, *r_lohi*, *r_hilo*.

Terms expanded: This specifies the number of terms to include in the expansion. When the number of terms expanded is zero, then only the initial query is run.

Feedback documents: This defines the number of top ranked documents to be treated as relevant and to provide the source for the terms for query expansion.

UCLA GSL: defines whether the standard Okapi GSL or the UCLA enhanced version of the GSL will be used.

Because of the many parameters involved in each run the names of runs have been deliberately made explicit, which however resulted in rather long names. For example, `bm15.phb.qey:r_lohi-10-5.uclagsly` means that for this run the weighting function used was the BM15, phrases were set to BOTH, query expansion took place, the *r_lohi* algorithm was used for the ranking of terms for query expansion, 10 terms were added in the expansion, 5 documents provided the source of the terms for the expansion, and the UCLA enhanced GSL was also used.

3.2 Go-See-List

The Go-See-List (GSL) is a look-up table that contains stopwords, semi-stopwords, prefixes, go-phrases and synonym classes. The GSL is used during the indexing of a database as well as during searching.

Stopwords contain an array of terms that are thought to contain no or little value for retrieval. These include, contractions, prepositions, adverbs, etc.

The semi-stopwords are terms that are thought to have low value for retrieval purposes. Therefore, a semi-stopword will be searched only during the initial search if it has been part of the user's search statement. If, however, the term has emerged as the result of a query expansion it is stopped, i.e. excluded from the pool of candidate terms for query expansion.

Go-phrases are mostly noun-phrases that need to be searched as one word or else the precision will be very low, e.g. New York. GSL contains a small number of selected go-phrases.

Synonym entries contain a mix of terms/concepts that are treated as synonyms for retrieval purposes. These may be true synonyms, quasi-synonyms, or unrelated semantically terms which are grouped together because of some common properties which have value for retrieval. Finally, the synonym entries also contain term variants that are known to "escape" from the conflation algorithm. The structure of the UCLA GSL is given in the table below.

The Go-See-List (GSL)			
	City	added by UCLA	UCLA total
stopwords	411	72	483
semi-stopwords	58		58
prefixes	18		18
Go-phrases	43	84	127
Synonyms	359	604	963

For the UCLA GSL, the Titles and Concepts of Topics 1–100 were analyzed and synonym classes were generated from the data. The list includes: 40 personal names, and 250 synonym classes. In addition, a list of organizations and a list of common business acronyms and abbreviations was compiled.

3.3 Query term selection

Query terms were selected from the Title and Concepts fields of the records. The processing of these fields was very simple. Programs written in `awk` and `perl` were used to isolate the required fields, which were then parsed and the resulting terms stemmed in accordance with the indexing procedures followed for building the WSJ database.

This process resulted in one-word query terms. When appropriate the procedure also output phrases by treating the punctuation available in these fields as the phrase delimeter.

Queries were then generated automatically from the Title and Concepts fields. Exactly the same queries were used in the **Routing** and **Ad hoc** searches.

3.4 Term selection for query expansion

a) **Routing searches:** Query expansion in the routing searches was performed through query modification without relevance information. As indicated in the table, that describes the construction of the runs inthe methodology section, the number of documents used could range from the top 0–20 documents, in increments of 5 documents. These top ranked documents were treated as relevant and were analyzed in order to provide terms for the expansion. Expansion terms were selected by pooling all the terms and then weighting these terms with one of the five ranking algorithms as specified by the run. Then the top 10, 20 or 30 terms were added to the original query terms and searched.

b) **Ad hoc searches:** The term pool consisted of all the terms of the documents judged as relevant. For the **Ad hoc** searches with feedback of the official results, the top 10 terms as determined by *wpq* were chosen for expansion and were searched together with the initial query terms.

c) **Rules for term selection:** The following rules were followed for the inclusion or exclusion of a term during selection for query expansion:
a) numbers were excluded as terms,
b) all terms whose frequency (n) is equal to the number of relevant documents seen (R), i.e., if $n <= R$, were excluded.

3.5 Search procedure

All searches, **Routing** and **Ad hoc**, were automatic and determined by the specifications made for each run. There were no manual searches.

3.5.1 Ad hoc searches and searchers

There were no manual searches. For the **Ad hoc** searches with relevance feedback, i.e. uclaf1 (official results), relevance assessments were provided by two searchers. The odd numbered topics were assessed by one searcher and the even numbered topics by the other.

3.5.2 Relevance assessments

During the **Ad hoc** searches, the guidelines for relevance judgements were:

a) review the entire document, when judging relevance, even if it seems to be peripheral or not relevant. The reason being that many of the articles were found to be collections of brief news stories, with the relevant part of the text hidden in (the middle or the end of) the text.

b) target for 10 relevant documents; stop as soon as 10 are found or at the 20th document. However, if 3 relevant have not been found continue till 3 are found (this is because OKAPI will not do an expansion if it has less than 3 documents).

3.5.3 Ad hoc additional runs

Following the TREC conference, a set of runs was conducted on the **Ad hoc** queries in order to complete the evaluation of the five ranking algorithms for query expansion that were studied.

The relevance judgements made in the **Ad hoc** run **uclaf1** (fdbk.bm15.phb.qey:wpq-10-10.uclagsly) were extracted and used in the subsequent runs. The process followed in these additional runs is described below:

- Four new **Ad hoc** runs were done; one for each of the remaining algorithms which were used for the ranking of terms for query expansion, i.e., *emim*, *porter*, *r_hilo*, *r_lohi*.
- The same initial query, which was generated automatically, was used for all searches.
- The relevance judgements made in the initially retrieved set of the official **Ad hoc** run were extracted and then simulated in the additional runs.
- Query expansion terms were ranked using the algorithm that was designated by each run. The 10 top ranked terms from the pool were added to the query.

3.6 Problems & Limitations

Lack of equipment has been a major problem in our participation. In order to participate in TREC, SUN Microsystems provided an equipment grant (SUN Sparc–2) in March, however no disk was initially available, but a 1-Gigabyte disk was acquired in June. Consequently, only the **Ad hoc** runs were included in the official results.

A limitation of the UCLA version of OKAPI is that it does not allow modifications of the basic retrieval functions (i.e., the BMs or best match functions).

4 Results and Discussion

The results of the `Routing` runs, the `Ad hoc` runs and the `Ad hoc` additional runs are given in Table 1, Table 2 and Table 3 respectively.

Routing runs

The 35 `Routing` runs given in Table 1 are presented in descending
recall values. The runs `bm15.ph[ynb].qen.uclags1[yn]`, i.e., the runs without query expansion, were used as baseline runs in order to facilitate comparisons. All other runs reported in the table include query expansion.

The results indicate that runs with query expansion, where the *r_lohi* or the *r_hilo* algorithm was used performed better than all other runs in terms of Recall, Average Precision, and R-Precision.

Ad hoc runs

>From the three official `Ad hoc` runs, `uclaa1`, was the automatic run that did not include query expansion and has been used as a baseline-run, `uclaa2`, was an automatic run that included query expansion without any relevance information, and `uclaf1`, was a run with user supplied relevance feedback and query expansion.

In terms of R-Precision and Average Precision the run with feedback and query expansion (`uclaf1`) did better than the automatic run with query expansion (`uclaa2`), but the baseline was slightly better.

Ad hoc additional runs

The results of the `Ad hoc` additional runs are given in Table 3. The official run with feedback (uclaf1) using *wpq* for the expansion is compared to the runs which used the *r_lohi*, *r_hilo*, *emim* and *porter* algorithms respectively for the expansion. The results indicate that *r_lohi* and *r_hilo* have performed better than the other algorithms. These results further corroborate the results obtained from the routing runs.

In order to further validate the results the *sign test* as well as the *t-test* were performed on the data. The results from the sign test are given on Tables 4–15. The tables are arranged in sequence starting from Precision at 15, 30, and 100 documents, Average Precision, Recall-Precision, to Recall. In each case, two tables are given; the first table gives the differences and the second the probabilities. As it can be expected there are no differences at Precision at 5 documents and at Precision at 10 documents because these were the same for all five runs. For this reason the corresponding pairs of tables have not been included in the paper. The results also show no significant differences at Precision at 15 documents and at 30 documents. Significant results appear at Precision at 100 documents where $r\_lohi \approx r\_hilo > emim \approx wpq \approx porter$.

The sign test results on Average Precision demonstrate that $r\_lohi \approx r\_hilo > wpq \approx emim \approx porter$, where $emim > porter$. The results on Recall show some grouping between the algorithms, so that $r\_lohi \approx r\_hilo > emim \approx wpq > porter$. The results from the Recall-Precision indicate that $r\_lohi \approx r\_hilo \approx emim > wpq \approx porter$ with $r\_lohi > emim$ but not significantly better and with *wpq* slightly better than *porter*.

>From the study of the sign test results certain overall comments emerge about the performance of the five algorithms. The results seem to be consistent throughout with *r_lohi* performing better than the other algorithms. Differences between *emim*, *wpq* and *porter* are not consistent but it seems that *emim* is slightly better than *wpq* which is better than *porter*.

To further strengthen the validity of the results the t-test was per formed on the data. The t-test results are given on Tables 16–21. The tables are arranged in sequence from Precision at 15, 30 and 100 documents, Average Precision, Recall-Precision, to Recall. Each table gives the Mean difference, the standard deviation difference, the t-statistic and the probability. As in the case with the sign test there were no differences for Precision at 5 documents and Precision at 10 documents and therefore the corresponding tables have not been included in the paper. Similarly, there are no significant differences at Precision at 15 documents and Precision at 30 documents. The results at Precision at 100 documents show that $r\_lohi \approx r\_hilo > emim \approx wpq \approx porter$, this result is the same as the sign test. The results from Average Precision demonstrate that $r\_lohi \approx r\_hilo > emim \approx wpq \approx porter$, with *emim* better than *porter*. For Recall the results are that $r\_lohi \approx r\_hilo > emim \approx wpq > porter$. Finally, the Recall-Precision results demonstrate that $r\_lohi \approx r\_hilo \approx emim > wpq > porter$, where *r_hilo* is better than *emim*.

The results of the t-tests are consistent for the algorithms

and corroborate the results obtained from the sign tests. The two tests indicate that *r_lohi* and *r_hilo* have performed consistently better than the other algorithms.

5 Conclusions

- The results obtained from the use of the standard and enhanced versions of the GSL indicate that further research is needed in order to determine the effectiveness of the GSL–synonym list in retrieval.
- The combination of adding 10 terms from the 5 or 10 top ranked documents contributed to better retrieval performance.

  The other term/document combinations, i.e. adding 20, 30, or 40 terms from 15 or 20 documents, etc., had a negative effect on retrieval performance.
- The results from the routing searches indicate that query expansion (i.e., feedback searches without relevance information, where X number of terms is extracted from Y number of top ranked documents that are treated as relevant to the query) improved retrieval performance depending on the algorithm used.
- The *r_lohi* algorithm (Efthimiadis, 1993a) improved retrieval performance in the routing runs when compared to the initial (baseline) search which did not involve either a feedback search or query expansion.
- In the `Ad hoc` searches the results of the evaluation of the five ranking algorithms indicate that *r_lohi* performed better than the other algorithms. These results were further validated by the results obtained from the sign test and the t-test.
- Although query expansion seems to work, the retrieval performance achieved was less than expected.

  There are many reasons that account for these results and which are briefly addressed below.

  1. Completeness of the TREC Queries:
     The major factor that is being attributed to these results is that the queries, i.e. TREC Topic Descriptions, are almost complete, i.e. contain all the important words required for the search.

     Query expansion is the process of supplementing the original query terms and is particularly effective when incomplete queries are available.

     Query expansion on these rather complete queries seemed to have contributed to a small or even a detrimental effect in overall retrieval performance.
  2. Size of the TREC collection:
     The large size of the TREC collection raises the issue of scalability and effectiveness of retrieval algorithms. The TREC collection is very different from that of the standard IR test collections, such as ADI, Cranfield, CACM, NPL. TREC is 1-4 Gigabytes of text whereas the other collections are smallish in size, i.e., only a few (1-50) Megabytes. The behavior and effectiveness of algorithms in information retrieval has been studied in small collections and TREC provides the challenge of scalability.
  3. Nature of documents:
     The documents in the WSJ database are mostly long documents; full-text as opposed to short bibliographic records; less structure when compared to bibliographic records; and with language and presentation less structured (journalistic style compared to scientific style);
  4. Length of documents:
     The records are long and often contain short multi-story, usually unrelated, items.

     When such documents contain relevant information for a topic, i.e., when one of the stories is relevant but all the others are not, these increase noise and interfere with the selection of terms for query expansion. This is because all the terms of that document will be included in the pool of the terms for query expansion and there may be a number of terms from other stories in that document that will be ranked higher than the terms from the relevant story.

     This reinforces the need to be able to retrieve at a paragraph level rather than at a document level.

6 Future Research

- evaluate in detail the level of the effect of the GSL-synonym list in retrieval performance
- evaluate the different effect of a local versus a global thesaurus for query expansion
- evaluate the effect of variable bias in query expansion term weighting
- investigate the retrieval overlap between different approaches, and
- explore data fusion techniques for output integration

Table 1: Runs with and without query expansion for topics 51-100.
(Runs are presented in descending 'Recall' values.)

Run_name	Avg Prec	Prec[5]	Prec[10]	Prec[15]	Prec[30]	Prec[100]	R-Prec	Recall
bm15.phb.qey:r_lohi-10-5.uclagsln	0.2960	0.5640	0.5160	0.4893	0.4400	0.3288	0.3465	0.7322
bm15.phb.qey:r_lohi-10-10.uclagsln	0.2960	0.5640	0.5160	0.4907	0.4400	0.3288	0.3472	0.7320
bm15.phb.qey:r_lohi-10-15.uclagsln	0.2961	0.5680	0.5160	0.4907	0.4400	0.3288	0.3472	0.7320
bm15.phb.qey:r_lohi-10-10.uclagsly	0.2960	0.5640	0.5160	0.4907	0.4400	0.3288	0.3472	0.7320
bm15.phb.qey:r_hilo-10-10.uclagsln	0.2860	0.5480	0.4900	0.4733	0.4167	0.3148	0.3327	0.7283
bm15.phb.qey:r_hilo-10-10.uclagsly	0.2860	0.5480	0.4900	0.4733	0.4167	0.3148	0.3327	0.7283
bm15.phn.qen.uclagsly	0.2849	0.5560	0.5040	0.4720	0.4200	0.3164	0.3328	0.7264
bm15.phn.qen.uclagsln	0.2849	0.5560	0.5040	0.4720	0.4200	0.3164	0.3328	0.7264
bm15.phb.qen.uclagsly	0.2846	0.5560	0.5040	0.4720	0.4200	0.3166	0.3325	0.7262
bm15.phb.qey:emi-10-10.uclagsly	0.2746	0.5240	0.4960	0.4573	0.4027	0.2962	0.3106	0.7113
bm15.phb.qey:emi-10-10.uclagsln	0.2746	0.5240	0.4960	0.4573	0.4027	0.2962	0.3106	0.7113
bm15.phb.qey:r_lohi-20-5.uclagsln	0.2968	0.5760	0.5240	0.5027	0.4527	0.3362	0.3508	0.7060
bm15.phb.qey:r_lohi-20-15.uclagsln	0.2969	0.5800	0.5280	0.5067	0.4520	0.3362	0.3512	0.7060
bm15.phb.qey:r_lohi-20-10.uclagsln	0.2967	0.5800	0.5260	0.5067	0.4520	0.3364	0.3512	0.7060
bm15.phy.qen.uclagsly	0.2406	0.4600	0.4680	0.4360	0.3800	0.2878	0.2967	0.6777
bm15.phb.qey:wpq-10-10.uclagsln	0.2590	0.5360	0.4980	0.4440	0.3900	0.2884	0.3017	0.6768
bm15.phb.qey:wpq-10-10.uclagsly	0.2590	0.5360	0.4980	0.4440	0.3900	0.2884	0.3017	0.6768
bm15.phb.qey:wpq-10-15.uclagsln	0.2570	0.5160	0.4780	0.4520	0.3953	0.2882	0.2963	0.6750
bm15.phb.qey:r_lohi-30-15.uclagsln	0.2896	0.5760	0.5320	0.5173	0.4573	0.3388	0.3456	0.6691
bm15.phb.qey:r_lohi-30-10.uclagsln	0.2890	0.5800	0.5420	0.5240	0.4520	0.3370	0.3445	0.6683
bm15.phb.qey:r_lohi-30-5.uclagsln	0.2894	0.5840	0.5380	0.5133	0.4520	0.3370	0.3450	0.6678
bm15.phb.qey:emi-20-10.uclagsly	0.2458	0.5280	0.4880	0.4547	0.3873	0.2696	0.2875	0.6599
bm15.phb.qey:wpq-20-10.uclagsln	0.2443	0.5120	0.4660	0.4413	0.3847	0.2706	0.2838	0.6437
bm15.phb.qey:wpq-10-5.uclagsln	0.2438	0.5440	0.4900	0.4533	0.3800	0.2740	0.2849	0.6423
bm15.phb.qey:wpq-10-5.uclagsly	0.2438	0.5440	0.4900	0.4533	0.3800	0.2740	0.2849	0.6423
bm15.phb.qey:por-10-10.uclagsly	0.2509	0.5320	0.4980	0.4560	0.4047	0.2862	0.2989	0.6362
bm15.phb.qey:por-10-10.uclagsln	0.2509	0.5320	0.4980	0.4560	0.4047	0.2862	0.2989	0.6362
bm15.phb.qey:emi-30-10.uclagsly	0.2377	0.5280	0.4860	0.4480	0.3780	0.2666	0.2809	0.6331
bm15.phb.qey:por-20-10.uclagsly	0.2558	0.5320	0.5080	0.4760	0.4113	0.2856	0.3030	0.6323
bm15.phb.qey:wpq-20-15.uclagsln	0.2342	0.5240	0.4940	0.4333	0.3827	0.2690	0.2701	0.6288
bm15.phb.qey:por-30-10.uclagsly	0.2494	0.5000	0.5080	0.4733	0.4167	0.2926	0.3088	0.6270
bm15.phb.qey:wpq-30-10.uclagsln	0.2318	0.5280	0.4920	0.4387	0.3753	0.2620	0.2744	0.6121
bm15.phb.qey:wpq-30-15.uclagsln	0.2271	0.5040	0.4860	0.4467	0.3700	0.2646	0.2668	0.6079
bm15.phb.qey:wpq-20-5.uclagsln	0.2266	0.5360	0.4820	0.4467	0.3707	0.2570	0.2699	0.6016
bm15.phb.qey:wpq-30-5.uclagsln	0.2173	0.5280	0.4680	0.4347	0.3660	0.2480	0.2651	0.5790

Table 2: **Ad hoc** results: Runs with and without query expansion for topics 101-150.
(Runs are presented in descending 'Recall' values.)

Run_name	Avg Prec	Prec[5]	Prec[10]	Prec[15]	Prec[30]	Prec[100]	R-Prec	Recall
uclaa1: auto.bm15.phb.qen.uclagsly	0.3345	0.5840	0.5380	0.4973	0.4333	0.3098	0.3629	0.8155
uclaa2: auto.bm15.phb.qey:wpq-10-10.uclagsly	0.2957	0.5440	0.5180	0.4920	0.4207	0.2760	0.3289	0.7786
uclaf1: fdbk.bm15.phb.qey:wpq-10-10.uclagsly	0.3090	0.5880	0.5220	0.4893	0.4360	0.2884	0.3459	0.7745

Table 3: Performace Averages over all Topics of the **Ad hoc** Runs with Query Expansion.
(Runs Named after the Algorithm used in the Expansion.)

Run_Name	Avg Prec	Prec[5]	Prec[10]	Prec[15]	Prec[30]	Prec[100]	R-Prec	Recall
r_lohi	0.3414	0.5880	0.5240	0.4947	0.4427	0.3152	0.3688	0.8290
r_hilo	0.3388	0.5880	0.5240	0.4960	0.4347	0.3160	0.3692	0.8333
emim	0.3176	0.5880	0.5240	0.4920	0.4433	0.2938	0.3554	0.7989
uclaf1: *wpq*	0.3087	0.5880	0.5240	0.4893	0.4360	0.2882	0.3460	0.7753
porter	0.2990	0.5880	0.5240	0.4893	0.4280	0.2798	0.3323	0.7457

Table 4: *Sign Test* Differences,
Precision at 15 Documents

	wpq	*porter*	*emim*	*r_lohi*	*r_hilo*
wpq	0	3	1	3	2
porter	4	0	4	3	1
emim	2	4	0	3	4
r_lohi	5	6	4	0	3
r_hilo	5	5	5	3	0

Table 5: *Sign Test* Probabilities,
Precision at 15 Documents

	wpq	*porter*	*emim*	*r_lohi*	*r_hilo*
wpq	1.0000				
porter	1.0000	1.0000			
emim	1.0000	1.0000	1.0000		
r_lohi	0.7266	0.5078	1.0000	1.0000	
r_hilo	0.4531	0.2188	1.0000	1.0000	1.0000

Table 6: *Sign Test* Differences,
Precision at 30 Documents

	wpq	*porter*	*emim*	*r_lohi*	*r_hilo*
wpq	0	18	12	13	15
porter	12	0	15	11	12
emim	14	18	0	16	19
r_lohi	18	18	18	0	15
r_hilo	13	18	16	7	0

Table 7: *Sign Test* Probabilities,
Precision at 30 Documents

	wpq	*porter*	*emim*	*r_lohi*	*r_hilo*
wpq	1.0000				
porter	0.3613	1.0000			
emim	0.8445	0.7277	1.0000		
r_lohi	0.4725	0.2652	0.8638	1.0000	
r_hilo	0.8501	0.3613	0.7353	0.1338	1.0000

Table 8: *Sign Test* Differences,
Precision at 100 Documents

	wpq	*porter*	*emim*	*r_lohi*	*r_hilo*
wpq	0	22	8	8	8
porter	15	0	15	8	8
emim	19	24	0	12	9
r_lohi	32	31	27	0	15
r_hilo	32	33	27	14	0

Table 9: *Sign Test* Probabilities,
Precision at 100 Documents

	wpq	*porter*	*emim*	*r_lohi*	*r_hilo*
wpq	1.0000				
porter	0.3239	1.0000			
emim	0.0543	0.2002	1.0000		
r_lohi	0.0003	0.0004	0.0250	1.0000	
r_hilo	0.0003	0.0002	0.0046	1.0000	1.0000

Table 10: *Sign Test* Differences,
Average Precision

	wpq	*porter*	*emim*	*r_lohi*	*r_hilo*
wpq	0	29	15	10	11
porter	20	0	17	9	10
emim	25	32	0	13	16
r_lohi	39	40	36	0	31
r_hilo	38	39	33	17	0

Table 11: *Sign Test* Probabilities,
Average Precision

	wpq	*porter*	*emim*	*r_lohi*	*r_hilo*
wpq	1.0000				
porter	0.2531	1.0000			
emim	0.1547	0.0455	1.0000		
r_lohi	0.0001	0.0000	0.0017	1.0000	
r_hilo	0.0002	0.0001	0.0223	0.0606	1.0000

Table 12: *Sign Test* Differences,
Recall

	wpq	*porter*	*emim*	*r_lohi*	*r_hilo*
wpq	0	24	8	10	7
porter	8	0	10	4	3
emim	15	25	0	9	6
r_lohi	27	28	25	0	14
r_hilo	29	32	26	14	0

Table 13: *Sign Test* Probabilities,
Recall

	wpq	*porter*	*emim*	*r_lohi*	*r_hilo*
wpq	1.0000				
porter	0.0080	1.0000			
emim	0.2100	0.0180	1.0000		
r_lohi	0.0085	0.0000	0.0101	1.0000	
r_hilo	0.0005	0.0000	0.0008	1.0000	1.0000

Table 14: *Sign Test* Differences,
Recall/Precision

	wpq	*porter*	*emim*	*r_lohi*	*r_hilo*
wpq	0	19	5	10	8
porter	11	0	7	6	7
emim	17	23	0	11	11
r_lohi	23	26	19	0	13
r_hilo	27	26	20	12	0

Table 15: *Sign Test* Probabilities,
Recall/Precision

	wpq	*porter*	*emim*	*r_lohi*	*r_hilo*
wpq	1.0000				
porter	0.2012	1.0000			
emim	0.0169	0.0062	1.0000		
r_lohi	0.0367	0.0008	0.2012	1.0000	
r_hilo	0.0023	0.0017	0.1508	1.0000	1.0000

Table 16: *t–test*,
Precision at 15 Documents

Runs	Mean Difference	SD Difference	*t*	Probability
r_hilo/r_lohi	0.0013	0.0285	0.3305	0.7424
r_hilo/emim	0.0040	0.0367	0.7712	0.4443
r_lohi/emim	0.0027	0.0300	0.6282	0.5328
r_hilo/porter	0.0067	0.0278	1.6973	0.0960
r_lohi/porter	0.0053	0.0325	1.1579	0.2525
emim/porter	0.0027	0.0380	0.4957	0.6224
r_hilo/wpq	0.0067	0.0337	1.3999	0.1678
r_lohi/wpq	0.0053	0.0352	1.0703	0.2897
emim/wpq	0.0027	0.0232	0.8137	0.4197
porter/wpq	0.0000	0.0404	0.0007	0.9994

Table 17: *t–test*,
Precision at 30 Documents

Runs	Mean Difference	SD Difference	*t*	Probability
r_hilo/r_lohi	-0.0080	0.0298	-1.8984	0.0635
r_hilo/emim	-0.0087	0.0583	-1.0521	0.2979
r_lohi/emim	-0.0007	0.0585	-0.0810	0.9358
r_hilo/porter	0.0067	0.0539	0.8748	0.3860
r_lohi/porter	0.0147	0.0518	2.0019	0.0508
emim/porter	0.0153	0.0694	1.5624	0.1246
r_hilo/wpq	-0.0013	0.0534	-0.1770	0.8602
r_lohi/wpq	0.0067	0.0530	0.8882	0.3788
emim/wpq	0.0073	0.0458	1.1312	0.2635
porter/wpq	-0.0080	0.0450	-1.2590	0.2140

Table 18: *t–test*,
Precision at 100 Documents

Runs	Mean Difference	SD Difference	*t*	Probability
r_hilo/r_lohi	0.0008	0.0267	0.2118	0.8332
r_hilo/emim	0.0222	0.0435	3.6060	0.0007
r_lohi/emim	0.0214	0.0469	3.2291	0.0022
r_hilo/porter	0.0362	0.0656	3.8991	0.0003
r_lohi/porter	0.0354	0.0648	3.8658	0.0003
emim/porter	0.0140	0.0535	1.8494	0.0704
r_hilo/wpq	0.0278	0.0557	3.5311	0.0009
r_lohi/wpq	0.0270	0.0600	3.1815	0.0025
emim/wpq	0.0056	0.0301	1.3150	0.1946
porter/wpq	-0.0084	0.0410	-1.4496	0.1536

Table 19: *t–test*,
Average Precision

Runs	Mean Difference	SD Difference	*t*	Probability
r_hilo/r_lohi	-0.0026	0.0153	-1.1935	0.2384
r_hilo/emim	0.0212	0.0421	3.5637	0.0008
r_lohi/emim	0.0238	0.0470	3.5786	0.0008
r_hilo/porter	0.0398	0.0615	4.5727	0.0000
r_lohi/porter	0.0424	0.0658	4.5547	0.0000
emim/porter	0.0186	0.0592	2.2172	0.0313
r_hilo/wpq	0.0301	0.0537	3.9615	0.0002
r_lohi/wpq	0.0327	0.0603	3.8291	0.0004
emim/wpq	0.0089	0.0335	1.8726	0.0671
porter/wpq	-0.0097	0.0358	-1.9107	0.0619

Table 20: *t–test*,
Recall

Runs	Mean Difference	SD Difference	*t*	Probability
r_hilo/r_lohi	0.0005	0.0312	0.1052	0.9166
r_hilo/emim	0.0363	0.0743	3.4503	0.0012
r_lohi/emim	0.0358	0.0819	3.0935	0.0033
r_hilo/porter	0.0731	0.1094	4.7267	0.0000
r_lohi/porter	0.0727	0.1108	4.6356	0.0000
emim/porter	0.0369	0.0965	2.7010	0.0095
r_hilo/wpq	0.0506	0.0835	4.2805	0.0001
r_lohi/wpq	0.0501	0.0908	3.9007	0.0003
emim/wpq	0.0143	0.0529	1.9106	0.0619
porter/wpq	-0.0226	0.0739	-2.1583	0.0358

Table 21: *t–test*,
Recall/Precision

Runs	Mean Difference	SD Difference	*t*	Probability
r_hilo/r_lohi	0.0003	0.0246	0.0978	0.9225
r_hilo/emim	0.0138	0.0450	2.1635	0.0354
r_lohi/emim	0.0134	0.0517	1.8396	0.0719
r_hilo/porter	0.0369	0.0636	4.1048	0.0002
r_lohi/porter	0.0366	0.0669	3.8626	0.0003
emim/porter	0.0231	0.0576	2.8398	0.0066
r_hilo/wpq	0.0231	0.0532	3.0729	0.0035
r_lohi/wpq	0.0228	0.0635	2.5401	0.0143
emim/wpq	0.0094	0.0313	2.1135	0.0397
porter/wpq	-0.0138	0.0441	-2.2100	0.0318

Acknowledgements

Thanks to Stephen Robertson and Stephen Walker for making OKAPI available to UCLA.

We are especially thankful to Stephen Walker for all his continuing help over and above that indicated in the text.

We wish to thank Donna Harman of NIST for her support during TREC–2.

This research was supported by a UCLA Academic Senate grant that also provided financial support for the graduate research assistantship of PVB.

Finally, ENE is grateful to SUN Microsystems Inc. for the equipment grant of the SPARC 2 that made this research possible.

References

Efthimiadis, E.N. (1993a) A user-centered evaluation of ranking algorithms for interactive query expansion. In: Korfhage R., Rasmussen E. and Willett P. (eds.) Proceedings of the 16th International Conference of the Association of Computing Machinery, Specialist Interest Group in Information Retrieval, June 1993. Pittsburgh, PA: ACM Press. pp. 146–159.

Efthimiadis, E.N. (in press) 'User choices: As the yardstick for the evaluation of ranking algorithms for interactive query expansion.' *Information Processing & Management.* In press.

Efthimiadis, E.N. 'Interactive Query Expansion: A User-based evaluation in a Relevance Feedback Environment.' Manuscript submitted for publication.

Harper, D.J. & van Rijsbergen, C.J. (1978) An evaluation of feedback in document retrieval using co-occurrence data. *Journal of Documentation, 34*(3), 189–216.

Porter, M.F. & Galpin, V. (1988) Relevance feedback in a public access catalogue for a research library: Muscat at the Scott Polar Research Institute. *Program, 22*(1), 1–20.

Robertson, S.E. (1990) On term selection for query expansion. *Journal of Documentation, 46*(4), 359–364.

Robertson, S.E. and Sparck Jones, K. (1976) Relevance weighting of search terms. *Journal of the American Society for Information Science, 27*(3), 129–146.

Robertson, S.E., Walker, S., Hancock-Beaulieu, M., Gull, A. & Lau, M. (1993a) Okapi at TREC. In: Harman, D. K. (Ed.) *The First Text Retrieval Conference (TREC–1).* Proceedings. NIST Special Publication 500-207. Gaithersburg, MD: NIST, 1993. pp21–30.

Robertson, S.E., Walker, S., Jones, S., Hancock-Beaulieu, M., & Gatford, M. (1993b) Okapi at TREC–2. In: Harman, D.K. (Ed.) *Text Retrieval Conference (TREC–2).* Gaithersburg, MD: NIST, August 30-September 1, 1993, Proceedings, to appear.

van Rijsbergen, C.J. (1977) A theoretical basis for the use of co-occurrence data in information retrieval. *Journal of Documentation, 33*, 106–119.

van Rijsbergen, C.J., Harper, D.J. & Porter, M.F. (1981) The selection of good search terms. *Information Processing and Management, 17*(2), 77–91.

A Runs

The runs presented here are grouped by algorithm used for the expansion. The list starts with runs that did not include query expansion.

bm15.phn.qen.uclagsln
bm15.phn.qen.uclagsly
bm15.phy.qen.uclagsly
bm15.phb.qen.uclagsly

bm15.phb.qey:wpq-10-5.uclagsln
bm15.phb.qey:wpq-10-5.uclagsly
bm15.phb.qey:wpq-10-10.uclagsln
bm15.phb.qey:wpq-10-10.uclagsly
bm15.phb.qey:wpq-10-15.uclagsln
bm15.phb.qey:wpq-20-5.uclagsln
bm15.phb.qey:wpq-20-10.uclagsln
bm15.phb.qey:wpq-20-15.uclagsln
bm15.phb.qey:wpq-30-5.uclagsln
bm15.phb.qey:wpq-30-10.uclagsln
bm15.phb.qey:wpq-30-15.uclagsln

bm15.phb.qey:emim-10-10.uclagsln
bm15.phb.qey:emim-10-10.uclagsly
bm15.phb.qey:emim-20-10.uclagsly
bm15.phb.qey:emim-30-10.uclagsly

bm15.phb.qey:port-10-10.uclagsln
bm15.phb.qey:port-10-10.uclagsly
bm15.phb.qey:port-20-10.uclagsly
bm15.phb.qey:port-30-10.uclagsly

bm15.phb.qey:r_hilo-10-10.uclagsln
bm15.phb.qey:r_hilo-10-10.uclagsly

bm15.phb.qey:r_lohi-10-5.uclagsln
bm15.phb.qey:r_lohi-10-10.uclagsln
bm15.phb.qey:r_lohi-10-10.uclagsly
bm15.phb.qey:r_lohi-10-15.uclagsln
bm15.phb.qey:r_lohi-20-5.uclagsln
bm15.phb.qey:r_lohi-20-10.uclagsln
bm15.phb.qey:r_lohi-20-15.uclagsln
bm15.phb.qey:r_lohi-30-5.uclagsln
bm15.phb.qey:r_lohi-30-10.uclagsln
bm15.phb.qey:r_lohi-30-15.uclagsln

Incorporating Semantics Within a Connectionist Model and a Vector Processing Model

Richard Boyd, James Driscoll, Inien Syu

Department of Computer Science
University of Central Florida
Orlando, Florida 32816
(407) 823-2341
FAX: (407) 823-5419
e-mail: driscoll@cs.ucf.edu

Abstract

Semantic information obtained from the public domain 1911 version of Roget's Thesaurus is combined with keywords to measure similarity between natural language topics and documents. Two approaches are explored. In one approach, a combination of keyword relevance and semantic relevance is achieved by using the vector processing model for calculating similarity, but extending the use of a keyword weight by using individual weights for each of its meanings. This approach is based on the database concept of semantic modeling and the linguistic concept of thematic roles. It is applicable to both routing and archival retrieval. The second approach is especially suited for routing. It is based on an AI connectionist model. In this approach, a probabilistic inference network is modified using semantic information to achieve a competitive activation mechanism that can be used for calculating similarity.

Keywords: vector processing model, semantic data model, semantic lexicon, inference network, connectionist model.

1. Introduction

The experiments reported here use a relatively efficient method to detect the semantic representation of text. Our original method is based on semantic modeling and is described in [4,17,19].

Semantic modeling was an object of considerable database research in the late 1970's and early 1980's. A brief overview can be found in [3]. Essentially, the semantic modeling approach identified concepts useful in talking informally about the real world. These concepts included the two notions of entities (objects in the real world) and relationships among entities (actions in the real world). Both entities and relationships have properties.

The properties of entities are often called attributes. There are basic or surface level attributes for entities in the real world. Examples of surface level entity attributes are General Dimensions, Color, and Position. These properties are prevalent in natural language. For example, consider the phrase "large, black book on the table" which indicates the General Dimensions, Color, and Position of the book.

In linguistic research, the basic properties of relationships are discussed and called thematic roles. Thematic roles are also referred to in the literature as participant roles, semantic roles and case roles. Examples of thematic roles are Beneficiary and Time. Thematic roles are prevalent in natural language; they reveal how sentence phrases and clauses are semantically related to the verbs in a sentence. For example, consider the phrase "purchase for Mary on Wednesday" which indicates who benefited from a purchase (Beneficiary) and when a purchase occurred (Time).

A main goal of our research has been to detect thematic information along with attribute information contained in natural language queries and documents. In order to use this additional information, the concept of text relevance needs to be modified.

In [17,19] the major modifications included the addition of a lexicon with thematic and attribute information, and a modified computation of a vector processing similarity coefficient. That research concerned a Question/Answer environment where queries were the length of a sentence and documents were either a sentence or at most a paragraph. At that time, our lexicon was based on 36 semantic categories, and in that environment, our semantic approach produced a significant improvement in retrieval performance.

However, for TREC-1 [4], document and topic length presented a problem and caused our semantic approach based on 36 semantic categories to be of little value. However, as reported in [4], by breaking the TREC documents into paragraphs, a significant improvement was demonstrated.

This work has been supported in part by NASA KSC Cooperative Agreement NCC 10-003 Project 2, Florida High Technology and Industry Council Grants 4940-11-28-721 and 4940-11-28-728.

In Section 2, we describe our original semantic lexicon and an extension which uses a larger number of semantic categories. Section 3 presents an application of an AI connectionist model to the task of routing. Section 4 presents an approach different than reported in TREC-1 [4], using our extended semantic lexicon within the vector processing model. Section 5 summarizes our research effort.

2. The Semantic Lexicon

Our semantic approach uses a thesaurus as a source of semantic categories (thematic and attribute information). For example, Roget's Thesaurus contains a hierarchy of word classes to relate word senses [14]. In TREC-1 [4] and in earlier research [17,19], we selected several classes from this hierarchy to be used for semantic categories. We defined thirty-six semantic categories as shown in Figure 1.

In order to explain the assignment of semantic categories to a given term using Roget's Thesaurus, consider the brief index quotation for the term "vapor":

vapor		
n.	fog	404.2
	fume	401
	illusion	519.1
	spirit	4.3
	steam	328.10
	thing imagined	535.3
v.	be bombastic	601.6
	bluster	911.3
	boast	910.6
	exhale	310.23
	talk nonsense	547.5

The eleven different meanings of the term "vapor" are given in terms of a numerical category. We developed a mapping of the numerical categories in Roget's Thesaurus to the thematic role and attribute categories given in Figure 1. In this example, "fog" and "fume" correspond to the attribute State; "steam" maps to the attribute Temperature; and "exhale" is a trigger for the attribute Motion with Reference to Direction. The remaining seven meanings associated with "vapor" do not trigger any thematic roles or attributes. Since there are eleven meanings associated with "vapor," we indicated in the lexicon a probability of 1/11 each time a category is triggered. Hence, a probability of 2/11 is assigned to State, 1/11 to Temperature, and 1/11 to Motion with Reference to Direction. This technique of calculating probabilities is being used as a simple alternative to a corpus analysis.

It should be pointed out that we are still experimenting with other ways of calculating probabilities. For example, as in [8], a probabilistic part-of-speech tagger could be used to further restrict the different meanings of a term, and existing lexical sources could be used to obtain an ordering based on frequency of use for the different meanings of a term.

As reported in [4], the use of 36 semantic categories caused problems when dealing with TREC documents. When the size of a document is large, a greater number of the 36 semantic categories are triggered in the document. Also, when using the semantic approach described in [19] the probability present for each category in a document is often very close to one. Consequently, almost every one of the

Thematic Role Categories	
TACM	Accompaniment
TAMT	Amount
TBNF	Beneficiary
TCSE	Cause
TCND	Condition
TCMP	Comparison
TCNV	Conveyance
TDGR	Degree
TDST	Destination
TDUR	Duration
TGOL	Goal
TINS	Instrument
TSPL	Location/Space
TMAN	Manner
TMNS	Means
TPUR	Purpose
TRNG	Range
TRES	Result
TSRC	Source
TTIM	Time

Attribute Categories	
ACOL	Color
AEID	External and Internal Dimensions
AFRM	Form
AGND	Gender
AGDM	General Dimensions
ALDM	Linear Dimensions
AMFR	Motion Conjoined with Force
AGMT	Motion in General
AMDR	Motion with Reference to Direction
AORD	Order
APHP	Physical Properties
APOS	Position
ASTE	State
ATMP	Temperature
AUSE	Use
AVAR	Variation

Figure 1. Thirty-Six Semantic Categories.

36 semantic categories becomes present in every document. This causes semantic category weights to become very low and useless within that approach.

As reported in [4], one way to solve this problem is to break TREC documents into paragraphs. But, another way to solve the problem of long documents causing semantic weights to be of little value is to have more semantic categories. A large number of "semantic" categories can be obtained (for example) by using all the categories and/or subcategories found in Roget's Thesaurus, instead of the 36 semantic categories we have used. This may be a deviation from database semantic modeling. In any case, it needs to be examined.

Consequently, for the experiments reported here, a semantic lexicon was created based on all the word senses found in the public domain 1911 version of Roget's Thesaurus. To provide an example, consider Topic 052 as shown in Figure 2. Figure 3 indicates the keywords and frequency information within Topic 052, along with the semantic categories obtained from our extended lexicon for those keywords. Note that stemming was not used for the processing of Topic 052; so, some keywords in Topic 052 were not located in our lexicon (e.g. sanctions).

The categories recorded in our extended semantic lexicon use the category numbers found in the 1911 version of Roget's Thesaurus. These numbers are then followed by a part-of-speech code also found in the 1911 version of Roget's Thesaurus. The number after the part-of-speech code represents a sub-category, but this number does not appear in the 1911 version of Roget's Thesaurus. That number was created based on groupings of words within the thesaurus.

3. Connectionist Model Routing Experiments

Recent work suggests that significant improvements in retrieval performance will require a technique that, in some sense, "understands" the content of documents and queries and can be used to infer probable relationships between documents and queries [2]. In this view, information retrieval is an inference or evidential reasoning process in which we estimate the probability that a user's information need is met given a document as "evidence". The techniques required to support this kind of inference are similar to those used in expert systems that must reason with uncertain information. Several probabilistically-oriented inference network models have been developed using experimental document collections [5] during the past few years for information retrieval [15]. These models are generally characterized by an architecture with two layers corresponding to documents and index terms. The documents and index terms are connected by direct links. Initially, the prior probabilities of all root nodes (nodes with no predecessors) and the conditional probabilities of all non-root nodes (given all possible combinations of their direct predecessors) must be specified. A retrieval consists of one or more documents with the highest posterior probability for the given set of index terms (evidences) which represent a user's information need.

Over the last few years, the technique of automated inference using probabilistic inference networks has become popular within the AI probability and uncertainty community, particular in the context of expert systems [6,7]. The most

<top>

<head> Tipster Topic Description

<num> Number: 052

<dom> Domain: International Economics

<title> Topic: South African Sanctions

<desc> Description:

Document discusses sanctions against South Africa.

<narr> Narrative:

A relevant document will discuss any aspect of South African sanctions, such as: sanctions declared/proposed by a country against the South African government in response to its apartheid policy, or in response to pressure by an individual, organization or another country; international sanctions against Pretoria imposed by the United Nations; the effects of sanctions against S. Africa; opposition to sanctions; or, compliance with sanctions by a company. The document will identify the sanctions instituted or being considered, e.g., corporate disinvestment, trade ban, academic boycott, arms embargo.

<con> Concept(s):

1. sanctions, international sanctions, economic sanctions
2. corporate exodus, corporate disinvestment, stock divestiture, ban on new investment, trade ban, import ban on South African diamonds, U.N. arms embargo, curtailment of defense contracts, cutoff of nonmilitary goods, academic boycott, reduction of cultural ties
3. apartheid, white domination, racism
4. antiapartheid, black majority rule
5. Pretoria

<fac> Factor(s):

<nat> Nationality: South Africa

</fac>

<def> Definition(s):

</top>

Figure 2. Topic 052.

important constraint on the use of a probabilistic network is the fact that in general, the computation of the exact posterior probabilities is NP-hard [1]. Thus it is unlikely that we could develop an efficient general-purpose algorithm which would work well for all kinds of inference networks. There are several alternatives, such as the use of approximation algorithms or heuristic algorithms, and creating special case algorithms [9,10].

The experiments here concern an attempt at a heuristic probabilistic inference network approach based on an AI connectionist model. The connectionist model uses a competitive activation rule to find the most probable retrieval. The term competitive activation rule refers to a spreading activation method in which nodes actively compete for available activation in a network. An initial formulation of a competitive activation mechanism was previously studied on three two-layer, abstract networks for diagnostic problem solving [11,13]. The connectionist model proposed here consists of a two-layer network architecture. Document nodes and index term nodes corresponding to each layer are connected by links whose weights represent association strengths between nodes. These links are also viewed as channels for sending information between nodes. Figure 4 is a simple network consisting of two document nodes and three index term nodes. At each moment of time, each node

Topic 52

curtailment	001	201n.1 38n.2
individual	001	372a.0 372n.2 549a.0 79a.0 87a.0 87n.0
considered	001	611d.0
compliance	001	602n.3 743n.0 762n.0 772n.0
reduction	001	103n.1 144n.0 195n.0 201n.1 308n.1 813n.0 85n.3
economics	002	692n.3
corporate	003	43a.0
response	002	462n.0 586v.3 714n.0 821n.0 888n.2 990n.3
relevant	001	23a.1 476a.2 9v.2
majority	001	102n.1 131n.0 33n.0
academic	002	514a.0 537a.0 542a.0
effects	001	780n.10 780n.5 798n.0
defense	001	717n.0 937n.2
country	002	181n.0 189n.1 189n.8 189n.9 266v.1 344n.0 371a.1
company	001	599n.14 712n.2 726n.8 72n.2 88n.1 892n.4
policy	001	626n.2 692n.2
aspect	001	183n.0 448n.3 7n.0
white	001	429a.1 430a.0 441n.5 996n.5
stock	001	11n.3 153n.4 166n.2 225n.13 25n.3 265a.0 31n.1 501n.2 613a.0 635n.0 636n.0 637v.0 637v.2 780n.10 798n.0 800n.0 811v.0
goods	001	780n.10 798n.0
black	001	349n.2 421a.0 431a.0 431n.3 432n.1 752n.0 945a.2
rule	001	136n.1 200n.1 466n.2 613n.5 693v.1 697n.1 737n.1 737v.2 737v.3 749v.2 80n.0 82n.3 963n.1
arms	002	459d.9 719v.2 722n.0 727n.0 877n.3 894v.3
international	003	12a.0 892a.4
organization	001	161n.0 329n.0 357n.0 357n.1 60n.0
opposition	001	179n.0 237n.0 708n.0 710n.0 719n.0 720n.0
investment	001	225n.0 716n.2 784n.0 809n.3
government	001	692n.3 693n.0 699n.3 737n.1 737n.5
domination	001	175n.0 737n.1
pressure	001	157n.2 175n.0 319n.0 642n.3 735n.1
identify	001	13v.0 464v.1 480av.5
document	003	467n.3 551n.2
embargo	001	265n.2 761n.0
discuss	001	298v.0 451v.0 460v.3 476v.0
boycott	002	297v.2
another	001	104a.1 15a.1 709v.0 714v.0
against	004	14a.0 179v.0 237d.0 276v.0 673d.0 704d.0 708d.0 708d.1 708v.0 708v.2 713v.2 716v.1 716v.5 717a.0 719v.0 764v.0 898a.0 932v.8
united	001	46a.1 714a.0
import	001	228v.1 296v.0 300v.0 516n.0 516v.0 642n.1 642v.0
exodus	001	293n.0 295v.1
trade	002	625n.4 734v.1 794n.1 794v.1
south	006	278n.1
being	001	1n.0 3n.0 831n.0 976n.2
will	002	360v.1 600n.0 600v.0 602d.0 604a.0 604n.0 737v.3 771n.11 784n.4
new	001	123a.0 146v.0 18a.0 614a.0 66n.0
ban	004	761n.0 908n.0 98n.3
any	001	25a.0 51n.0 609an.1

Figure 3: Word Frequency and Semantic Categories for Topic 052.

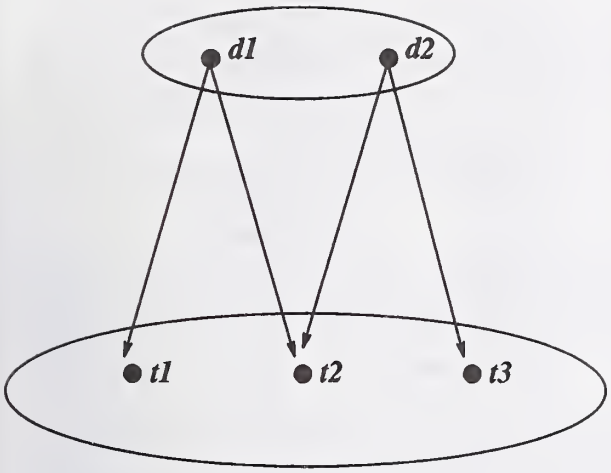


Figure 4: A Simple Network Consisting of Two Document Nodes and Three Index Term Nodes.

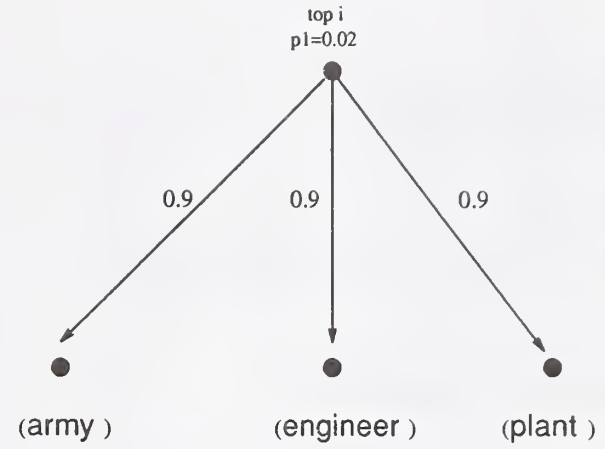


Figure 5: A Sample Network of the Experimental Model.

receives information about the activation levels of its immediate neighboring nodes (nodes connected to it via direct links), and then uses this information to calculate its own activation level. Through this process of spreading activation, the network settles down to equilibrium representing a retrieval to a user's information need.

The computation of the information retrieval inference process is based on a formalization of the causal and probabilistic associative knowledge underlying diagnostic problem-solving [18]. We do not discuss the formulation architecture and activation mechanism of the connectionist model. This information can be found in [11,13,16,18]. For TREC-2, we managed to complete only one official routing experiment for this approach, and it did not involve semantics. The experiment was intended to be a baseline experiment for our semantic experiments.

For TREC-2, a specific network was constructed for 50 topics. A list of index terms was assembled based on keywords in the concept section of each topic. In this network, each output node represented a topic, and each input node represented a keyword. The prior probability assigned to each topic node was equal to 1/(total number of topics). The connection strengths were assigned equal weights (0.9).

The network contained 50 topic nodes and 848 index term nodes. These nodes were connected via 1449 links. An example of this network is shown in Figure 5, where p_i is the prior probability of topic top_i. The keywords "army", "engineer", and "plant" were obtained by processing the concept section of topic top_i. Currently, the network is enhanced by using an estimated weighting scheme.

We performed a Category B routing experiment. Using just keywords, the results were not good. The main problem was due to the fact that, in the document ranking, many documents had the same score used to generate the ranking. In order to satisfy the requirements for the ranking, we had to artificially rank those documents with the same score. This was done based on order of appearance. The performance was terrible except for Topic 66. This topic had only two known relevant documents for Category B routing experiments and our inference network retrieved one of them in the top 20 documents! No further connectionist model experiments have been completed. We were unable to modify the baseline keyword experiment or perform semantic experiments for this approach.

4. Vector Processing Model Experiments

In this section, we explain the manner in which semantics is incorporated within a vector processing model using the semantic lexicon explained in Section 2. Please note that an entry in our semantic lexicon has the form of a word followed by codes for each of the semantic categories the word triggers. We explain our approach using a text relevance determination procedure intended to show what is being calculated rather than show the actual computations for the approach. The procedure presented here generates several outputs that are really not necessary, but are included just to help explain the approach. The relevance determination procedure is explained using the four documents and query shown in Figure 6. A few preliminary computations are reviewed in order to explain the procedure.

First, the number of documents each word is in must be determined. Figure 7 shows a list of words from the four documents and the query of Figure 6 along with the number of documents each word is in (*df*).

Next, the inverse document frequency (*idf*) of each word is determined by the equation $\log_{10}(N/df)$, where $N = 4$, the total number of documents. Figure 8 provides the *idf* of each word. Sometimes, the *idf* of a word is undefined. This can happen when a word does not occur in the documents but does occur in a query. For example, the words "depart", "do" and "when" do not appear in the four documents. Thus, the *idf* of these terms cannot be defined here. Later, we will see that an adjustment can be made for these undefined values.

Next, the category probability of each query word is determined. Figure 9 shows an alphabetized list of all the unique words from the query, the frequency of each word in the query, and the semantic categories each word triggers.

Document #1

Locomotives pull the trains.

Document #2

People meet people under the canopy and within trains.

Document #3

Trains carry freight from the station.

Document #4

Trains leave the station hourly until noon.

Query

When do trains depart the station?

Figure 6. Four Documents and a Query.

word	number of documents the word is in (*df*)
and	1
canopy	1
carry	1
do	0
depart	0
freight	1
from	1
hourly	1
leave	1
locomotives	1
meet	1
noon	1
people	1
pull	1
station	2
the	4
trains	4
under	1
until	1
when	0
within	1

Figure 7. List of Words in the Documents and Query.

word	*idf* of the word $\log_{10} \frac{N}{df}$
and	.6
canopy	.6
carry	.6
depart	undefined
do	undefined
freight	.6
from	.6
hourly	.6
leave	.6
locomotives	.6
meet	.6
noon	.6
people	.6
pull	.6
station	.3
the	0
trains	0
under	.6
until	.6
when	undefined
within	.6

Figure 8. The *idf* of Each Word.

word	frequency	category	probability
depart	1	AMDR	1/4
		TAMT	1/8
do	1	AUSE	1/21
		ATMP	1/21
		TCSE	1/21
		TCNV	2/21
		TRES	1/21
		TSRC	1/21
station	1	APOS	3/16
		AORD	1/8
		TAMT	1/16
		TCND	1/8
		TDGR	1/16
		TSPL	3/16
the	1	----	----
trains	1	AORD	7/24
		AMDR	1/12
		AMFR	1/12
		TACM	1/24
		TCNV	1/12
when	1	TAMT	1/3
		TTIM	2/3

Figure 9. Words in the Query.

The semantic categories in our example are those shown in Figure 1. For example, consider the word "depart" which occurs one time in the query as shown in Figure 9. The semantic lexicon entry for the word "depart" using the categories of Figure 1 is as follows:

depart: NONE NONE NONE NONE NONE AMDR AMDR TAMT

where NONE represents a word sense not included in the 36 semantic categories of Figure 1. If a uniform distribution is assumed, then AMDR is triggered 1/4 of the time and TAMT is triggered 1/8 of the time. This is shown in Figure 9 as the probabilities for each semantic category.

A similar category probability determination is done for each document. Figure 10 is an alphabetized list of all the unique words in Document #4 of Figure 6. The semantic categories each word triggers along with probabilities are also shown.

The text relevance determination procedure is shown in Figure 11. The procedure uses three input lists:

a. List of words and the *idf* of each word, as shown in Figure 8.

b. List of words in the query and the semantic categories they trigger along with the probability of triggering those categories, as shown in Figure 9.

c. List of words in a document and the semantic categories they trigger along with the probability of triggering those categories, as shown in Figure 10.

The procedure operates as follows:

Step 1.

This step determines the common meanings between the query and the document. Figure 12 corresponds to the output of Step 1 for Document #4. In Step 1, a new list is created as follows:

For each word in the query, follow either subsection (a) or (b), whichever applies:

a. For each category the word triggers, find each word in the document that triggers the category and output three things:

   1) The word in the query and its frequency of occurrence.
   2) The word in the document and its frequency of occurrence.
   3) The category.

b. If the word does not trigger a category, then look for the word in the document and if found, output two things and a "----":

   1) The word in the query and its frequency of occurrence.
   2) The word in the document and its frequency of occurrence.

word	frequency	category	probability
hourly	1	TTIM	1.0
leave	1	AMDR	1/7
		TAMT	1/7
noon	1	ALDM	1/3
		TTIM	2/3
the	1	----	---
station	1	APOS	3/16
		AORD	1/8
		TAMT	1/16
		TCNP	1/8
		TDGR	1/16
		TSPL	3/16
trains	1	AORD	7/24
		AMDR	1/12
		AMFR	1/12
		TACM	1/24
		TCNV	1/12
until	1	TTIM	1.0

Figure 10. Words in Document #4.

Step 1 - Refer to Figure 12.
Determine common meaning between query and the document.

Step 2 - Refer to Figure 13.
Adjust for words in the query that are not in any of the documents.

Step 3 - Refer to Figure 14.
Calculate the weight of a semantic component in the query and calculate the weight of a semantic component in the document.

Step 4 - Refer to Figure 15.
Multiply the weight in the query by the weight in the document.

Step 5 - Refer to Figure 15.
Sum all the individual products of Step 4 into a single value which is the semantic similarity coefficient.

Figure 11. Relevance Determination Procedure to Explain Semantic Similarity.

First List

Item Number	First Entry Word & Frequency in Query	Second Entry Word & Frequency in Document #4	Third Entry Category
1	(depart,1)	(leave,1)	AMDR
2	(depart,1)	(trains,1)	AMDR
3	(depart,1)	(leave,1)	TAMT
4	(depart,1)	(station,1)	TAMT
5	(do,1)	(trains,1)	TCNV
6	(station,1)	(station,1)	APOS
7	(station,1)	(station,1)	AORD
8	(station,1)	(trains,1)	AORD
9	(station,1)	(leave,1)	TAMT
10	(station,1)	(station,1)	TAMT
11	(station,1)	(station,1)	TCND
12	(station,1)	(station,1)	TDGR
13	(station,1)	(station,1)	TSPL
14	(the,1)	(the,1)	----
15	(trains,1)	(trains,1)	AORD
16	(trains,1)	(leave,1)	AMDR
17	(trains,1)	(trains,1)	AMDR
18	(trains,1)	(trains,1)	AMFR
19	(trains,1)	(trains,1)	TACM
20	(trains,1)	(trains,1)	TCNV
21	(when,1)	(leave,1)	TAMT
22	(when,1)	(hourly,1)	TTIM
23	(when,1)	(noon,1)	TTIM
24	(when,1)	(until,1)	TTIM

Figure 12. Common Meaning.

Second List

Item Number	First Entry Word & Frequency in Query	Second Entry Word & Frequency in Document #4	Third Entry Category
1	(leave,1)	(leave,1)	AMDR
2	(trains,1)	(trains,1)	AMDR
3	(leave,1)	(leave,1)	TAMT
4	(station,1)	(station,1)	TAMT
5	(trains,1)	(trains,1)	TCNV
6	(station,1)	(station,1)	APOS
7	(station,1)	(station,1)	AORD
8	(station,1)	(trains,1)	AORD
9	(station,1)	(leave,1)	TAMT
10	(station,1)	(station,1)	TAMT
11	(station,1)	(station,1)	TCND
12	(station,1)	(station,1)	TDGR
13	(station,1)	(station,1)	TSPL
14	(the,1)	(the,1)	----
15	(trains,1)	(trains,1)	AORD
16	(trains,1)	(leave,1)	AMDR
17	(trains,1)	(trains,1)	AMDR
18	(trains,1)	(trains,1)	AMFR
19	(trains,1)	(trains,1)	TACM
20	(trains,1)	(trains,1)	TCNV
21	(leave,1)	(leave,1)	TAMT
22	(hourly,1)	(hourly,1)	TTIM
23	(noon,1)	(noon,1)	TTIM
24	(until,1)	(until,1)	TTIM

Figure 13. Adjustment for Words with no *idf*.

Considering Figure 12, the word "depart" occurs in the query one time and triggers the category AMDR. The word "leave" occurs in Document #4 once and also triggers the category AMDR. Thus, item 1 in Figure 12 corresponds to subsection (a) as described above. An example using subsection (b) occurs in item 14 of Figure 12.

Step 2.

This step adjusts for words in the query that are not in any of the documents. Figure 13 shows the output of Step 2 for Document #4. In this step, another list is created from the list created in Step 1. For each item in the Step 1 list which has a word with undefined *idf*, this step replaces the word in the First Entry column by the word in the Second Entry column. For example, the word "depart" has an undefined *idf* as shown in Figure 8. Thus, the word "depart" in item 1 of Figure 12 should be replaced by the word "leave" from the Second Entry column. This is shown in item 1 of Figure 13. Likewise, the words "do" and "when" also have an undefined *idf* and are respectively replaced by the words from the Second Entry column.

Step 3.

This step calculates the weight of a semantic component in the query and calculates the weight of a semantic component in the document. Figure 14 shows the output of Step 3 for Document #4. In Step 3, another list is created from the list created in Step 2 as follows:

For each item in the Step 2 list, follow either subsection (a) or (b), whichever applies:

a. If the Third Entry specifies a category, then

   1) Replace the First Entry by computing:

$$\begin{pmatrix} idf \text{ of} \\ \text{word in} \\ \text{First Entry} \end{pmatrix} * \begin{pmatrix} \text{frequency of} \\ \text{word in} \\ \text{First Entry} \end{pmatrix} * \begin{pmatrix} \text{probability the word} \\ \text{triggers the category} \\ \text{in the Third Entry} \end{pmatrix}$$

   2) Replace the Second Entry by computing:

$$\begin{pmatrix} idf \text{ of} \\ \text{word in} \\ \text{Second Entry} \end{pmatrix} * \begin{pmatrix} \text{frequency of} \\ \text{word in} \\ \text{Second Entry} \end{pmatrix} * \begin{pmatrix} \text{probability the word} \\ \text{triggers the category} \\ \text{in the Third Entry} \end{pmatrix}$$

   3) Omit the Third Entry.

b. If the Third Entry does not specify a category, then

   1) Replace the First Entry by computing:

$$\begin{pmatrix} idf \text{ of} \\ \text{word in} \\ \text{First Entry} \end{pmatrix} * \begin{pmatrix} \text{frequency of} \\ \text{word in} \\ \text{First Entry} \end{pmatrix}$$

   2) Replace the Second Entry by computing:

$$\begin{pmatrix} idf \text{ of} \\ \text{word in} \\ \text{Second Entry} \end{pmatrix} * \begin{pmatrix} \text{frequency of} \\ \text{word in} \\ \text{Second Entry} \end{pmatrix}$$

   3) Omit the Third Entry.

In Figure 14, item 1 is an example of using subsection (a), and item 14 is an example of using subsection (b).

Step 4.

This step multiplies the weights in the query by the weights in the document. The top portion of Figure 15 shows the output of Step 4. In the list created here, the numerical value created in the First Entry column of Figure 14 is multiplied by the numerical value created in the Second Entry column of Figure 14.

Step 5.

This step sums the values in the Step 4 list to compute the semantic similarity coefficient for a particular document. The bottom portion of Figure 15 shows the output of step 5 for Document #4.

We have finally observed an improved Precision/Recall performance using the semantic similarity coefficient explained here. For example, in a Category B filtering experiment where the words being considered were only those in the topics and *idf* values were determined by the number of topics a word is in, we have observed the keyword and semantic results shown in Figure 16 and Figure 17, respectively. The 11-pt average for these two experiments reveals a 23% increase due to the use of semantic categories. According to Sparck Jones' criteria, this change would be classified as "significant" (greater than 10.0%) [12]. We believe further improvement is possible by considering more words, stemming for plurals and tenses of words, better *idf* values (like those used for archival retrieval), a modern lexicon, and a focus on paragraphs instead of whole documents.

5. Summary

Our progress during TREC-1 and TREC-2 has been the following:

a. We created efficient code for a UNIX platform. Originally our code used B+ tree structures for implementing inverted files on a DOS platform. We now use hashing to replace B+trees, establishing codes to replace character strings; and the UNIX platform provides faster processing than the DOS platform.

b. We built an index for a semantic lexicon based on the public domain 1911 version of Roget's Thesaurus. To do this, we had to create our own category numbering system similar to today's version of Roget's Thesaurus.

c. We solved part of the blend problem for semantic and keyword weights. We now base semantic category weights on the *idf* of words which generate the semantic categories.

We can now index or scan TREC documents at rates faster than 60 Megabytes per hour depending on the workstation. We have a semantic lexicon of approximately 20,000 words with flexible category codes that allow a course (36 categories) through fine (more than 15,000 categories) semantic analysis. As shown in Section 4, our procedure for determining relevance is based on the senses of each word. For example, using the vector processing model and the similarity coefficient

$$sim(Q, D_i) = \sum_{j=1}^{t} w_{qj} \cdot d_{ij},$$

Third List

Item Number	First Entry	Second Entry
1	.6 * 1 * 1/7 = .0857	.6 * 1 * 1/7 = .0857
2	0 * 1 * 1/12 = 0	0 * 1 * 1/12 = 0
3	.6 * 1 * 1/7 = .0857	.6 * 1 * 1/7 = .0857
4	.3 * 1 * 1/16 = .0188	.3 * 1 * 1/16 = .0188
5	0 * 1 * 1/12 = 0	0 * 1 * 1/12 = 0
6	.3 * 1 * 3/16 = .0563	.3 * 1 * 3/16 = .0563
7	.3 * 1 * 7/24 = .0875	.3 * 1 * 7/24 = .0875
8	.3 * 1 * 1/8 = .0375	0 * 1 * 7/24 = 0
9	.3 * 1 * 1/16 = .0188	.6 * 1 * 1/7 = .0857
10	.3 * 1 * 1/16 = .0188	.3 * 1 * 1/16 = .0188
11	.3 * 1 * 1/8 = .0375	.3 * 1 * 1/8 = .0375
12	.3 * 1 * 1/16 = .0188	.3 * 1 * 1/16 = .0188
13	.3 * 1 * 3/16 = .0563	.3 * 1 * 3/16 = .0563
14	0 * 1 = 0	0 * 1 = 0
15	0 * 1 * 7/24 = 0	0 * 1 * 7/24 = 0
16	0 * 1 * 1/12 = 0	.6 * 1 * 1/7 = .0857
17	0 * 1 * 1/12 = 0	0 * 1 * 1/12 = 0
18	0 * 1 * 1/12 = 0	0 * 1 * 1/12 = 0
19	0 * 1 * 1/24 = 0	0 * 1 * 1/24 = 0
20	0 * 1 * 1/12 = 0	0 * 1 * 1/12 = 0
21	.6 * 1 * 1/7 = .0857	.6 * 1 * 1/7 = .0857
22	.6 * 1 * 1.0 = .6000	.6 * 1 * 1.0 = .6000
23	.6 * 1 * 2/3 = .4000	.6 * 1 * 2/3 = .4000
24	.6 * 1 * 1.0 = .6000	.6 * 1 * 1.0 = .6000

Figure 14. Weights of Semantic Components.

Fourth List

Item Number	Value
1	.00734
2	0
3	.00734
4	.00035
5	0
6	.00317
7	.00734
8	0
9	.00170
10	.00035
11	.00141
12	.00035
13	.00317
14	0
15	0
16	0
17	0
18	0
19	0
20	0
21	.00734
22	.36000
23	.16000
24	.36000

Sum of all values in Fourth List
0.91986

Figure 15. Multiplied Weights and Their Sum.

Queryid (Num):	47 of 50
Total number of documents over all queries	
Retrieved:	36610
Relevant:	2064
Rel_ret:	913
Interpolated Recall - Precision Averages	
at 0.00	0.3514
at 0.10	0.1968
at 0.20	0.1367
at 0.30	0.1082
at 0.40	0.0894
at 0.50	0.0752
at 0.60	0.0276
at 0.70	0.0105
at 0.80	0.0062
at 0.90	0.0013
at 1.00	0.0007
Average precision (non-interpolated) over all rel docs	
	0.0746
Precision:	
At 5 docs:	0.1660
At 10 docs:	0.1532
At 15 docs:	0.1433
At 20 docs:	0.1298
At 30 docs:	0.1057
At 100 docs:	0.0643
At 200 docs:	0.0465
At 500 docs:	0.0302
At 1000 docs:	0.0194
R-Precision (precision after R (= num_rel for a query) docs retrieved):	
Exact:	0.1035

Figure 16. Filtering Using Keywords.

Queryid (Num):	47 of 50
Total number of documents over all queries	
Retrieved:	36383
Relevant:	2064
Rel_ret:	956
Interpolated Recall - Precision Averages	
at 0.00	0.3961
at 0.10	0.2479
at 0.20	0.1734
at 0.30	0.1258
at 0.40	0.1067
at 0.50	0.0838
at 0.60	0.0372
at 0.70	0.0195
at 0.80	0.0100
at 0.90	0.0029
at 1.00	0.0009
Average precision (non-interpolated) over all rel docs	
	0.0919
Precision:	
At 5 docs:	0.2426
At 10 docs:	0.2149
At 15 docs:	0.1801
At 20 docs:	0.1574
At 30 docs:	0.1383
At 100 docs:	0.0745
At 200 docs:	0.0522
At 500 docs:	0.0320
At 1000 docs:	0.0203
R-Precision (precision after R (= num_rel for a query) docs retrieved):	
Exact:	0.1283

Figure 17. Filtering Using Semantic Categories.

if the word "trains" is in the Query and the word "leaves" is in the Document and we look at the semantic category Motion with Reference to Direction (AMDR), then one of the vector product elements in the formula becomes:

$$\left(\begin{matrix}\text{weight of "trains"}\\ \text{in Query}\end{matrix}\right)\left(\begin{matrix}\text{probability}\\ \text{"trains" triggers AMDR}\end{matrix}\right) \cdot \left(\begin{matrix}\text{weight of "leaves"}\\ \text{in Document}\end{matrix}\right)\left(\begin{matrix}\text{probabilty}\\ \text{"leaves" triggers AMDR}\end{matrix}\right)$$

where the probabilities are obtained from our semantic lexicon.

We plan to do more experiments incorporating the following improvements:

a. Modernize the semantic lexicon. Since our lexicon is based on the 1911 version of Roget's Thesaurus, many modern words are not present and the senses of recorded words are not accurate. We plan to correct this. For example, we could try to get permission to use the current version of Roget's Thesaurus.

b. Base similarity on paragraphs instead of whole documents. We have had success using as few as 36 categories in a paragraph environment. We also feel that relevance decisions are made by humans looking at roughly a paragraph of information. We plan to modify our code to use paragraphs as a basis for the similarity measure.

c. Experiment with the number of possible semantic categories and the probability assigned to a triggered category. The experiment behind the performance improvement shown in Figure 16 and Figure 17 uses a very fine number of semantic categories and treats the triggered semantic categories for a word uniformly. We plan to experiment with a fewer number of categories, and we plan to obtain a probability distribution for categories based on word usage.

Basically, we are trying to establish a statistically sound approach to using word sense information. Intuition is that word sense information should improve retrieval performance. Furthermore, our approach to using word sense information has shown a significant performance improvement in a question/answer environment where paragraphs represent documents. We feel that other word sense approaches, such as query expansion or word sense disambiguation, may not be statistically sound, and that may be why successful experiments have not been reported.

References

[1] G. F. Cooper (1988). Probabilistic inference using belief networks is np-hard. *Technical Report KSL-87-27*, Stanford University, Stanford, CA.

[2] W. B. Croft (1987). Approaches to intelligent information retrieval. *Information Processing & Management*, 23(4):95-110.

[3] C. Date (1990). *An Introduction to Database Systems*, Vol. I, Addison Wesley.

[4] J. Driscoll, L. Lautenschlager and M. Zhao (1993). The QA system, *Proc. of the First Text Retrieval Conference (TREC-1), NIST Special Publication 500-207* (D. K. Harman, editor).

[5] E. A. Fox (1983) Characterization of two new experimental collections in computer and information science containing textual and bibliographic concepts. *Technical Report 83-561*, Department of Computer Science, Cornell Univ., Ithaca, NY.

[6] L. N. Kanal and J. F. Lemmer Eds (1986). *Uncertainty in Artificial Intelligence*, North-Holland, Amsterdam.

[7] J. F. Lemmer and L. N. Kanal Eds (1988). *Uncertainty in Artificial Intelligence 2*, North-Holland, Amsterdam.

[8] E. L. Liddy and S. H. Myaeng (1993). DR-Link, *Proc. of the First Text Retrieval Conference (TREC-1), NIST Special Publication 500-207* (D. K. Harman, editor).

[9] R. E. Neapolitan (1990). *Probabilistic Reasoning in Expert System, Theory and Algorithms*, A Wiley-Interscience Publication, John Wiley & Sons, Inc.

[10] J. Pearl (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann.

[11] Y. Peng and J. A. Reggia (1989). A connectionist model for diagnostic problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(2):285-298.

[12] K. Sparck Jones and R. Bates (1977). Research on Automatic Indexing 1974-1976, *Technical Report*, Computer Laboratory, University of Cambridge.

[13] M. Tagamets, J. Wald, M. Farach and J. A. Reggia (1989). Generating plausible diagnostic hypothesis with self-processing causal networks. *Journal of Experimental Theories in Artificial Intelligence*, 2:91-112.

[14] *Roget's International Thesaurus* (1977). Harper & Row, New York, Fourth Edition.

[15] H. Turtle and W. B. Croft (1991). Evaluation of an inference network-based retrieval model. *ACM Transaction on Information Systems*, 9(3):187-222.

[16] P. Van Laarh and E. Arts (1987). *Simulated Annealing: Theory and Applications*. Boston: D. Reidel.

[17] D. Voss and J. Driscoll (1992). Text Retrieval Using a Comprehensive Semantic Lexicon, *Proceedings of ISMM First International Conference on Information and Knowledge Management*, Baltimore, Maryland.

[18] P. Wang, J. Reggia, D. Nao and Y. Peng (1985). A formal model for diagnostic inference. *Information Sciences*, 37:227-285.

[19] E. Wendlandt and J. Driscoll (1991). Incorporating a Semantic Analysis into a Document Retrieval Strategy, *Proceedings of the Fourteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, Chicago, Illinois, 270-279.

The Efficiency Issues Workshop Report

James P. Callan
Computer Science Department
University of Massachusetts
Box 34610
Amherst, MA 01003-4610, USA
(callan@cs.umass.edu)

David D. Lewis
AT&T Bell Laboratories
600 Mountain Avenue
Room 2C409
Murray Hill, NJ 07974, USA
(lewis@research.att.com)

Although most groups participating in TREC-2 emphasized precision and recall, the conference was also an appropriate forum in which to discuss the efficiency of document indexing and retrieval. For some participants, running out of disk space was their worst problem, while for others running out of time was. Some groups were unable to run on the entire collection, and several remarked that they were happy to have gotten anything running at all. No group reported finding TREC trivial.

Two discussion groups were organized to address efficiency issues, one focusing on document indexing, the other on document retrieval. Since efficiency has not been emphasized by the research IR community, participants in both groups felt that current algorithms have a lot of room for improvement. TREC provides one motivation for such improvements, as do ever-growing real world databases. However, there was concern in both groups that the TREC format, which encourages participation in both ad hoc (retrospective) and routing (filtering) tasks, might discourage research on efficient task-specific architectures.

The following sections provide more detail on each of the two discussion groups.

1 Document Indexing

The raw text for the TREC collection (routing and ad hoc) required approximately 3 GB of space. Index structures required from 7.550themselves sufficient to recreate the original text, so they would be additional overhead in an operational system. (A research system might be able to discard the original text, reporting just document ids for evaluation.)

Several groups, including CITRI, Thinking Machines, and UMass, stored inverted lists in compressed form. There was general agreement that for sites willing to invest the programming effort, substantial space savings could be achieved in this fashion. (CITRI demonstrated a factor of six reduction in index file size.) There was more debate on the potential for index compression speeding up query processing as well, with some participants saying their query processing was I/O bound, but others saying theirs was CPU bound. The peak amount of space used during indexing (for text, indices, and auxiliary files) varied from 110that this may be worth more attention.

Efficiency improvements will not come immediately, and some may require significant expense in programming time. Sharing of software between groups, which increased from TREC-1 to TREC-2, helps limit this expense. In addition, TREC research groups primarily interested in, say, query analysis, may in the future want to team up with groups that have addressed or are addressing issues of scale. As the size of test collections increases, it makes less sense to have them replicated at dozens of sites, particular when interactive access across networks is usually easily available.

Time to build index structures was tolerable for most participants, though it was mentioned that it never seems to go as easily or automatically as one might hope. It was a serious issue for groups doing some form of natural language processing. Times of 2 to 4 MB/hr were mentioned by at least two of the NL groups, making TREC a very daunting task. The opinion was expressed by some participants that an NL technique would have to provide as yet undemonstrated improvements in effectiveness to be worth the slowdown in indexing and query processing.

Complex text representations, such as those produced by NL, require additional information in the index structures. The different ways of dealing with this problem are most noticeable in the handling of phrases in the TREC, with some groups indexing on phrases just as on words, others relying on word position information stored in an inverted file, and still others reparsing the raw text of a subset of documents at retrieval time to find phrase occurrences.

2 Document Retrieval

The second discussion group was organized around general issues in document retrieval. Participants were encouraged to use their experience with the one and two gigabyte TREC collections to forecast the issues that will arise when collections are larger and more distributed. Two issues dominated the discussion.

2.1 Will Existing Methods Scale?

Recent trends in information retrieval are towards gigabyte and terabyte document collections, retrieval by subsections/paragraphs, and multiple representations of document content. The first focus questions were whether existing storage and retrieval methods can cope with this explosion of information, and if new methods are needed, what might they be?

Participants identified two approaches as currently dominating IR:

1. Search all available subcollections (e.g. the TIPSTER/TREC task), or
2. have a user specify which subcollection to search (e.g. commercial systems).

Both approaches require modification if they are to scale up.

One problem with the "search everything" approach is that the growth rate of online information was felt to exceed the growth rate in computer performance. Even if a collection is distributed across multiple processors, detailed consideration of every document may be too expensive when an IR system faces a terabyte of data. One solution is to do a fast first-pass retrieval to produce a reduced set of documents for more detailed consideration. This first pass might involve generating approximate scores for each document (e.g. ETH in TREC-2), scoring documents based on a smaller amount of text (e.g. abstracts or introductions), or scoring cluster centroids rather than individual documents.

One problem with the "user chooses subcollections" approach is that the task will be overwhelming when there are many subcollections. A significant portion of users of one commercial service already choose to search everything rather than select subcollections. The system will have to provide assistance if user selection is to be viable. If subcollections can be characterized succinctly, perhaps by centroid vectors, automatically generated thesauri, or controlled vocabulary terms (assigned manually or automatically), then one could use the query to rank subcollections, and then search only the top-ranked subcollections. Other approaches include assistance by an expert system, or browsing interfaces for hyperlinked subcollections.

Global statistics that summarize some aspect of a collection (e.g. *idf*) were expected to be a problem for searching multiple subcollections and distributed document collections. If a collection is formed at indexing time, statistics can be gathered and saved when indices are built. If a single processor performs retrieval, statistics can be gathered during retrieval. However, if multiple subcollections or processors are involved, it is less clear how to compute global statistics. Methods that rely only on *local statistics* that summarize a document (e.g. Berkeley in TREC-2) offer a computational advantage in this environment.

2.2 Specialized Hardware and Software

A related question posed by increasingly large and distributed text databases is whether existing hardware and software platforms will be up to the challenge. The consensus among participants was that conventional architectures will suffice, because IR is a data-parallel task that lends itself to distributed computation. Participants also felt that they had generally ignored issues of efficiency, and could increase their speeds if necessary. There was little support for supercomputers, massively parallel computers, or specialized architectures.

One could argue that the participants were biased towards conventional hardware and software by their own need for flexibility, their small budgets, their insulation from the time constraints of real users, and their desire not to think about 'systems' issues not directly relevant to their research. However, the recent fielding of ranked retrieval systems using conventional mainframes by some of the largest online vendors provides additional support for their views that conventional architectures will suffice.

Workshop Report
Use of training materials in constructing routing queries

William S. Cooper
Stephen E. Robertson

The participants outlined various methods of exploiting the training data for routing retrieval that had been used in the conference. In all cases the data had been used in a topic- specific manner; i.e. each query was constructed or expanded using relevance judgements for that particular topic only.

In some systems, terms taken directly from the topic were weighted or reweighted using the training data. In others, terms taken from the training documents relevant to the topic were used in addition to topic terms, or were used instead with the original topic terms playing no part. In a few cases, terms from both relevant and non-relevant documents were added, the latter with negative weights. Relevant documents with a high preliminary retrieval ranking coefficient were preferred as a source of expansion terms in one system. Probabilistic, feedback and ad-hoc methods had all been tried as ways of modifying the query in response to the training data.

How far might a query profitably be expanded on the basis of the training data? Though this question was not answered definitively, some participants indicated a greater willingness to consider drastic expansion than had been thought advisable before TREC 2.

The sample of relevance judgements for TREC 2 was thought to be adequate in size and not unrealistically large. It was sufficiently representative in its inclusiveness of feedback generated from a wide variety of systems. However, this variety indicates a possible lack of realism, in that a real system would probably have access only to relevant documents retrieved by itself. Thus the use of only those relevant documents found in a search on the training data by the system in question might be regarded as more realistic.

APPENDIX A

This appendix contains tables of results for all the TREC-2 participants. The tables in Appendix A show various measures of the performance on the adhoc and routing tasks. The adhoc results come first, followed by the routing results, with the tables in the same order as the presentation order of the papers. The definitions of the evaluation measures are given in the Overview, section 4, and readers unfamiliar with these measures should read that section first.

Care should be taken in comparing the tables across systems. These measures show performance only, with no measure of user or system effort.

Each table contains four major boxes of statistics and three graphs.

Box 1 -- Summary Statistics

line 1 -- unique run identifier, data subset, and query construction method used

Data subset
full (disks 1 and 2 for adhoc, disk 3 for routing)
category B (the official subset of data, 1/4 of the data using the Wall Street Journal articles for adhoc and the San Jose Mercury News articles for routing)

Query construction method
automatic
manual
feedback (frozen evaluation used)

line 2 -- Number of topics included in averages.

line 3 -- Total number of documents retrieved over all topics. Here, "retrieved" means having a rank less 1001.

line 4 -- Total number of relevant documents for all topics in the collection (whether retrieved or not).

line 5 -- Total number of relevant retrieved documents for this run.

Box 2 -- Recall Level Averages

lines 1-11 -- The average over all topics of the precision at each of the 11 recall points given. Note that this is interpolated precision: e.g., for a particular topic, if the precision at 0.50 recall is greater than the precision at 0.40 recall, then the precision at 0.50 recall is used for both the 0.50 and 0.40 recall levels.

line 12 -- The average precision as calculated in a non-interpolated manner (see section 4 of the Overview for details on this calculation).

Box 3 -- Document Level Averages

lines 1-9 -- The average recall and precision after the given number of documents have been retrieved.

line 10 -- the R precision. This is a new evaluation measure being tried that averages the precisions found for each topic at the document level of R, where R is the number of relevant documents for that topic.

Box 4 -- Fallout Level Averages

lines 1-11 -- The average recall (probability of detection) at a fixed fallout (probability of false alarm rate). Ten equally spaced fallout points in the high precision end of the curve were used, and for each topic, the highest recall value in the region surrounding that fallout point is selected. The table shows the averages of these points across all topics.

Graph 1 -- Recall-Precision Curve

This is a plot of the data shown in Box 2.

Graph 2 -- Fallout-Recall Curve

This is a plot of the data shown in Box 4.

Graph 3 -- Normal Deviate - Fallout-Recall

This is a plot of the data shown in Box 4, but plotted on probability scales.

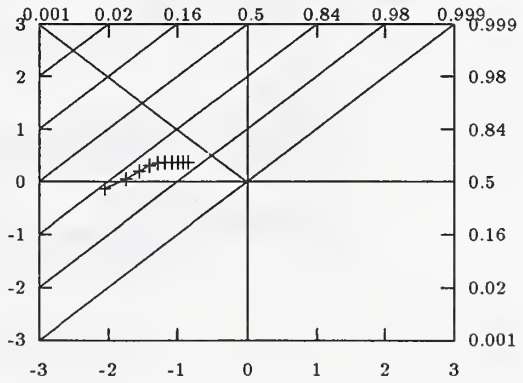
Summary Statistics	
Run Number	cityau–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	6412

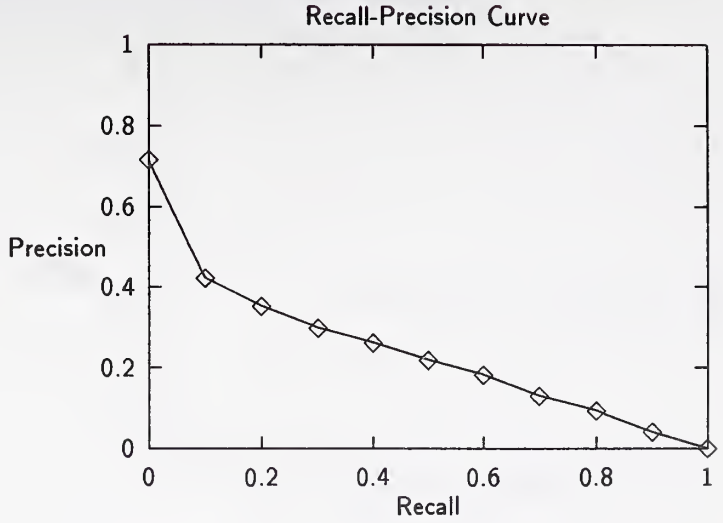
Recall Level Averages	
Recall	Precision
0.00	0.7148
0.10	0.4224
0.20	0.3528
0.30	0.2983
0.40	0.2623
0.50	0.2191
0.60	0.1835
0.70	0.1299
0.80	0.0937
0.90	0.0421
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.2272

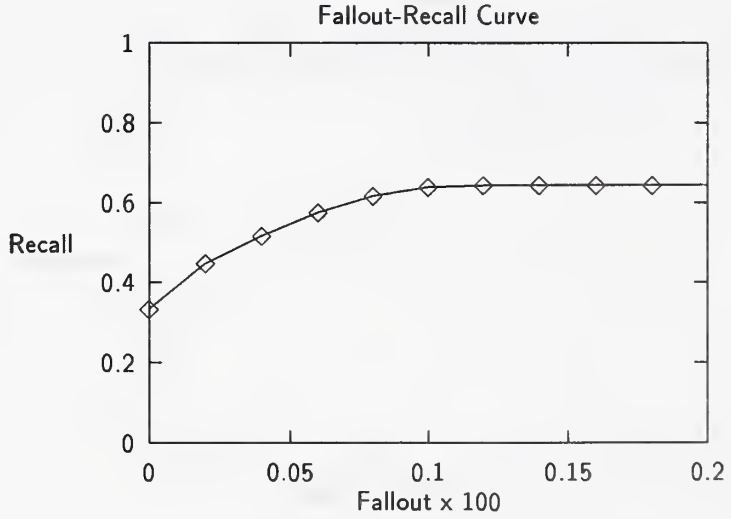
Document Level Averages	
	Precision
At 5 docs	0.5000
At 10 docs	0.4640
At 15 docs	0.4667
At 20 docs	0.4570
At 30 docs	0.4340
At 100 docs	0.3512
At 200 docs	0.2970
At 500 docs	0.1966
At 1000 docs	0.1282
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2849

Fallout Level Averages	
Fallout	Recall
0.00000	0.3344
0.00020	0.4481
0.00040	0.5173
0.00060	0.5761
0.00080	0.6171
0.00100	0.6384
0.00120	0.6438
0.00140	0.6438
0.00160	0.6438
0.00180	0.6438
0.00200	0.6438

NORMAL DEVIATE - Fallout-Recall





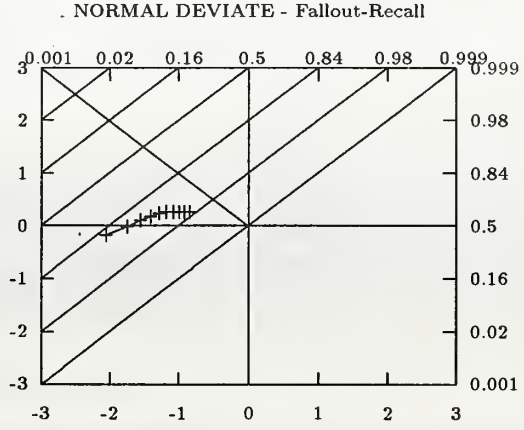


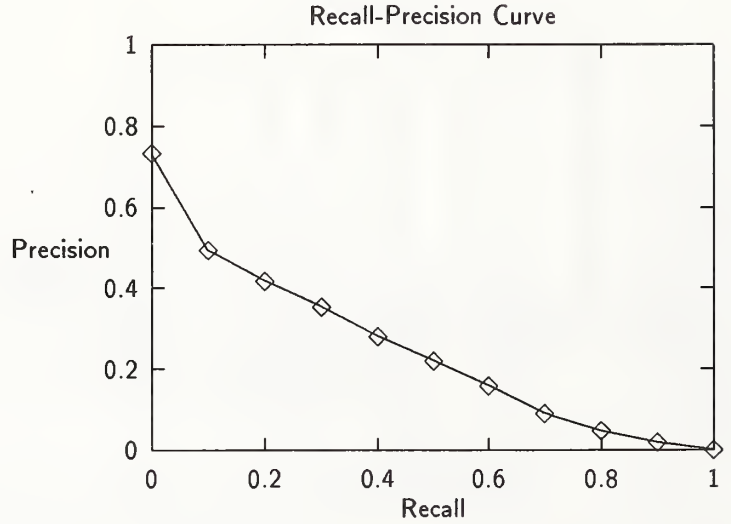
Summary Statistics	
Run Number	citymf–full, feedback
Num of Queries	50
Total number of documents over all queries	
Retrieved:	51221
Relevant:	10785
Rel_ret:	6409

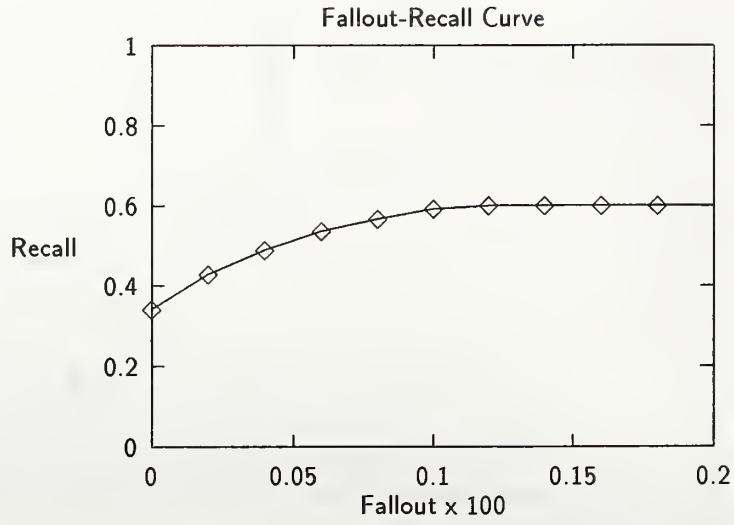
Recall Level Averages	
Recall	Precision
0.00	0.7328
0.10	0.4944
0.20	0.4180
0.30	0.3552
0.40	0.2809
0.50	0.2205
0.60	0.1584
0.70	0.0901
0.80	0.0482
0.90	0.0184
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.2324

Document Level Averages	
	Precision
At 5 docs	0.4920
At 10 docs	0.5040
At 15 docs	0.4827
At 20 docs	0.4750
At 30 docs	0.4680
At 100 docs	0.4004
At 200 docs	0.3162
At 500 docs	0.1979
At 1000 docs	0.1275
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2971

Fallout Level Averages	
Fallout	Recall
0.00000	0.3417
0.00020	0.4289
0.00040	0.4898
0.00060	0.5366
0.00080	0.5682
0.00100	0.5932
0.00120	0.6012
0.00140	0.6012
0.00160	0.6012
0.00180	0.6012
0.00200	0.6012







Summary Statistics	
Run Number	rutcombx–full, partial topics, manual
Num of Queries	25

Total number of documents over all queries	
Retrieved:	25000
Relevant:	4119
Rel_ret:	2378

Recall Level Averages	
Recall	Precision
0.00	0.7145
0.10	0.4617
0.20	0.3894
0.30	0.3325
0.40	0.2841
0.50	0.2007
0.60	0.1502
0.70	0.1057
0.80	0.0701
0.90	0.0345
1.00	0.0104

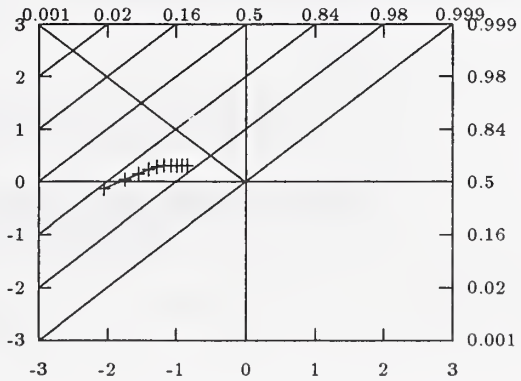
Average precision over all relevant docs	
non-interpolated	0.2284

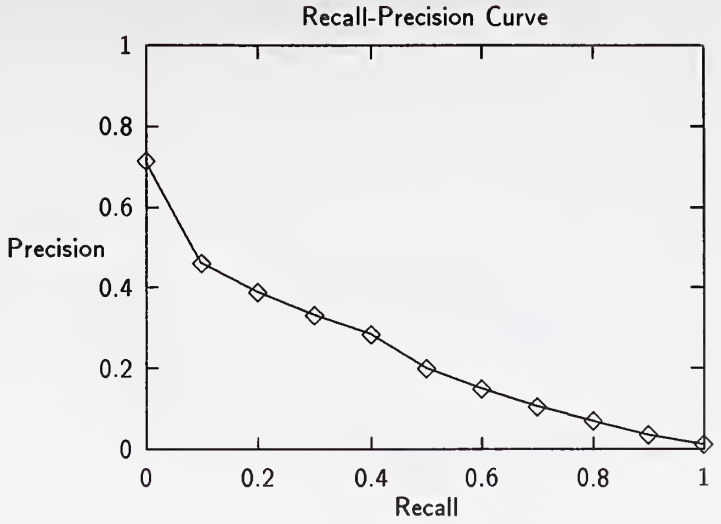
Document Level Averages	
	Precision
At 5 docs	0.4720
At 10 docs	0.4600
At 15 docs	0.4640
At 20 docs	0.4520
At 30 docs	0.4280
At 100 docs	0.3396
At 200 docs	0.2432
At 500 docs	0.1489
At 1000 docs	0.0951

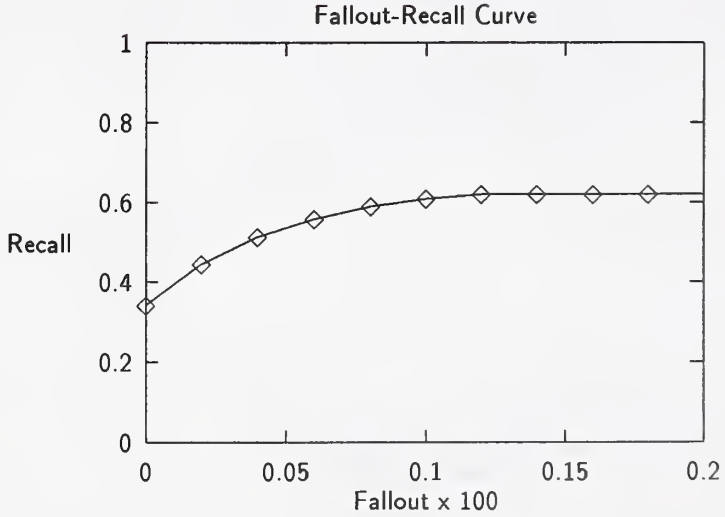
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2785

Fallout Level Averages	
Fallout	Recall
0.00000	0.3413
0.00020	0.4453
0.00040	0.5131
0.00060	0.5584
0.00080	0.5901
0.00100	0.6090
0.00120	0.6202
0.00140	0.6202
0.00160	0.6202
0.00180	0.6202
0.00200	0.6202

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	rutfmed–full, partial topics, manual
Num of Queries	25

Total number of documents over all queries	
Retrieved:	24001
Relevant:	4119
Rel_ret:	2314

Recall Level Averages	
Recall	Precision
0.00	0.6054
0.10	0.4215
0.20	0.3574
0.30	0.2726
0.40	0.2152
0.50	0.1459
0.60	0.1067
0.70	0.0753
0.80	0.0496
0.90	0.0204
1.00	0.0000

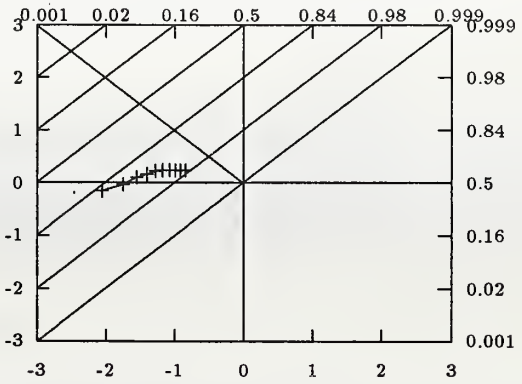
Average precision over all relevant docs	
non-interpolated	0.1852

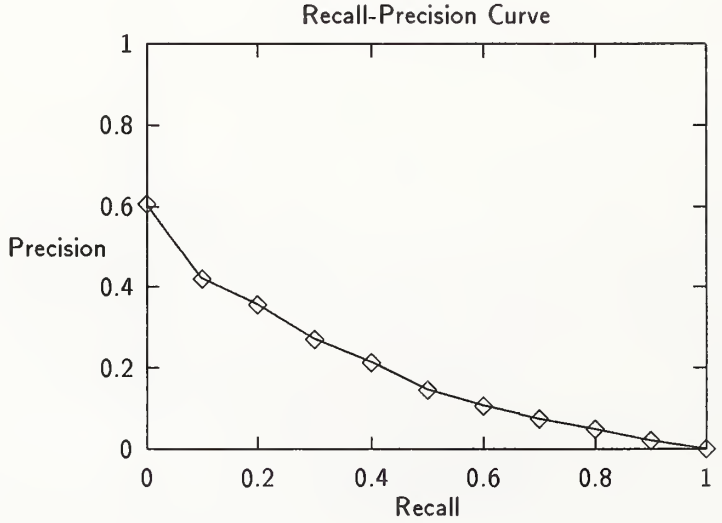
Document Level Averages	
	Precision
At 5 docs	0.3920
At 10 docs	0.3880
At 15 docs	0.3787
At 20 docs	0.3860
At 30 docs	0.3733
At 100 docs	0.2892
At 200 docs	0.2256
At 500 docs	0.1418
At 1000 docs	0.0926

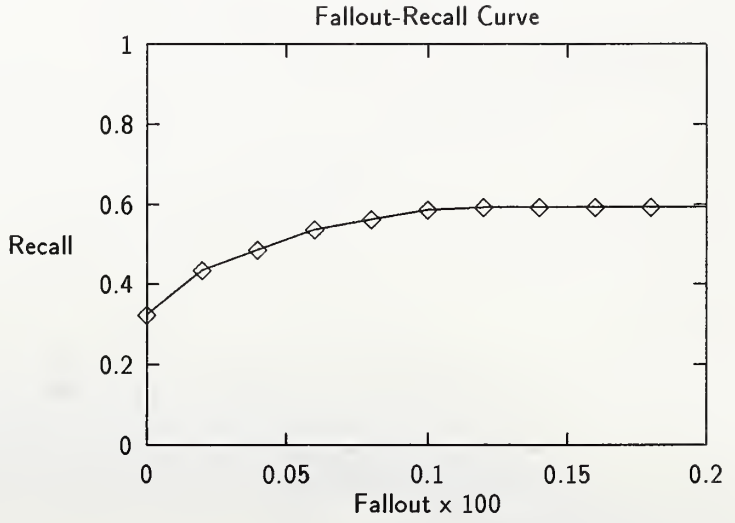
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2464

Fallout Level Averages	
Fallout	Recall
0.00000	0.3238
0.00020	0.4356
0.00040	0.4861
0.00060	0.5370
0.00080	0.5625
0.00100	0.5863
0.00120	0.5933
0.00140	0.5933
0.00160	0.5933
0.00180	0.5933
0.00200	0.5933

NORMAL DEVIATE - Fallout-Recall





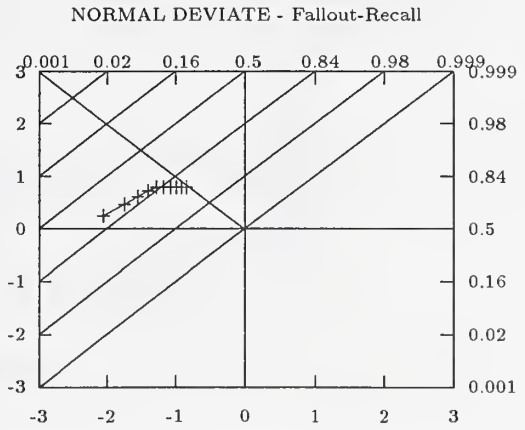


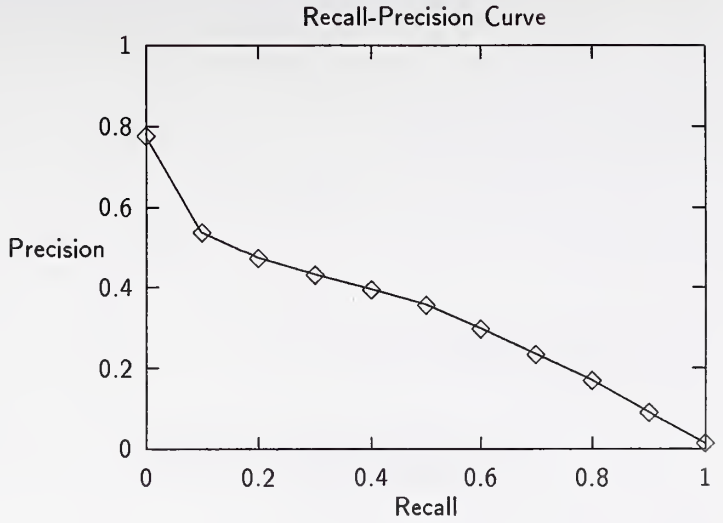
Summary Statistics	
Run Number	crnlL2–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	8224

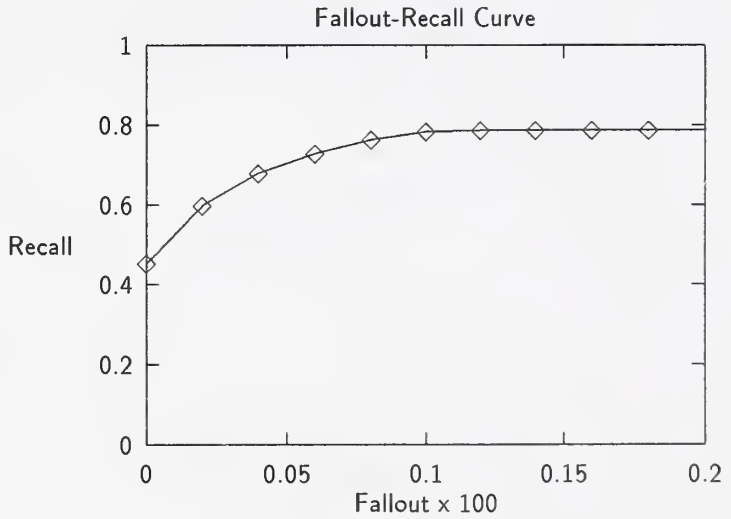
Recall Level Averages	
Recall	Precision
0.00	0.7756
0.10	0.5381
0.20	0.4732
0.30	0.4321
0.40	0.3959
0.50	0.3585
0.60	0.2991
0.70	0.2357
0.80	0.1705
0.90	0.0904
1.00	0.0148
Average precision over all relevant docs	
non-interpolated	0.3258

Document Level Averages	
	Precision
At 5 docs	0.5640
At 10 docs	0.5460
At 15 docs	0.5373
At 20 docs	0.5210
At 30 docs	0.4933
At 100 docs	0.4406
At 200 docs	0.3852
At 500 docs	0.2600
At 1000 docs	0.1645
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3641

Fallout Level Averages	
Fallout	Recall
0.00000	0.4533
0.00020	0.5983
0.00040	0.6796
0.00060	0.7287
0.00080	0.7634
0.00100	0.7838
0.00120	0.7864
0.00140	0.7864
0.00160	0.7864
0.00180	0.7864
0.00200	0.7864







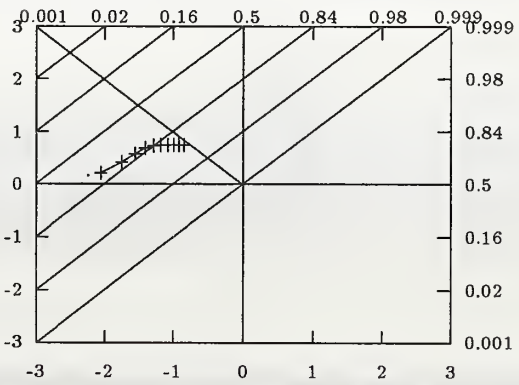
Summary Statistics	
Run Number	crnlV2–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	8018

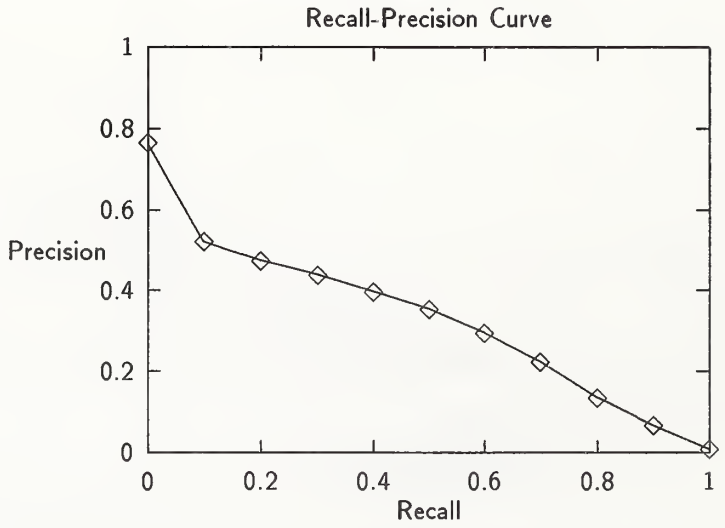
Recall Level Averages	
Recall	Precision
0.00	0.7644
0.10	0.5209
0.20	0.4739
0.30	0.4388
0.40	0.3971
0.50	0.3544
0.60	0.2950
0.70	0.2241
0.80	0.1356
0.90	0.0661
1.00	0.0076
Average precision over all relevant docs	
non-interpolated	0.3163

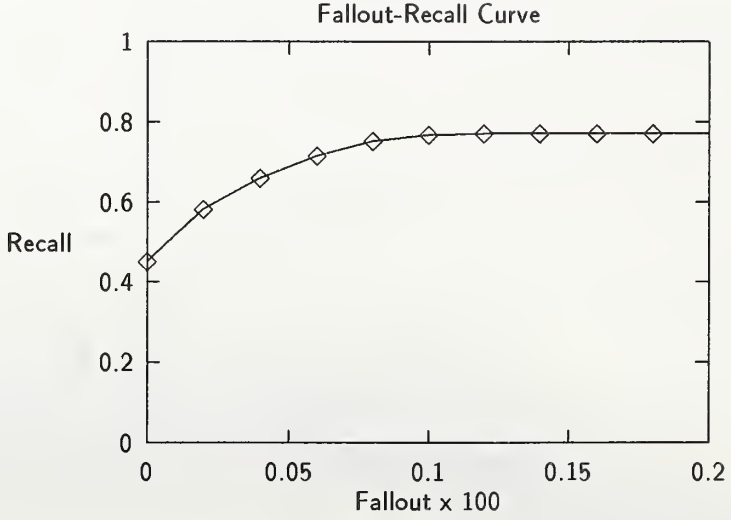
Document Level Averages	
	Precision
At 5 docs	0.5480
At 10 docs	0.5220
At 15 docs	0.5187
At 20 docs	0.5130
At 30 docs	0.5013
At 100 docs	0.4434
At 200 docs	0.3789
At 500 docs	0.2529
At 1000 docs	0.1604
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3640

Fallout Level Averages	
Fallout	Recall
0.00000	0.4516
0.00020	0.5815
0.00040	0.6605
0.00060	0.7148
0.00080	0.7514
0.00100	0.7675
0.00120	0.7713
0.00140	0.7713
0.00160	0.7713
0.00180	0.7713
0.00200	0.7713

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	Brkly3–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	7766

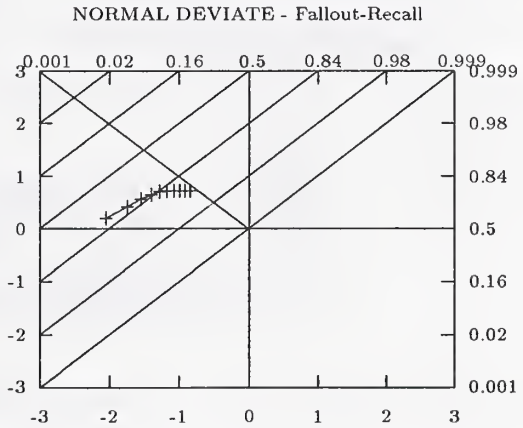
Recall Level Averages	
Recall	Precision
0.00	0.8467
0.10	0.5630
0.20	0.5016
0.30	0.4423
0.40	0.3946
0.50	0.3497
0.60	0.2963
0.70	0.2126
0.80	0.1490
0.90	0.0543
1.00	0.0040

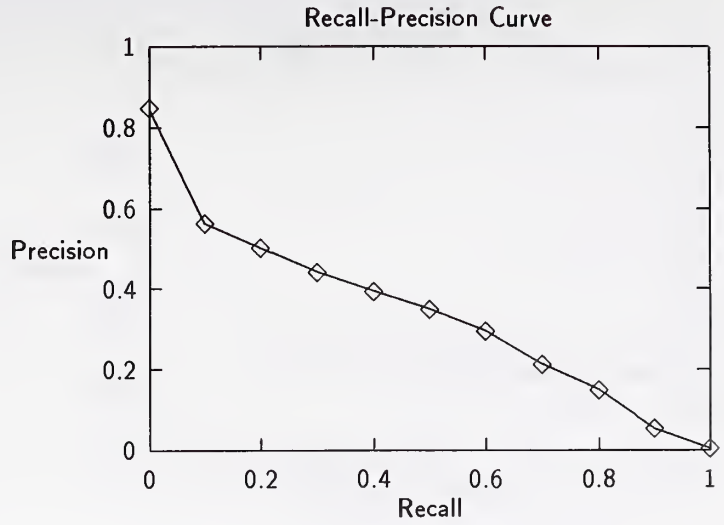
Average precision over all relevant docs	
non-interpolated	0.3270

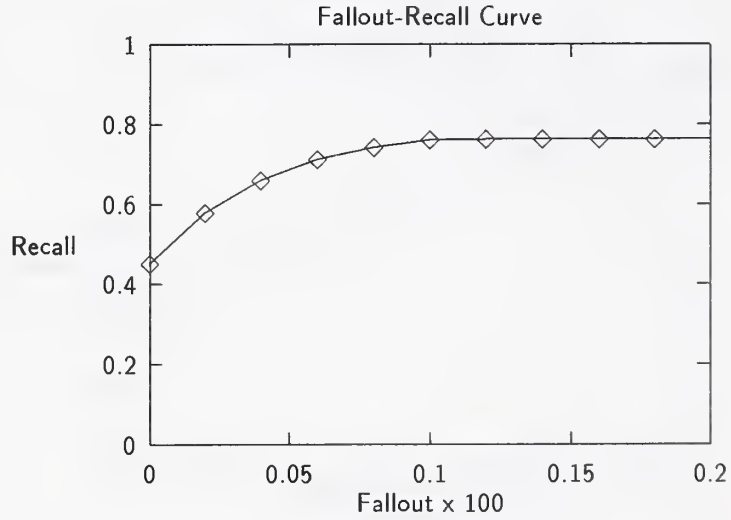
Document Level Averages	
	Precision
At 5 docs	0.6000
At 10 docs	0.5760
At 15 docs	0.5587
At 20 docs	0.5400
At 30 docs	0.5287
At 100 docs	0.4568
At 200 docs	0.3795
At 500 docs	0.2485
At 1000 docs	0.1553

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3697

Fallout Level Averages	
Fallout	Recall
0.00000	0.4522
0.00020	0.5786
0.00040	0.6602
0.00060	0.7125
0.00080	0.7417
0.00100	0.7611
0.00120	0.7635
0.00140	0.7635
0.00160	0.7635
0.00180	0.7635
0.00200	0.7635







Summary Statistics	
Run Number	dortL2–full, automatic
Num of Queries	50

Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	7774

Recall Level Averages	
Recall	Precision
0.00	0.7324
0.10	0.5410
0.20	0.4939
0.30	0.4370
0.40	0.3850
0.50	0.3324
0.60	0.2821
0.70	0.2133
0.80	0.1340
0.90	0.0707
1.00	0.0063

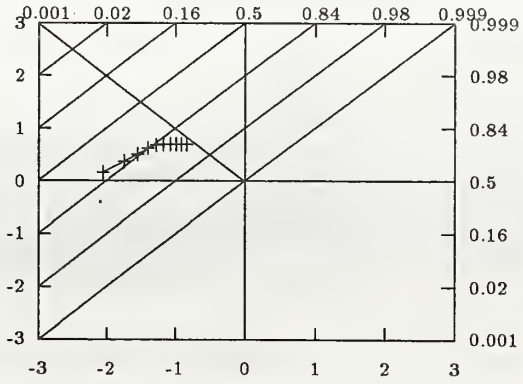
Average precision over all relevant docs	
non-interpolated	0.3151

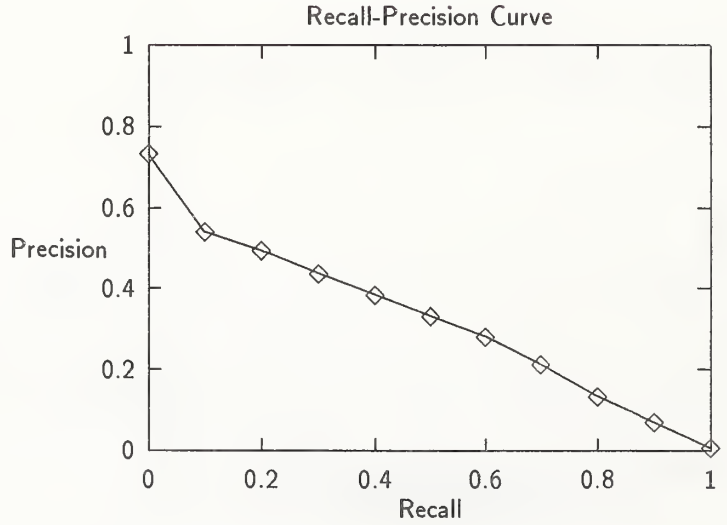
Document Level Averages	
	Precision
At 5 docs	0.5120
At 10 docs	0.5280
At 15 docs	0.5213
At 20 docs	0.5210
At 30 docs	0.5120
At 100 docs	0.4562
At 200 docs	0.3781
At 500 docs	0.2439
At 1000 docs	0.1555

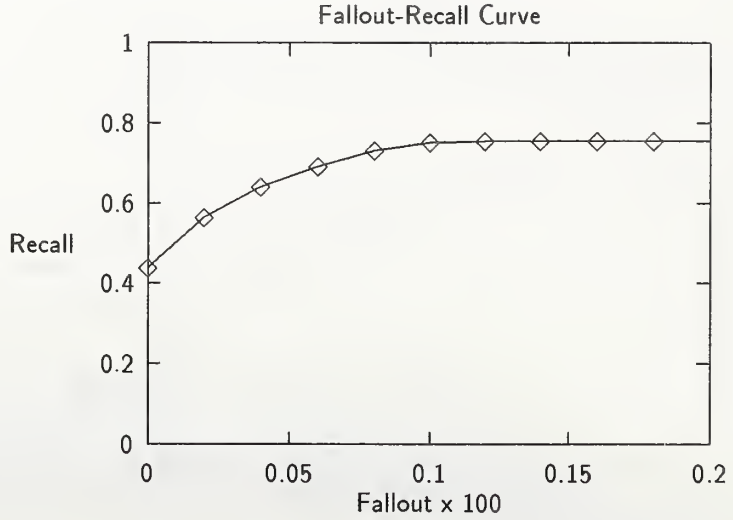
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3604

Fallout Level Averages	
Fallout	Recall
0.00000	0.4380
0.00020	0.5641
0.00040	0.6401
0.00060	0.6920
0.00080	0.7313
0.00100	0.7519
0.00120	0.7557
0.00140	0.7557
0.00160	0.7557
0.00180	0.7557
0.00200	0.7557

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	dortQ2–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	8259

Recall Level Averages	
Recall	Precision
0.00	0.7673
0.10	0.5518
0.20	0.5060
0.30	0.4572
0.40	0.4114
0.50	0.3635
0.60	0.3048
0.70	0.2388
0.80	0.1565
0.90	0.0777
1.00	0.0086

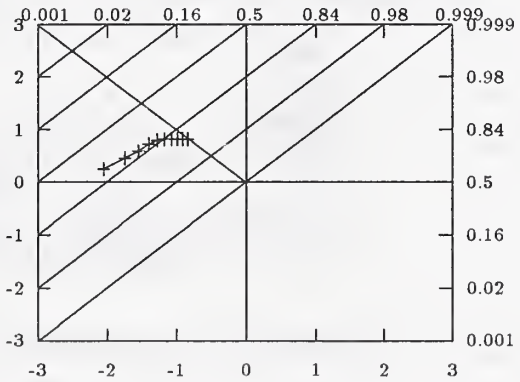
Average precision over all relevant docs	
non-interpolated	0.3340

Document Level Averages	
	Precision
At 5 docs	0.5560
At 10 docs	0.5440
At 15 docs	0.5427
At 20 docs	0.5440
At 30 docs	0.5213
At 100 docs	0.4626
At 200 docs	0.3959
At 500 docs	0.2607
At 1000 docs	0.1652

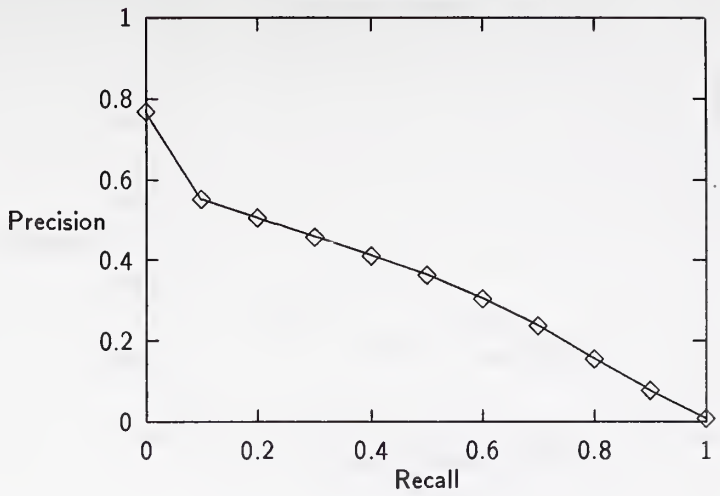
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3722

Fallout Level Averages	
Fallout	Recall
0.00000	0.4625
0.00020	0.5959
0.00040	0.6731
0.00060	0.7216
0.00080	0.7643
0.00100	0.7850
0.00120	0.7908
0.00140	0.7908
0.00160	0.7908
0.00180	0.7908
0.00200	0.7908

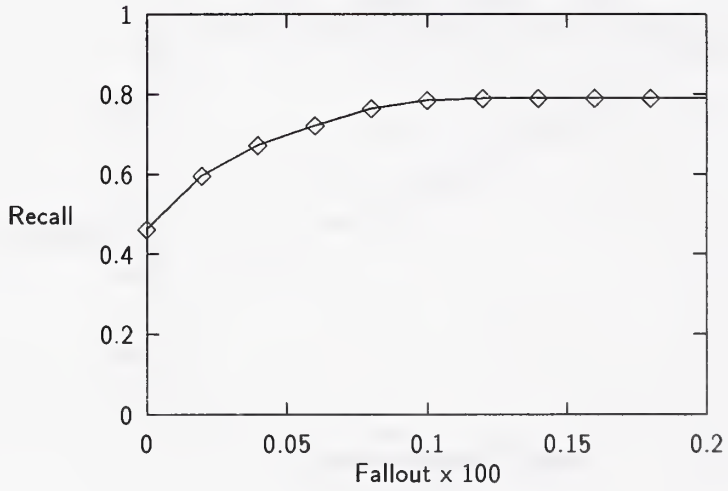
NORMAL DEVIATE - Fallout-Recall



Recall-Precision Curve



Fallout-Recall Curve



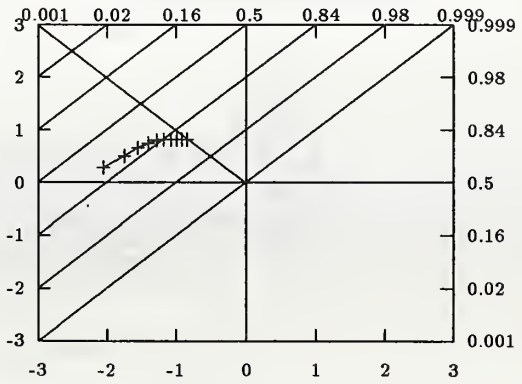
Summary Statistics	
Run Number	INQ001–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	8281

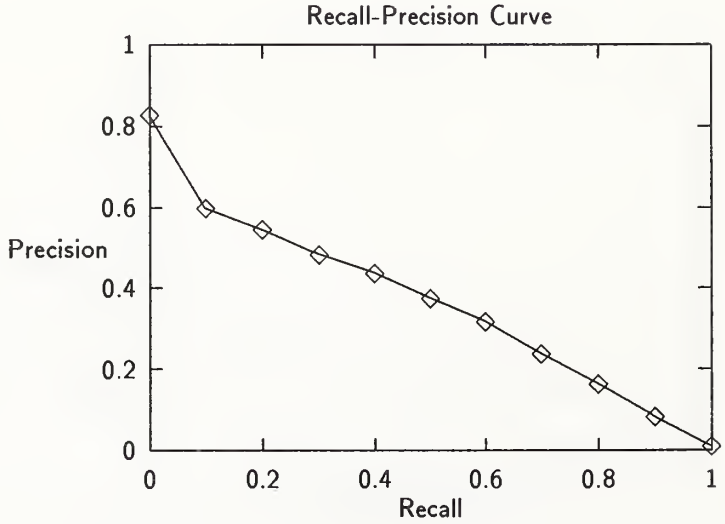
Recall Level Averages	
Recall	Precision
0.00	0.8275
0.10	0.5979
0.20	0.5458
0.30	0.4832
0.40	0.4369
0.50	0.3758
0.60	0.3181
0.70	0.2382
0.80	0.1634
0.90	0.0825
1.00	0.0094
Average precision over all relevant docs	
non-interpolated	0.3556

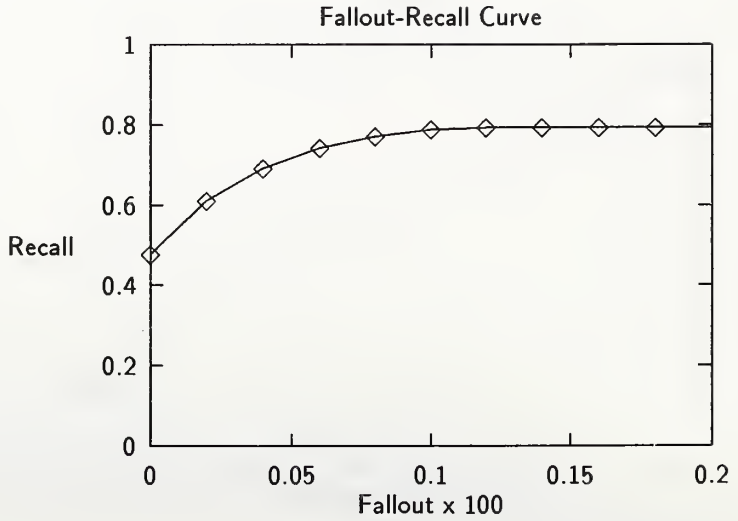
Document Level Averages	
	Precision
At 5 docs	0.6160
At 10 docs	0.5960
At 15 docs	0.5880
At 20 docs	0.5840
At 30 docs	0.5733
At 100 docs	0.4934
At 200 docs	0.4099
At 500 docs	0.2665
At 1000 docs	0.1656
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3946

Fallout Level Averages	
Fallout	Recall
0.00000	0.4769
0.00020	0.6107
0.00040	0.6913
0.00060	0.7427
0.00080	0.7713
0.00100	0.7890
0.00120	0.7933
0.00140	0.7933
0.00160	0.7933
0.00180	0.7933
0.00200	0.7933

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	INQ002–full, manual
Num of Queries	50

Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	8165

Recall Level Averages	
Recall	Precision
0.00	0.7762
0.10	0.6299
0.20	0.5604
0.30	0.4920
0.40	0.4331
0.50	0.3722
0.60	0.3133
0.70	0.2268
0.80	0.1545
0.90	0.0875
1.00	0.0122

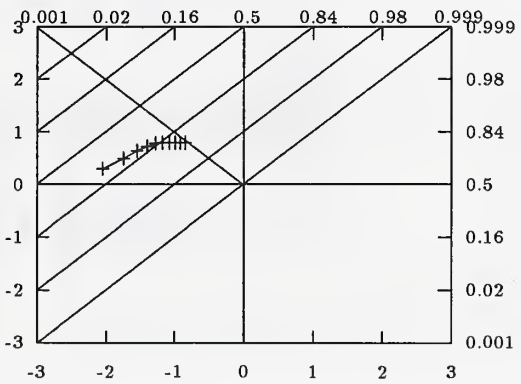
Average precision over all relevant docs	
non-interpolated	0.3565

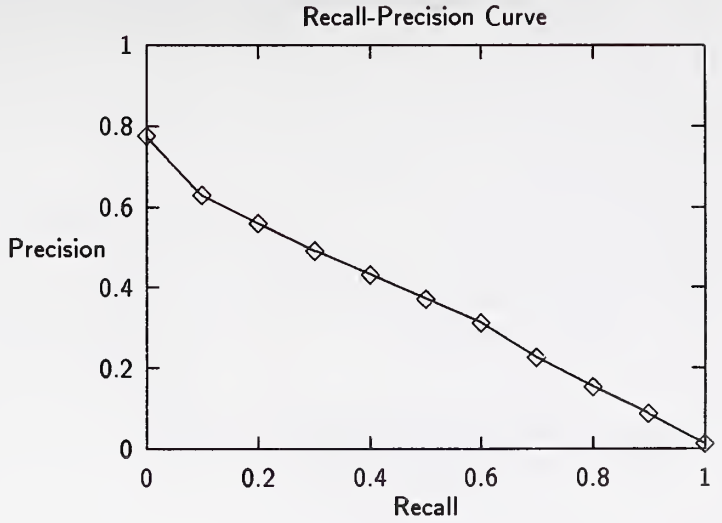
Document Level Averages	
	Precision
At 5 docs	0.6000
At 10 docs	0.6160
At 15 docs	0.6040
At 20 docs	0.6140
At 30 docs	0.5880
At 100 docs	0.5058
At 200 docs	0.4089
At 500 docs	0.2626
At 1000 docs	0.1633

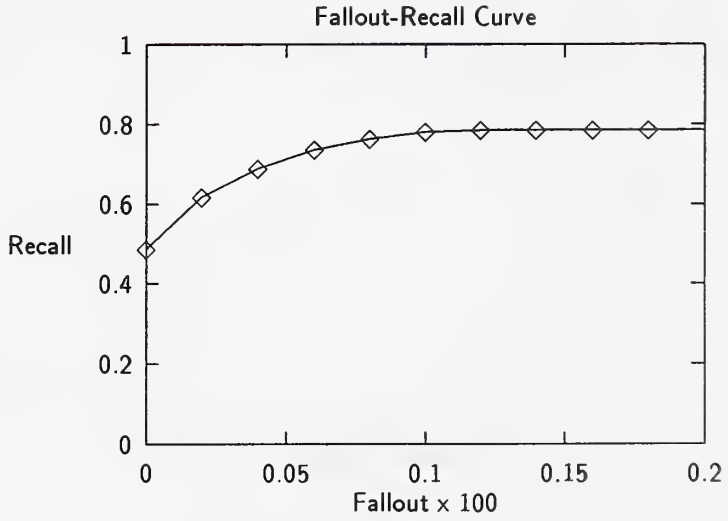
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3954

Fallout Level Averages	
Fallout	Recall
0.00000	0.4860
0.00020	0.6161
0.00040	0.6885
0.00060	0.7359
0.00080	0.7639
0.00100	0.7802
0.00120	0.7859
0.00140	0.7859
0.00160	0.7859
0.00180	0.7859
0.00200	0.7859

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	HNCad1–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	6944

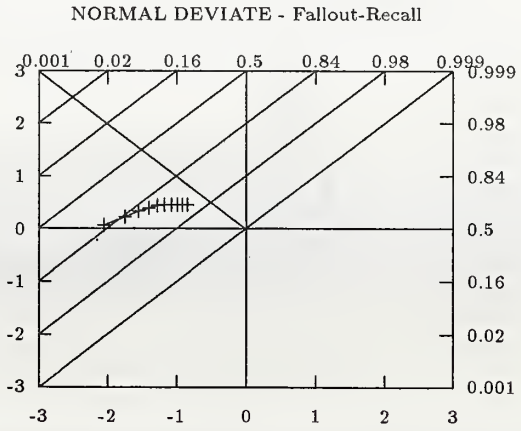
Recall Level Averages	
Recall	Precision
0.00	0.7750
0.10	0.5762
0.20	0.4934
0.30	0.4215
0.40	0.3536
0.50	0.2672
0.60	0.2041
0.70	0.1271
0.80	0.0575
0.90	0.0215
1.00	0.0007

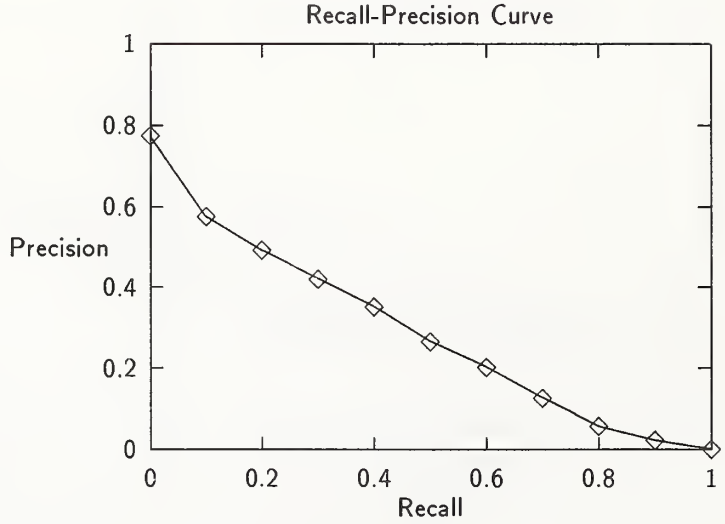
Average precision over all relevant docs	
non-interpolated	0.2787

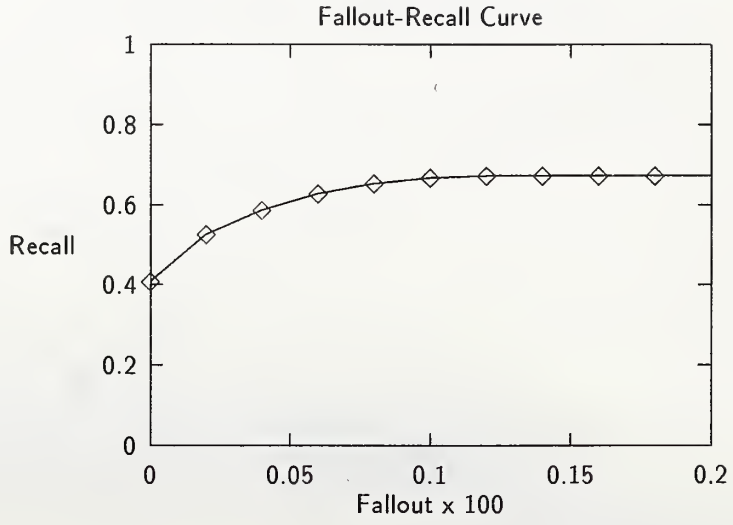
Document Level Averages	
	Precision
At 5 docs	0.5560
At 10 docs	0.5680
At 15 docs	0.5507
At 20 docs	0.5470
At 30 docs	0.5367
At 100 docs	0.4520
At 200 docs	0.3652
At 500 docs	0.2247
At 1000 docs	0.1389

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3474

Fallout Level Averages	
Fallout	Recall
0.00000	0.4087
0.00020	0.5261
0.00040	0.5867
0.00060	0.6272
0.00080	0.6533
0.00100	0.6681
0.00120	0.6720
0.00140	0.6720
0.00160	0.6720
0.00180	0.6720
0.00200	0.6720







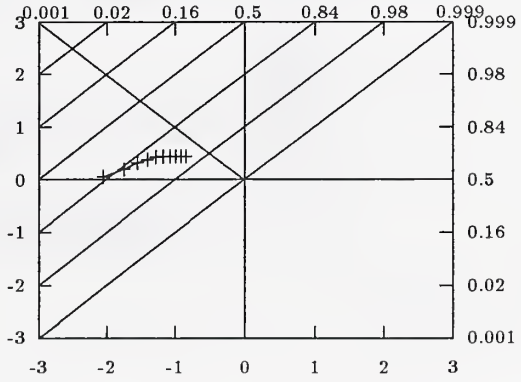
Summary Statistics	
Run Number	HNCad2–full, feedback
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	6911

Recall Level Averages	
Recall	Precision
0.00	0.7811
0.10	0.5920
0.20	0.5262
0.30	0.4429
0.40	0.3574
0.50	0.2767
0.60	0.2059
0.70	0.1250
0.80	0.0713
0.90	0.0218
1.00	0.0006
Average precision over all relevant docs	
non-interpolated	0.2882

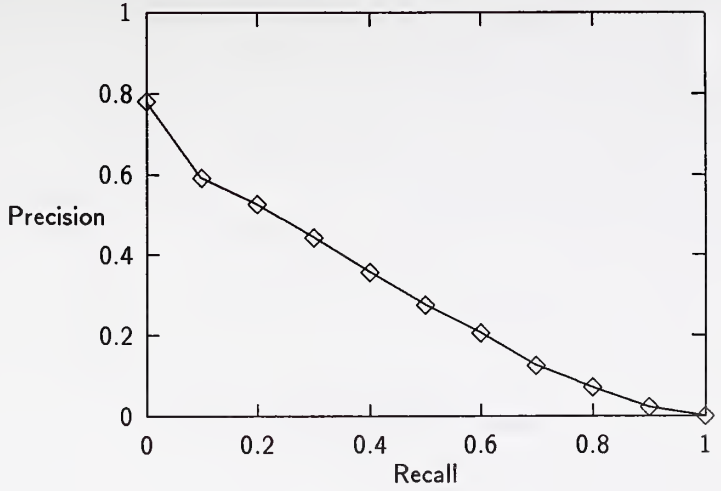
Document Level Averages	
	Precision
At 5 docs	0.5560
At 10 docs	0.5680
At 15 docs	0.5507
At 20 docs	0.5470
At 30 docs	0.5513
At 100 docs	0.4784
At 200 docs	0.3714
At 500 docs	0.2243
At 1000 docs	0.1382
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3575

Fallout Level Averages	
Fallout	Recall
0.00000	0.4132
0.00020	0.5207
0.00040	0.5790
0.00060	0.6233
0.00080	0.6485
0.00100	0.6643
0.00120	0.6676
0.00140	0.6676
0.00160	0.6676
0.00180	0.6676
0.00200	0.6676

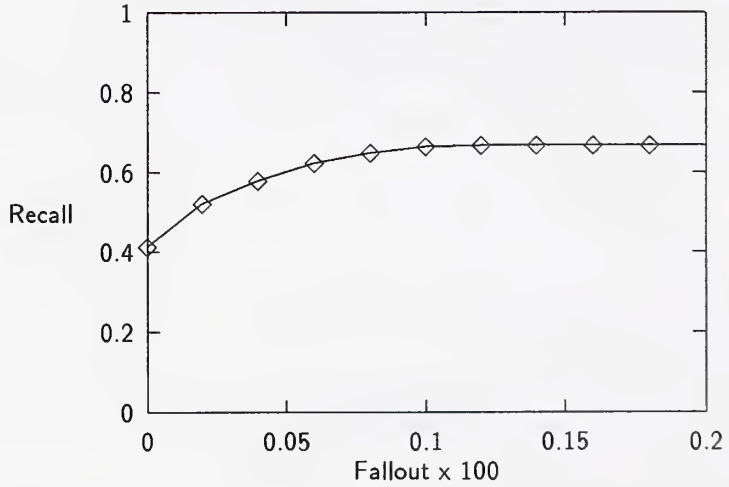
NORMAL DEVIATE - Fallout-Recall



Recall-Precision Curve



Fallout-Recall Curve



Summary Statistics	
Run Number	lsia1–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	4756

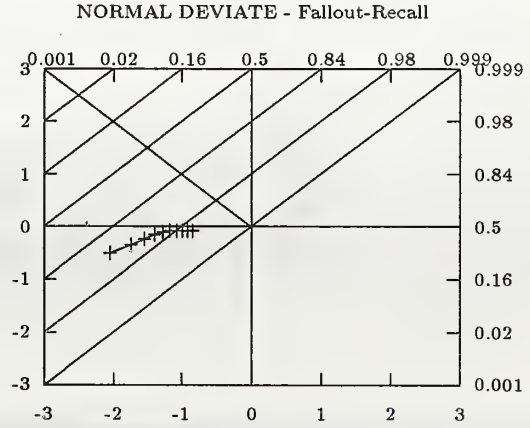
Recall Level Averages	
Recall	Precision
0.00	0.5897
0.10	0.3100
0.20	0.2392
0.30	0.1929
0.40	0.1482
0.50	0.1077
0.60	0.0616
0.70	0.0257
0.80	0.0159
0.90	0.0000
1.00	0.0000

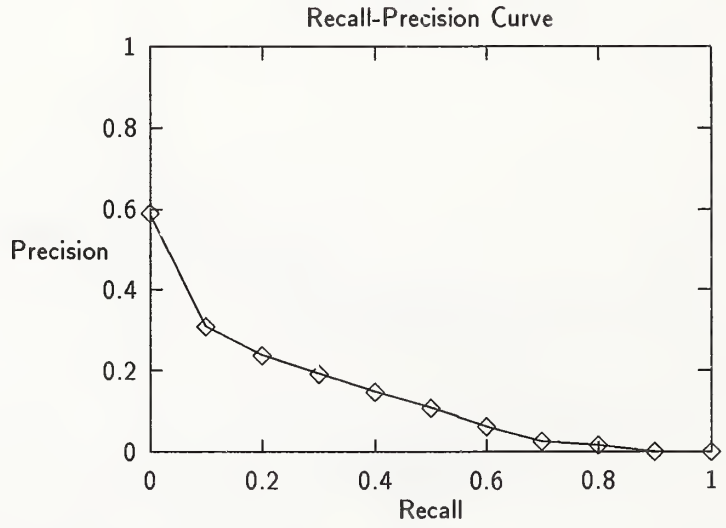
Average precision over all relevant docs	
non-interpolated	0.1307

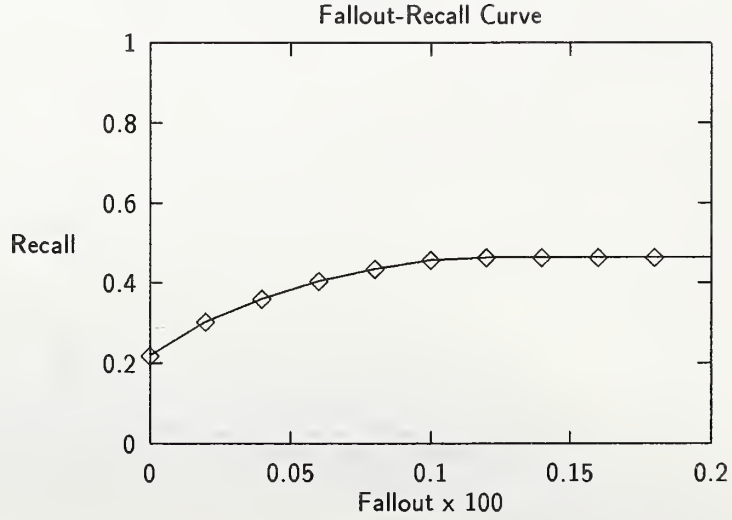
Document Level Averages	
	Precision
At 5 docs	0.3720
At 10 docs	0.3340
At 15 docs	0.3200
At 20 docs	0.3210
At 30 docs	0.3060
At 100 docs	0.2664
At 200 docs	0.2092
At 500 docs	0.1406
At 1000 docs	0.0951

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1937

Fallout Level Averages	
Fallout	Recall
0.00000	0.2200
0.00020	0.3034
0.00040	0.3601
0.00060	0.4050
0.00080	0.4350
0.00100	0.4576
0.00120	0.4642
0.00140	0.4642
0.00160	0.4642
0.00180	0.4642
0.00200	0.4642







Summary Statistics	
Run Number	Isiasm–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	7869

Recall Level Averages	
Recall	Precision
0.00	0.7127
0.10	0.5122
0.20	0.4698
0.30	0.4247
0.40	0.3753
0.50	0.3379
0.60	0.2758
0.70	0.2073
0.80	0.1331
0.90	0.0443
1.00	0.0000

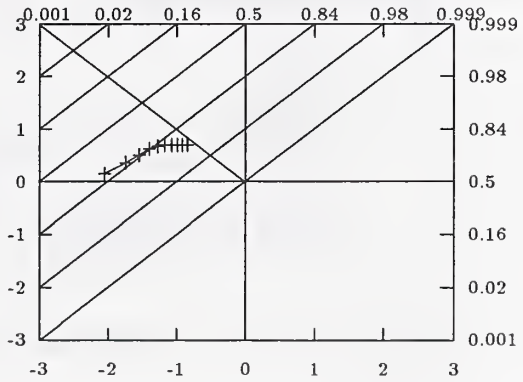
Average precision over all relevant docs	
non-interpolated	0.3018

Document Level Averages	
	Precision
At 5 docs	0.5240
At 10 docs	0.5020
At 15 docs	0.5027
At 20 docs	0.4950
At 30 docs	0.4873
At 100 docs	0.4306
At 200 docs	0.3716
At 500 docs	0.2469
At 1000 docs	0.1574

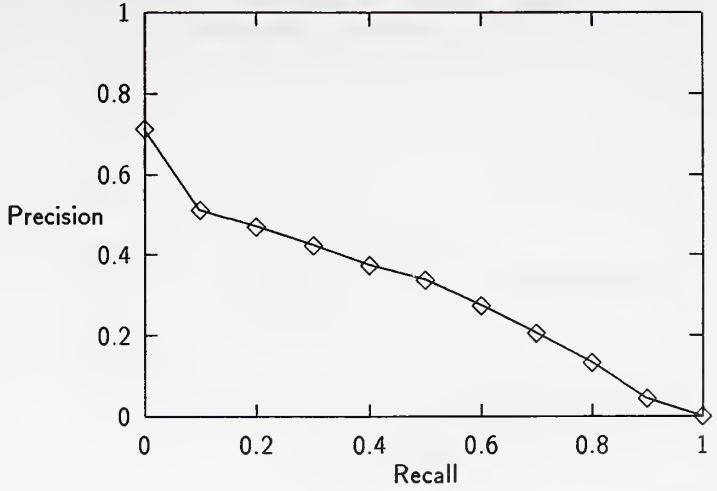
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3580

Fallout Level Averages	
Fallout	Recall
0.00000	0.4272
0.00020	0.5560
0.00040	0.6362
0.00060	0.6927
0.00080	0.7323
0.00100	0.7493
0.00120	0.7540
0.00140	0.7540
0.00160	0.7540
0.00180	0.7540
0.00200	0.7540

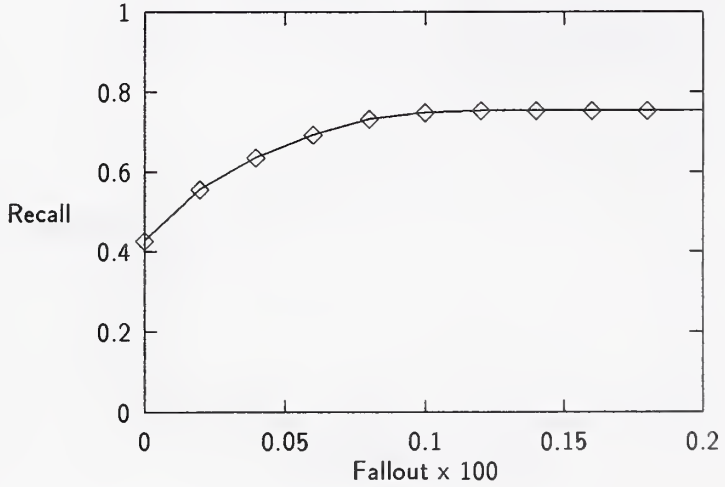
NORMAL DEVIATE - Fallout-Recall



Recall-Precision Curve



Fallout-Recall Curve



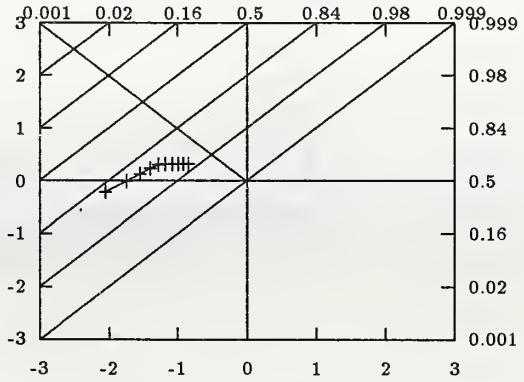
Summary Statistics	
Run Number	TMC8–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	49998
Relevant:	10785
Rel_ret:	6497

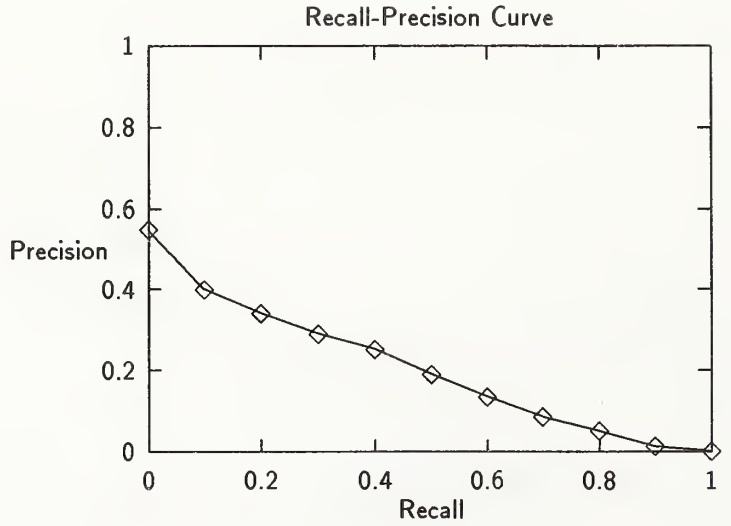
Recall Level Averages	
Recall	Precision
0.00	0.5482
0.10	0.4005
0.20	0.3411
0.30	0.2917
0.40	0.2525
0.50	0.1914
0.60	0.1347
0.70	0.0855
0.80	0.0509
0.90	0.0125
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.1939

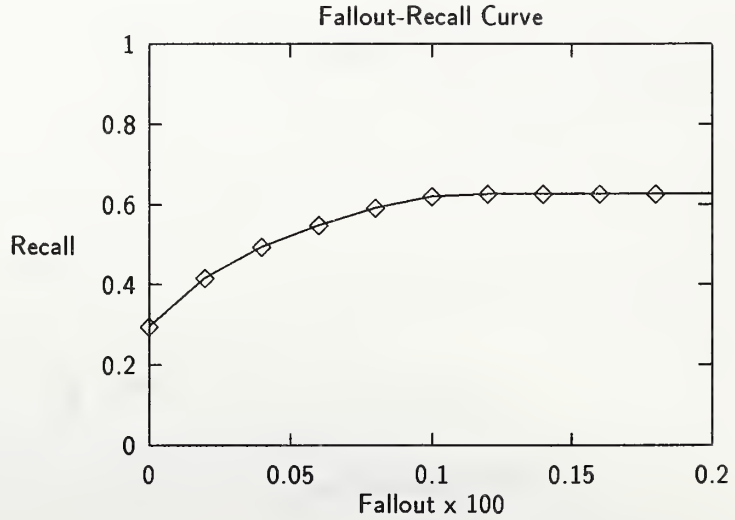
Document Level Averages	
	Precision
At 5 docs	0.3520
At 10 docs	0.3680
At 15 docs	0.3733
At 20 docs	0.3780
At 30 docs	0.3820
At 100 docs	0.3342
At 200 docs	0.2835
At 500 docs	0.1942
At 1000 docs	0.1299
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2612

Fallout Level Averages	
Fallout	Recall
0.00000	0.2962
0.00020	0.4158
0.00040	0.4942
0.00060	0.5490
0.00080	0.5917
0.00100	0.6197
0.00120	0.6260
0.00140	0.6260
0.00160	0.6260
0.00180	0.6260
0.00200	0.6260

NORMAL DEVIATE - Fallout-Recall







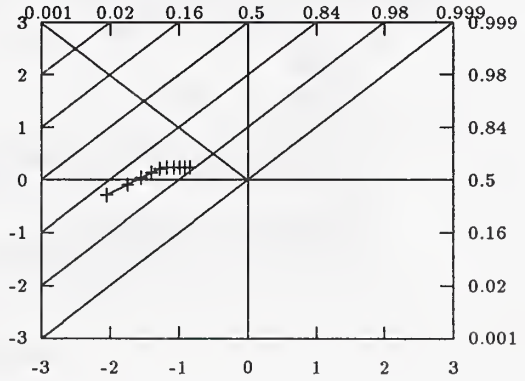
Summary Statistics	
Run Number	TMC9–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	49998
Relevant:	10785
Rel_ret:	6187

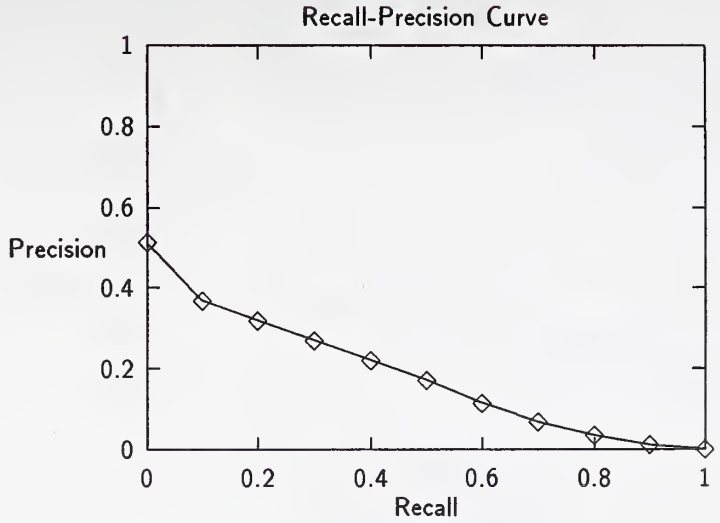
Recall Level Averages	
Recall	Precision
0.00	0.5141
0.10	0.3692
0.20	0.3195
0.30	0.2704
0.40	0.2203
0.50	0.1713
0.60	0.1144
0.70	0.0682
0.80	0.0357
0.90	0.0104
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.1736

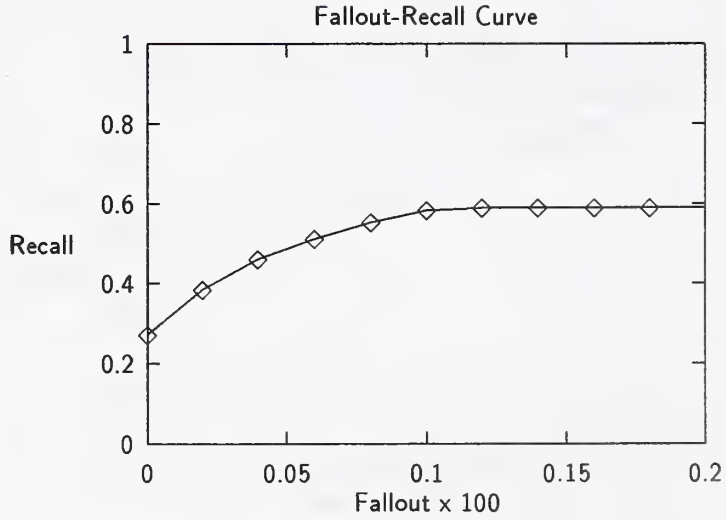
Document Level Averages	
	Precision
At 5 docs	0.2800
At 10 docs	0.3180
At 15 docs	0.3320
At 20 docs	0.3450
At 30 docs	0.3507
At 100 docs	0.3144
At 200 docs	0.2661
At 500 docs	0.1840
At 1000 docs	0.1237
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2436

Fallout Level Averages	
Fallout	Recall
0.00000	0.2725
0.00020	0.3846
0.00040	0.4610
0.00060	0.5126
0.00080	0.5540
0.00100	0.5827
0.00120	0.5904
0.00140	0.5904
0.00160	0.5904
0.00180	0.5904
0.00200	0.5904

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	nyuir1–category B, automatic
Num of Queries	50

Total number of documents over all queries	
Retrieved:	49884
Relevant:	3929
Rel_ret:	2983

Recall Level Averages	
Recall	Precision
0.00	0.7013
0.10	0.4874
0.20	0.4326
0.30	0.3531
0.40	0.3076
0.50	0.2637
0.60	0.2175
0.70	0.1617
0.80	0.1176
0.90	0.0684
1.00	0.0102

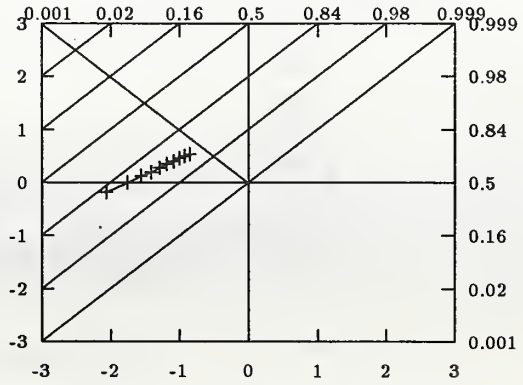
Average precision over all relevant docs	
non-interpolated	0.2649

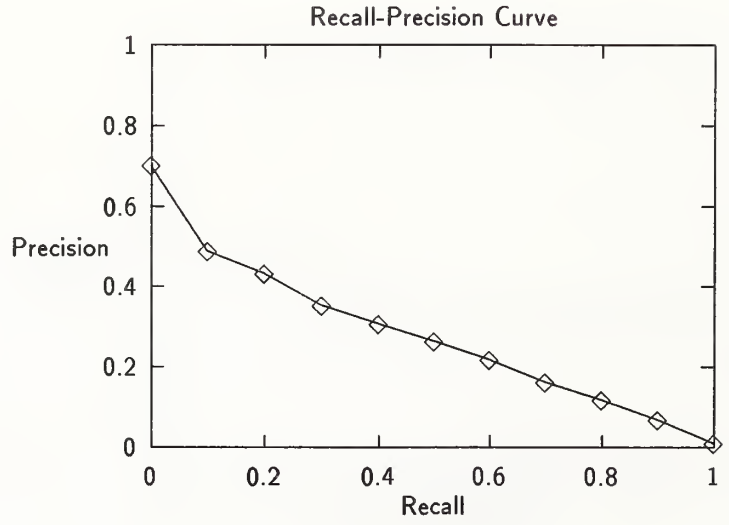
Document Level Averages	
	Precision
At 5 docs	0.4920
At 10 docs	0.4420
At 15 docs	0.4240
At 20 docs	0.4050
At 30 docs	0.3640
At 100 docs	0.2720
At 200 docs	0.1886
At 500 docs	0.1026
At 1000 docs	0.0597

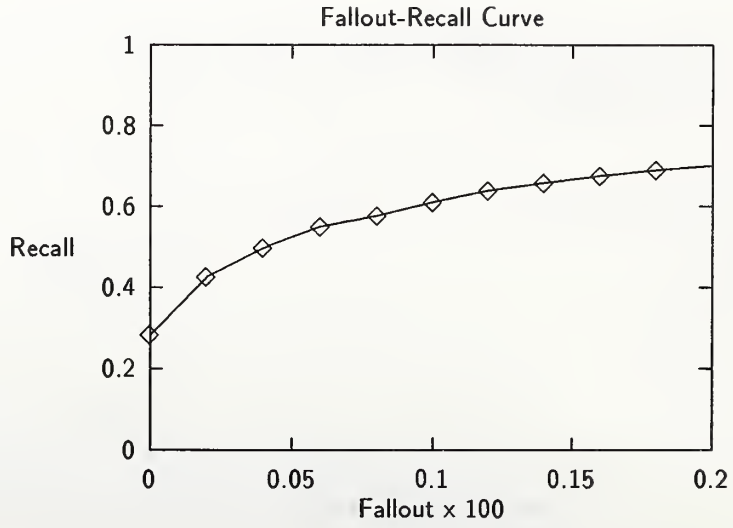
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3003

Fallout Level Averages	
Fallout	Recall
0.00000	0.2825
0.00020	0.4257
0.00040	0.4979
0.00060	0.5494
0.00080	0.5768
0.00100	0.6112
0.00120	0.6395
0.00140	0.6574
0.00160	0.6757
0.00180	0.6905
0.00200	0.7014

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	nyuir2–category B, automatic
Num of Queries	50

Total number of documents over all queries	
Retrieved:	49876
Relevant:	3929
Rel_ret:	3274

Recall Level Averages	
Recall	Precision
0.00	0.7528
0.10	0.5567
0.20	0.4721
0.30	0.4060
0.40	0.3617
0.50	0.3135
0.60	0.2703
0.70	0.2231
0.80	0.1667
0.90	0.0915
1.00	0.0154

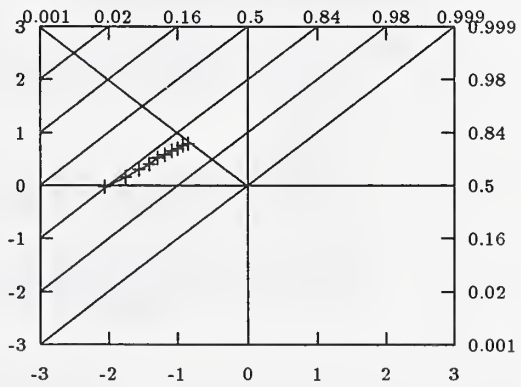
Average precision over all relevant docs	
non-interpolated	0.3111

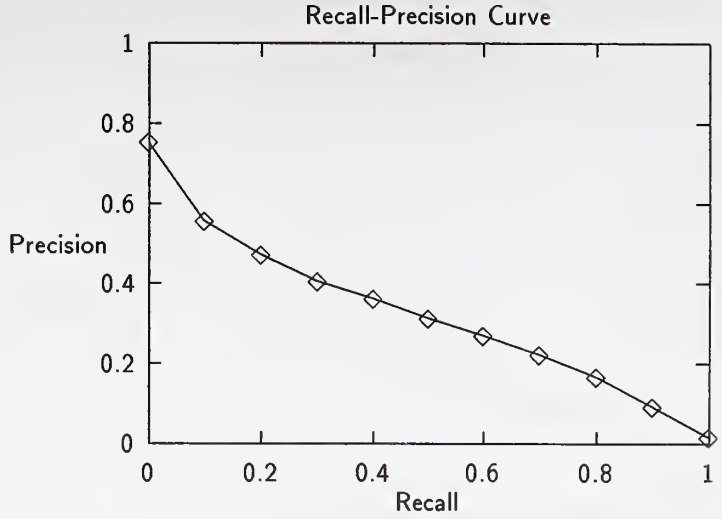
Document Level Averages	
	Precision
At 5 docs	0.5360
At 10 docs	0.4880
At 15 docs	0.4693
At 20 docs	0.4390
At 30 docs	0.4067
At 100 docs	0.3094
At 200 docs	0.2139
At 500 docs	0.1137
At 1000 docs	0.0655

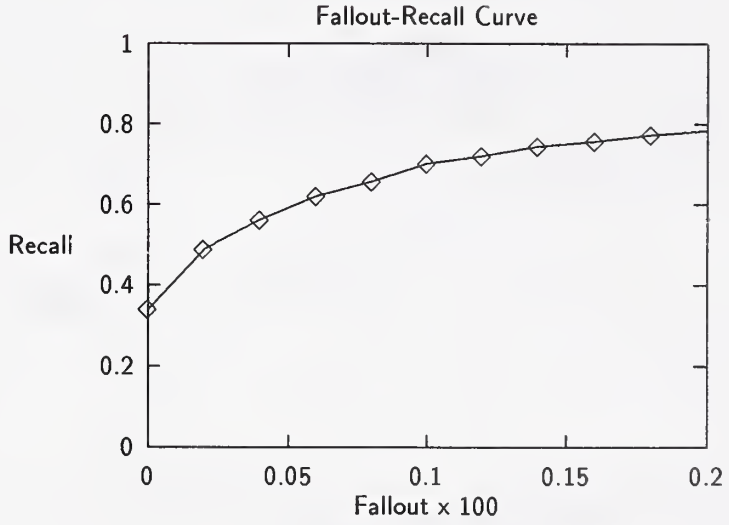
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3320

Fallout Level Averages	
Fallout	Recall
0.00000	0.3407
0.00020	0.4886
0.00040	0.5620
0.00060	0.6193
0.00080	0.6569
0.00100	0.7012
0.00120	0.7202
0.00140	0.7435
0.00160	0.7572
0.00180	0.7732
0.00200	0.7842

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	nyuir3–category B, automatic
Num of Queries	50

Total number of documents over all queries	
Retrieved:	49877
Relevant:	3929
Rel_ret:	3281

Recall Level Averages	
Recall	Precision
0.00	0.7528
0.10	0.5574
0.20	0.4724
0.30	0.4076
0.40	0.3621
0.50	0.3142
0.60	0.2711
0.70	0.2237
0.80	0.1697
0.90	0.0916
1.00	0.0160

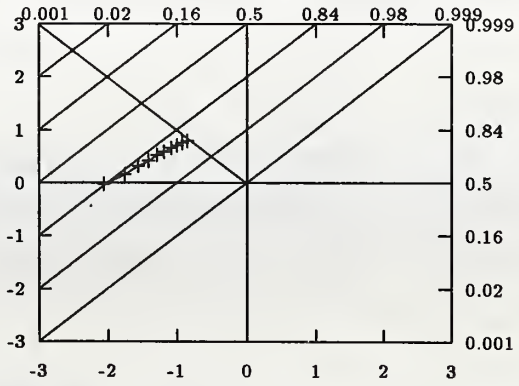
Average precision over all relevant docs	
non-interpolated	0.3118

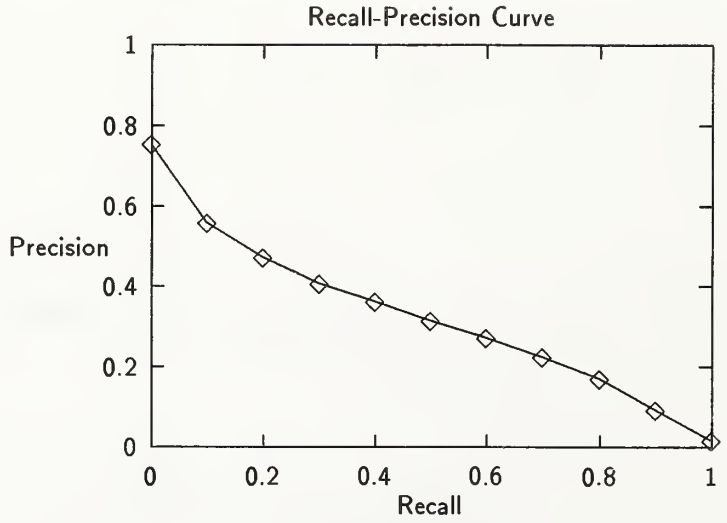
Document Level Averages	
	Precision
At 5 docs	0.5360
At 10 docs	0.4880
At 15 docs	0.4707
At 20 docs	0.4410
At 30 docs	0.4080
At 100 docs	0.3094
At 200 docs	0.2140
At 500 docs	0.1140
At 1000 docs	0.0656

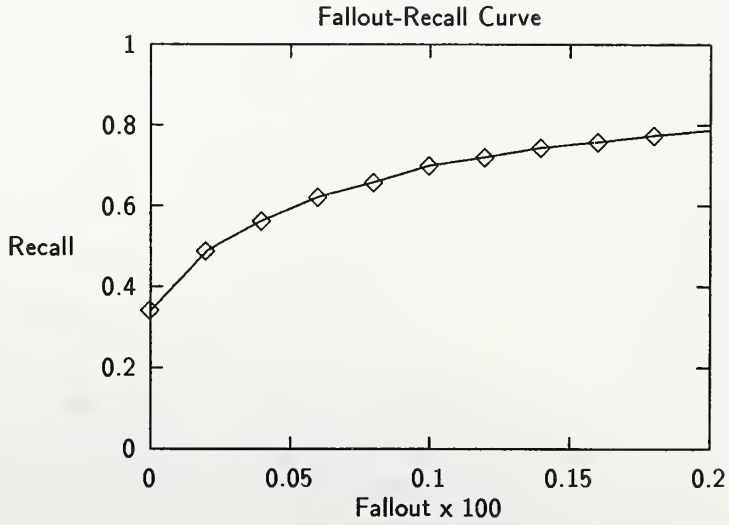
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3321

Fallout Level Averages	
Fallout	Recall
0.00000	0.3412
0.00020	0.4885
0.00040	0.5629
0.00060	0.6220
0.00080	0.6586
0.00100	0.6991
0.00120	0.7206
0.00140	0.7445
0.00160	0.7581
0.00180	0.7744
0.00200	0.7866

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	CLARTA–full, automatic
Num of Queries	50

Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	8109

Recall Level Averages	
Recall	Precision
0.00	0.7587
0.10	0.5676
0.20	0.5038
0.30	0.4489
0.40	0.3938
0.50	0.3455
0.60	0.2806
0.70	0.2172
0.80	0.1463
0.90	0.0827
1.00	0.0070

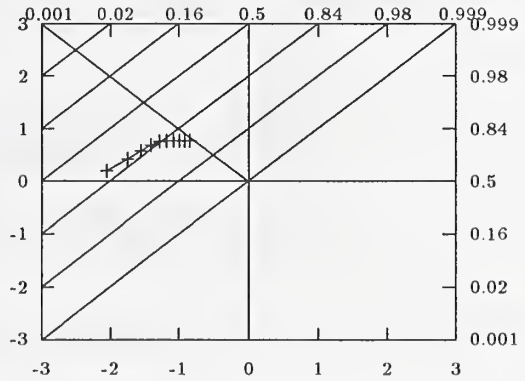
Average precision over all relevant docs	
non-interpolated	0.3264

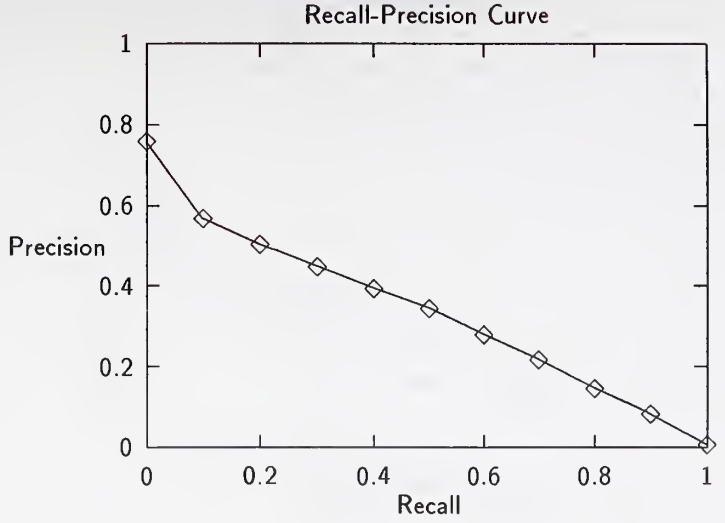
Document Level Averages	
	Precision
At 5 docs	0.5800
At 10 docs	0.5680
At 15 docs	0.5547
At 20 docs	0.5490
At 30 docs	0.5253
At 100 docs	0.4644
At 200 docs	0.3874
At 500 docs	0.2542
At 1000 docs	0.1622

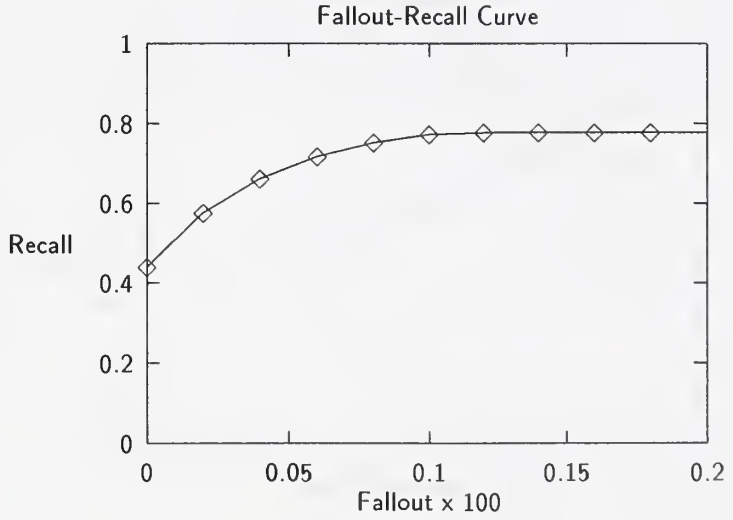
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3645

Fallout Level Averages	
Fallout	Recall
0.00000	0.4407
0.00020	0.5756
0.00040	0.6613
0.00060	0.7169
0.00080	0.7520
0.00100	0.7732
0.00120	0.7767
0.00140	0.7767
0.00160	0.7767
0.00180	0.7767
0.00200	0.7767

NORMAL DEVIATE - Fallout-Recall







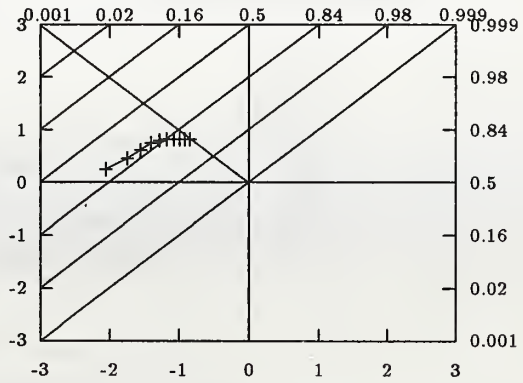
Summary Statistics	
Run Number	CLARTM–full, manual
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	8229

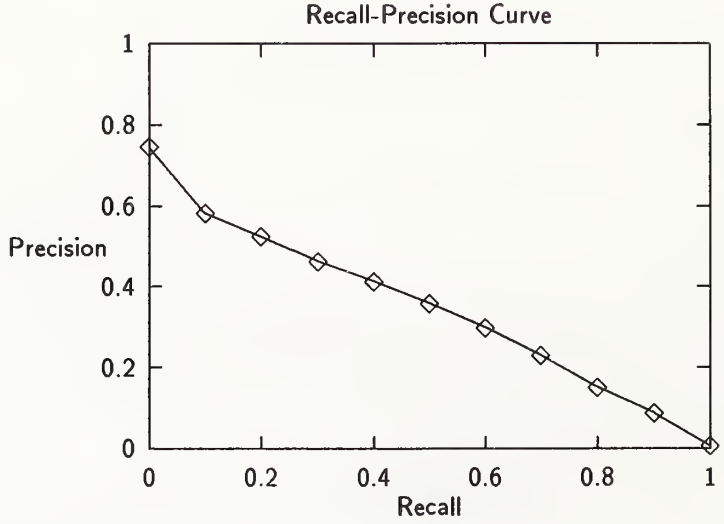
Recall Level Averages	
Recall	Precision
0.00	0.7455
0.10	0.5811
0.20	0.5240
0.30	0.4622
0.40	0.4135
0.50	0.3593
0.60	0.2983
0.70	0.2312
0.80	0.1516
0.90	0.0874
1.00	0.0062
Average precision over all relevant docs	
non-interpolated	0.3383

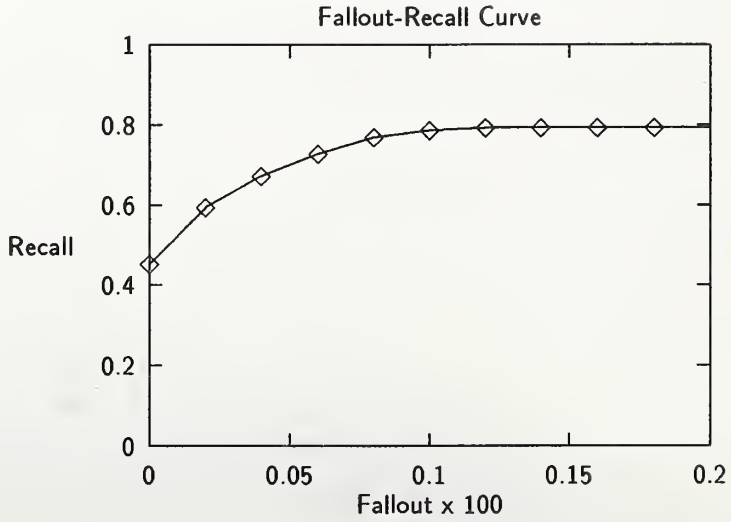
Document Level Averages	
	Precision
At 5 docs	0.5840
At 10 docs	0.5740
At 15 docs	0.5640
At 20 docs	0.5590
At 30 docs	0.5433
At 100 docs	0.4846
At 200 docs	0.3975
At 500 docs	0.2601
At 1000 docs	0.1646
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3741

Fallout Level Averages	
Fallout	Recall
0.00000	0.4527
0.00020	0.5940
0.00040	0.6730
0.00060	0.7277
0.00080	0.7696
0.00100	0.7868
0.00120	0.7932
0.00140	0.7932
0.00160	0.7932
0.00180	0.7932
0.00200	0.7932

NORMAL DEVIATE - Fallout-Recall







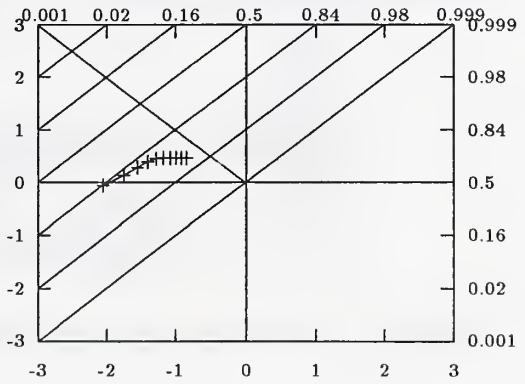
Summary Statistics	
Run Number	schau1–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	7081

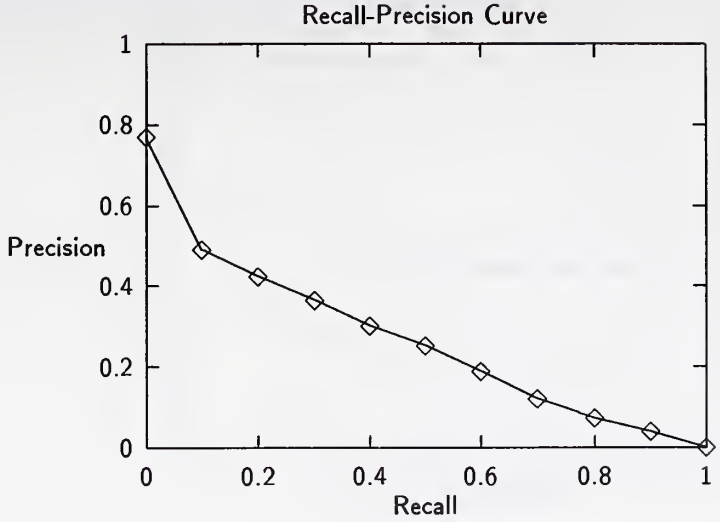
Recall Level Averages	
Recall	Precision
0.00	0.7692
0.10	0.4912
0.20	0.4237
0.30	0.3649
0.40	0.3025
0.50	0.2529
0.60	0.1887
0.70	0.1213
0.80	0.0727
0.90	0.0403
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.2517

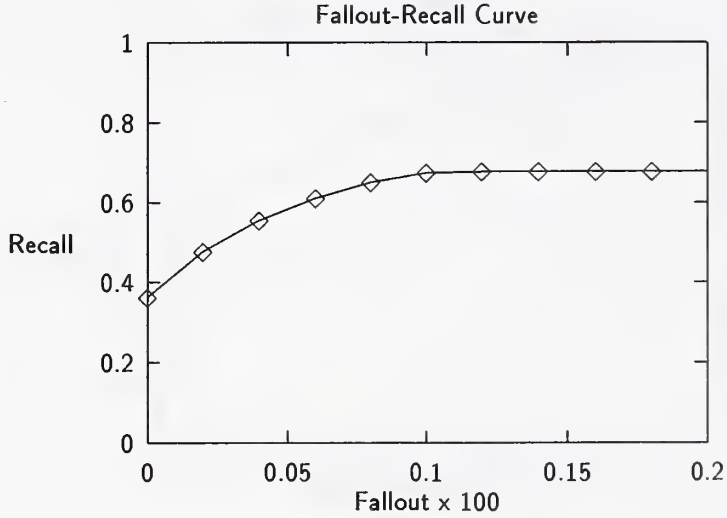
Document Level Averages	
	Precision
At 5 docs	0.5360
At 10 docs	0.4960
At 15 docs	0.4800
At 20 docs	0.4720
At 30 docs	0.4513
At 100 docs	0.4000
At 200 docs	0.3302
At 500 docs	0.2158
At 1000 docs	0.1416
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3148

Fallout Level Averages	
Fallout	Recall
0.00000	0.3620
0.00020	0.4763
0.00040	0.5547
0.00060	0.6102
0.00080	0.6507
0.00100	0.6743
0.00120	0.6771
0.00140	0.6771
0.00160	0.6771
0.00180	0.6771
0.00200	0.6771

NORMAL DEVIATE - Fallout-Recall





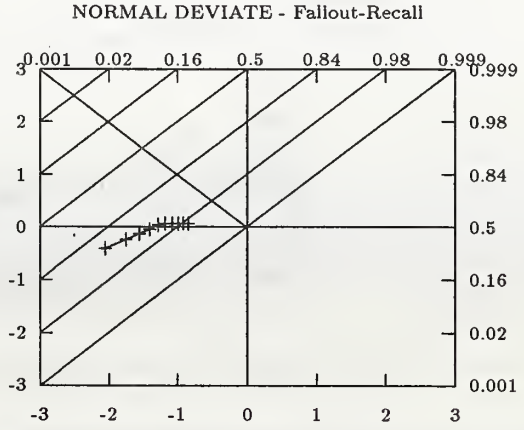


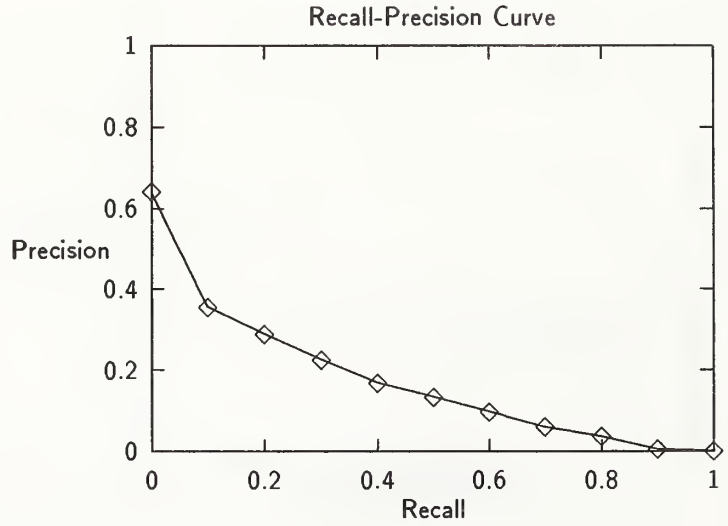
Summary Statistics	
Run Number	erima1–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	48809
Relevant:	10785
Rel_ret:	5349

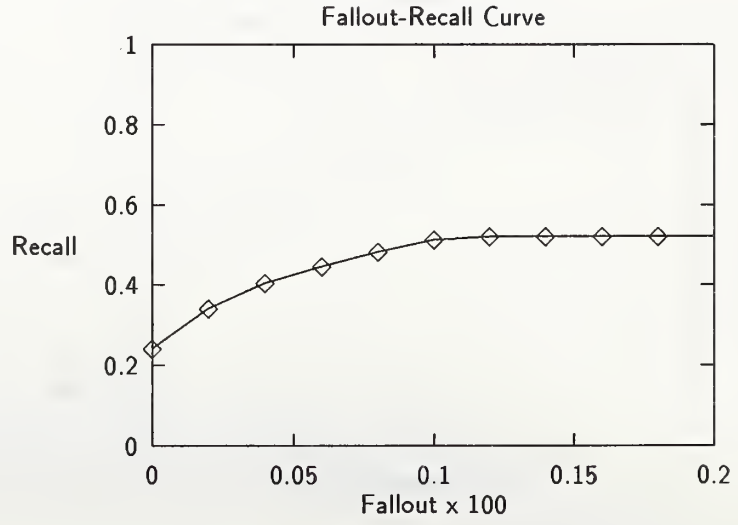
Recall Level Averages	
Recall	Precision
0.00	0.6409
0.10	0.3564
0.20	0.2893
0.30	0.2257
0.40	0.1693
0.50	0.1332
0.60	0.0971
0.70	0.0610
0.80	0.0370
0.90	0.0048
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.1589

Document Level Averages	
	Precision
At 5 docs	0.4360
At 10 docs	0.3960
At 15 docs	0.3827
At 20 docs	0.3810
At 30 docs	0.3660
At 100 docs	0.3016
At 200 docs	0.2378
At 500 docs	0.1598
At 1000 docs	0.1070
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2179

Fallout Level Averages	
Fallout	Recall
0.00000	0.2423
0.00020	0.3411
0.00040	0.4053
0.00060	0.4460
0.00080	0.4827
0.00100	0.5138
0.00120	0.5221
0.00140	0.5221
0.00160	0.5221
0.00180	0.5221
0.00200	0.5221







Summary Statistics	
Run Number	erima2–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	47943
Relevant:	10785
Rel_ret:	5975

Recall Level Averages	
Recall	Precision
0.00	0.7096
0.10	0.4059
0.20	0.3287
0.30	0.2723
0.40	0.2276
0.50	0.1698
0.60	0.1302
0.70	0.0707
0.80	0.0480
0.90	0.0011
1.00	0.0000

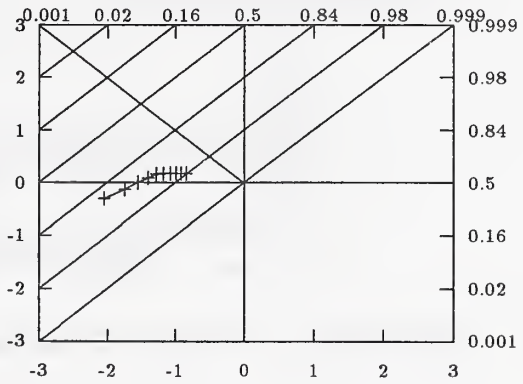
Average precision over all relevant docs	
non-interpolated	0.1885

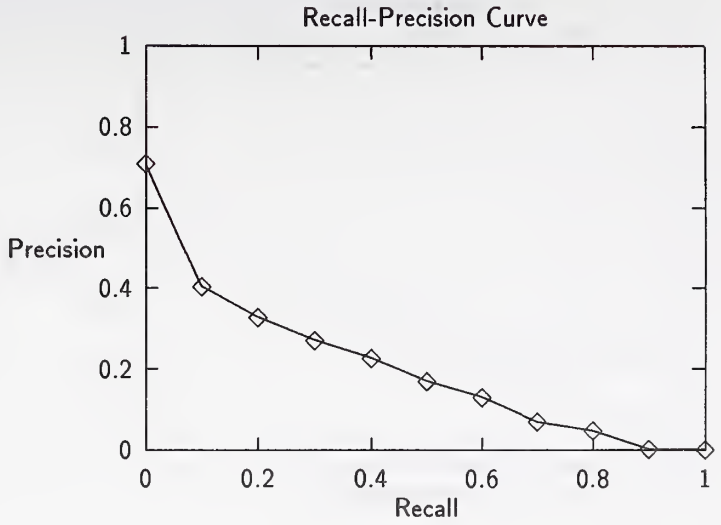
Document Level Averages	
	Precision
At 5 docs	0.4760
At 10 docs	0.4480
At 15 docs	0.4320
At 20 docs	0.4180
At 30 docs	0.3993
At 100 docs	0.3426
At 200 docs	0.2707
At 500 docs	0.1799
At 1000 docs	0.1195

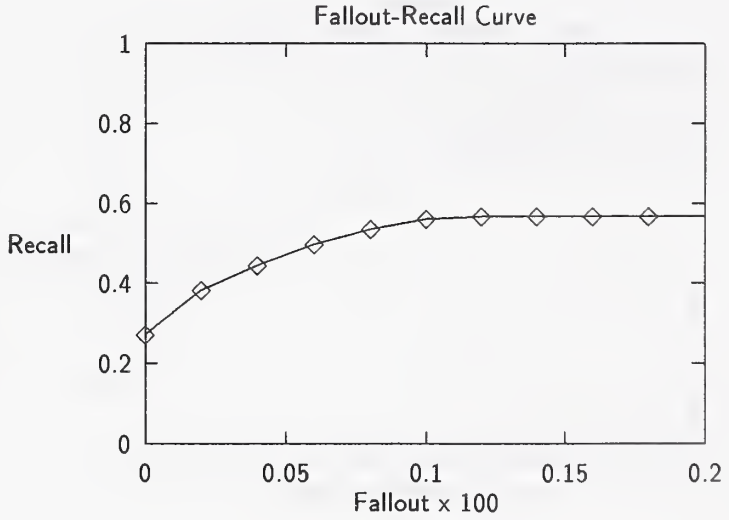
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2494

Fallout Level Averages	
Fallout	Recall
0.00000	0.2717
0.00020	0.3824
0.00040	0.4449
0.00060	0.4979
0.00080	0.5353
0.00100	0.5611
0.00120	0.5678
0.00140	0.5678
0.00160	0.5678
0.00180	0.5678
0.00200	0.5678

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	citri1–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	7439

Recall Level Averages	
Recall	Precision
0.00	0.8023
0.10	0.5298
0.20	0.4570
0.30	0.4027
0.40	0.3482
0.50	0.2951
0.60	0.2271
0.70	0.1514
0.80	0.0887
0.90	0.0379
1.00	0.0000

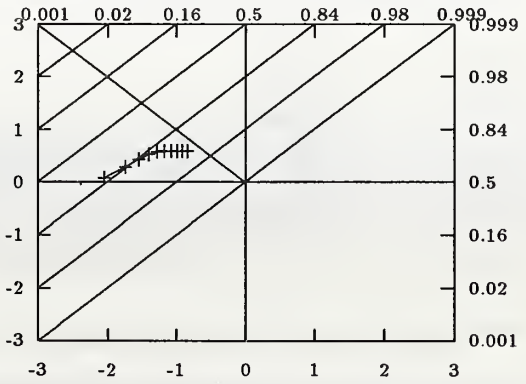
Average precision over all relevant docs	
non-interpolated	0.2804

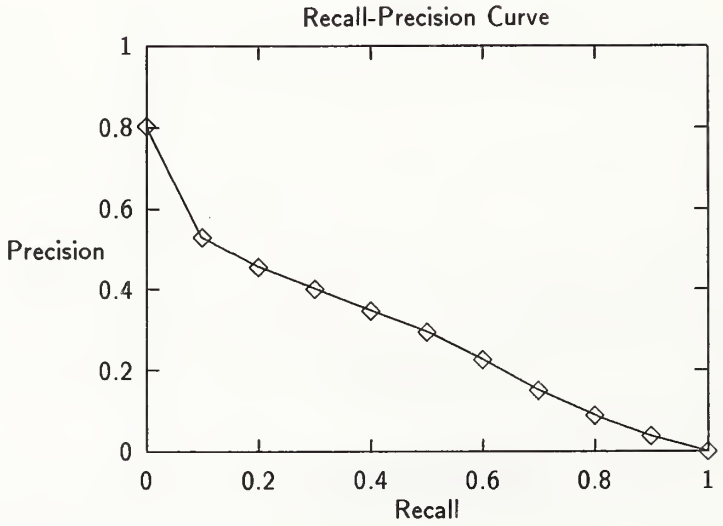
Document Level Averages	
	Precision
At 5 docs	0.5680
At 10 docs	0.5320
At 15 docs	0.5173
At 20 docs	0.5140
At 30 docs	0.4913
At 100 docs	0.4224
At 200 docs	0.3541
At 500 docs	0.2340
At 1000 docs	0.1488

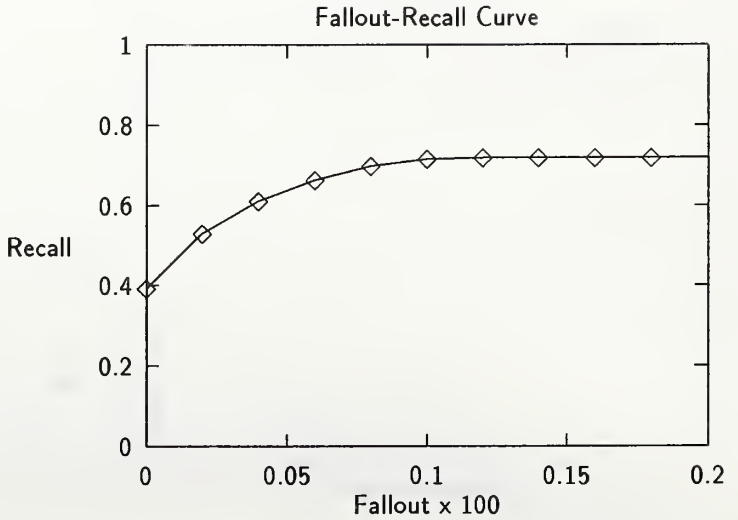
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3359

Fallout Level Averages	
Fallout	Recall
0.00000	0.3934
0.00020	0.5301
0.00040	0.6112
0.00060	0.6630
0.00080	0.6972
0.00100	0.7158
0.00120	0.7192
0.00140	0.7192
0.00160	0.7192
0.00180	0.7192
0.00200	0.7192

NORMAL DEVIATE - Fallout-Recall







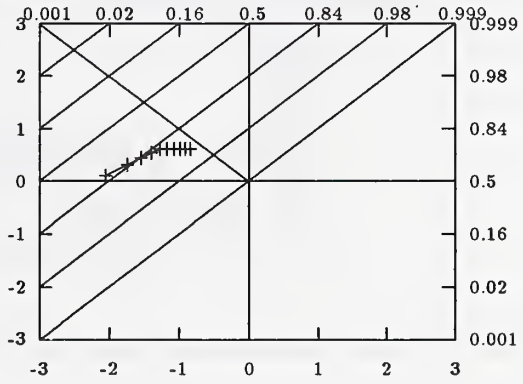
Summary Statistics	
Run Number	citri2–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	7495

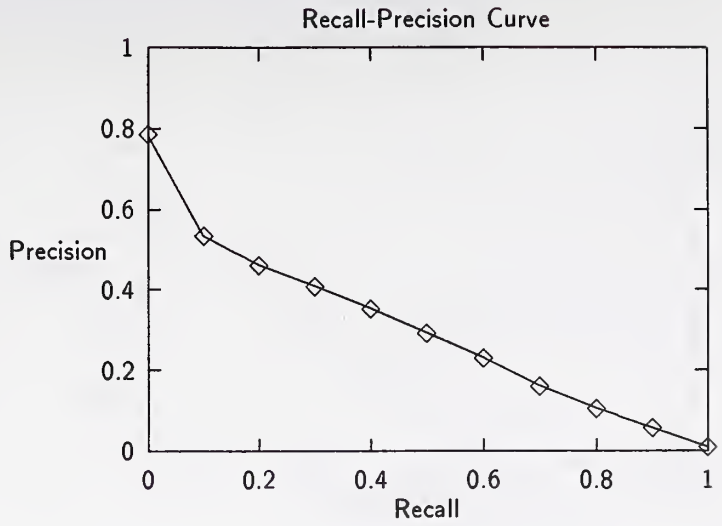
Recall Level Averages	
Recall	Precision
0.00	0.7856
0.10	0.5335
0.20	0.4611
0.30	0.4086
0.40	0.3526
0.50	0.2932
0.60	0.2306
0.70	0.1613
0.80	0.1053
0.90	0.0570
1.00	0.0097
Average precision over all relevant docs	
non-interpolated	0.2874

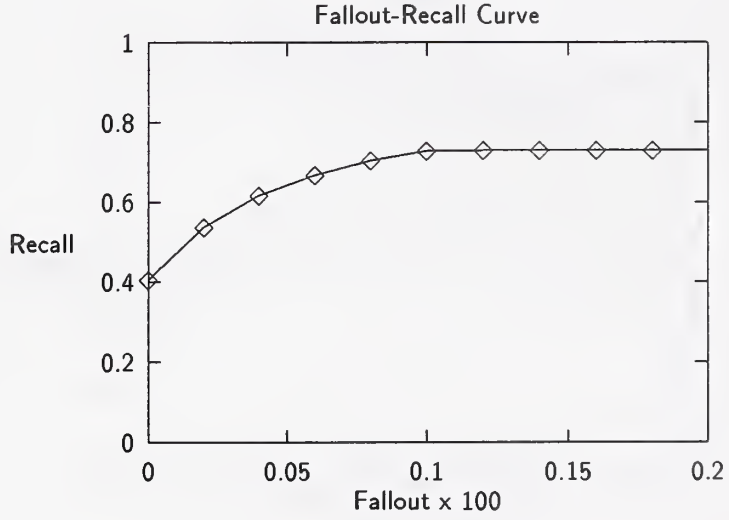
Document Level Averages	
	Precision
At 5 docs	0.5600
At 10 docs	0.5280
At 15 docs	0.5133
At 20 docs	0.5000
At 30 docs	0.4900
At 100 docs	0.4354
At 200 docs	0.3606
At 500 docs	0.2358
At 1000 docs	0.1499
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3388

Fallout Level Averages	
Fallout	Recall
0.00000	0.4050
0.00020	0.5373
0.00040	0.6176
0.00060	0.6684
0.00080	0.7040
0.00100	0.7276
0.00120	0.7291
0.00140	0.7291
0.00160	0.7291
0.00180	0.7291
0.00200	0.7291

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	gecrd2–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	6080

Recall Level Averages	
Recall	Precision
0.00	0.7047
0.10	0.4433
0.20	0.3753
0.30	0.3112
0.40	0.2503
0.50	0.2049
0.60	0.1496
0.70	0.0999
0.80	0.0690
0.90	0.0381
1.00	0.0088

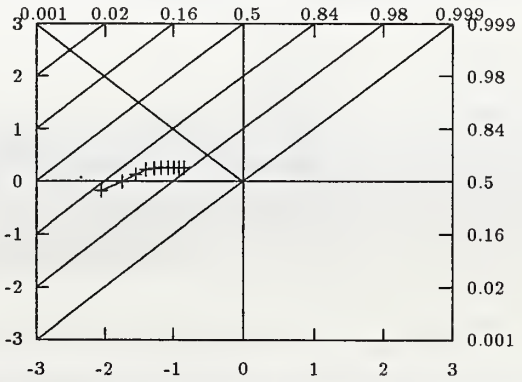
Average precision over all relevant docs	
non-interpolated	0.2183

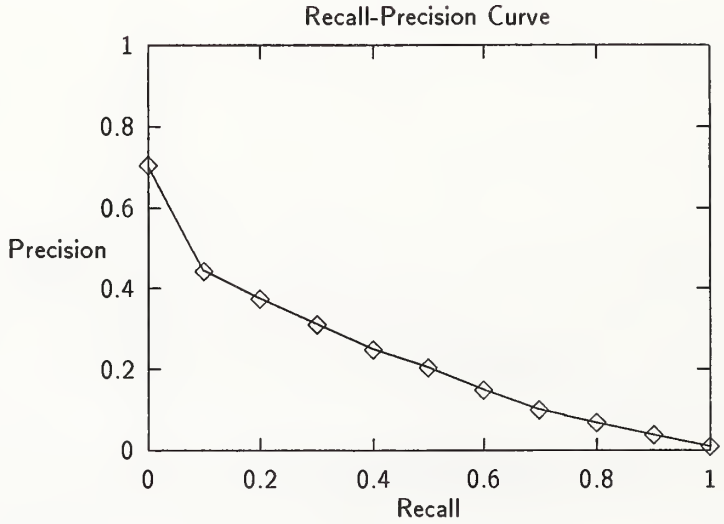
Document Level Averages	
	Precision
At 5 docs	0.4880
At 10 docs	0.4760
At 15 docs	0.4653
At 20 docs	0.4510
At 30 docs	0.4253
At 100 docs	0.3662
At 200 docs	0.2896
At 500 docs	0.1876
At 1000 docs	0.1216

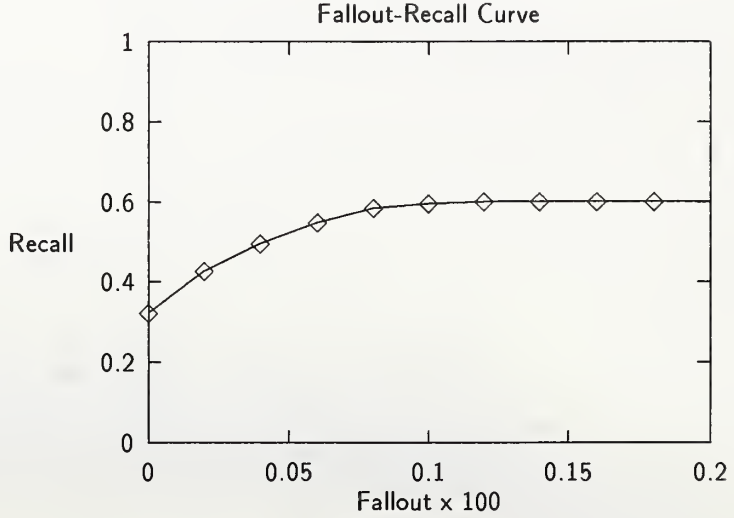
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2816

Fallout Level Averages	
Fallout	Recall
0.00000	0.3221
0.00020	0.4269
0.00040	0.4957
0.00060	0.5489
0.00080	0.5850
0.00100	0.5968
0.00120	0.6014
0.00140	0.6014
0.00160	0.6014
0.00180	0.6014
0.00200	0.6014

NORMAL DEVIATE - Fallout-Recall







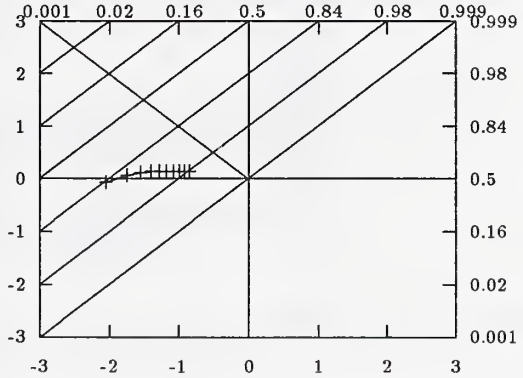
Summary Statistics	
Run Number	TOPIC2–full, manual
Num of Queries	50
Total number of documents over all queries	
Retrieved:	31624
Relevant:	10785
Rel_ret:	5614

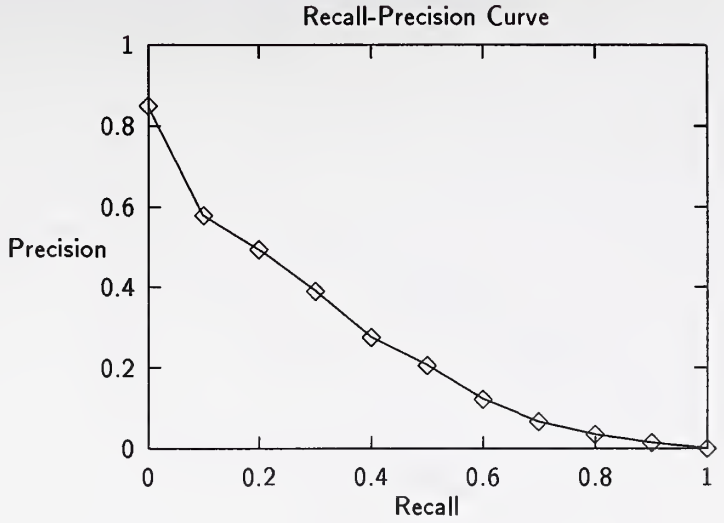
Recall Level Averages	
Recall	Precision
0.00	0.8497
0.10	0.5786
0.20	0.4943
0.30	0.3906
0.40	0.2774
0.50	0.2062
0.60	0.1221
0.70	0.0663
0.80	0.0353
0.90	0.0137
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.2464

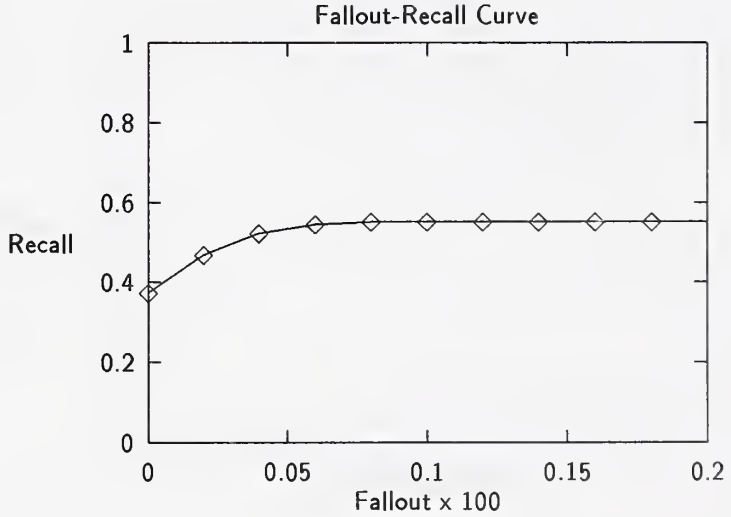
Document Level Averages	
	Precision
At 5 docs	0.6280
At 10 docs	0.5880
At 15 docs	0.5827
At 20 docs	0.5680
At 30 docs	0.5407
At 100 docs	0.4624
At 200 docs	0.3347
At 500 docs	0.2002
At 1000 docs	0.1123
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3211

Fallout Level Averages	
Fallout	Recall
0.00000	0.3744
0.00020	0.4689
0.00040	0.5232
0.00060	0.5446
0.00080	0.5512
0.00100	0.5512
0.00120	0.5512
0.00140	0.5512
0.00160	0.5512
0.00180	0.5512
0.00200	0.5512

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	siems2–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	8068

Recall Level Averages	
Recall	Precision
0.00	0.7964
0.10	0.5766
0.20	0.5267
0.30	0.4808
0.40	0.4244
0.50	0.3810
0.60	0.3088
0.70	0.2418
0.80	0.1416
0.90	0.0513
1.00	0.0000

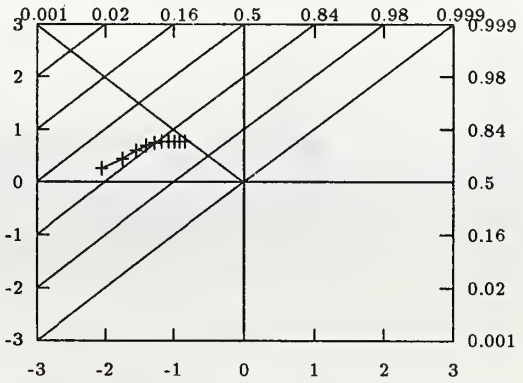
Average precision over all relevant docs	
non-interpolated	0.3408

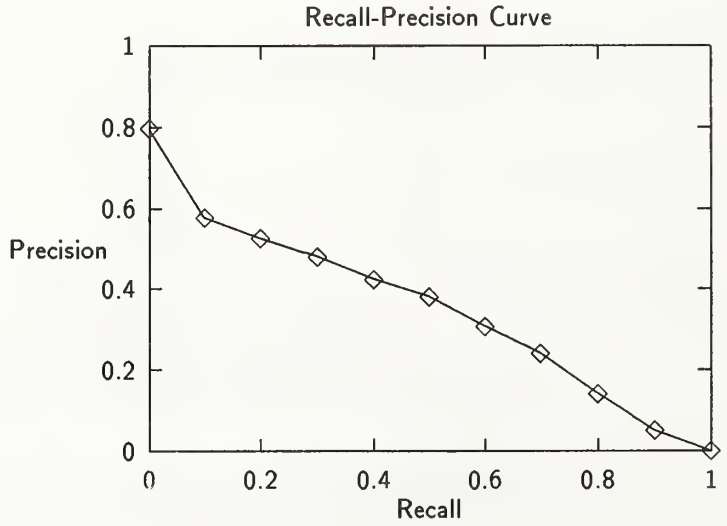
Document Level Averages	
	Precision
At 5 docs	0.6000
At 10 docs	0.5860
At 15 docs	0.5733
At 20 docs	0.5600
At 30 docs	0.5447
At 100 docs	0.4690
At 200 docs	0.4030
At 500 docs	0.2606
At 1000 docs	0.1614

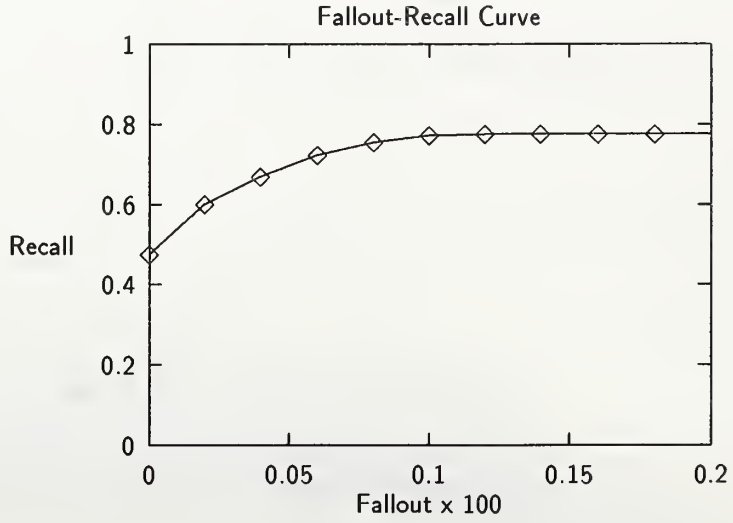
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3938

Fallout Level Averages	
Fallout	Recall
0.00000	0.4758
0.00020	0.6006
0.00040	0.6695
0.00060	0.7229
0.00080	0.7558
0.00100	0.7722
0.00120	0.7763
0.00140	0.7763
0.00160	0.7763
0.00180	0.7763
0.00200	0.7763

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	siems3–full, automatic
Num of Queries	50

Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	8065

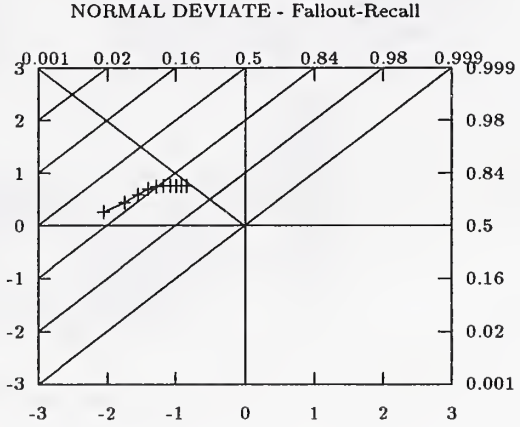
Recall Level Averages	
Recall	Precision
0.00	0.8164
0.10	0.5849
0.20	0.5262
0.30	0.4774
0.40	0.4262
0.50	0.3771
0.60	0.3111
0.70	0.2338
0.80	0.1362
0.90	0.0513
1.00	0.0000

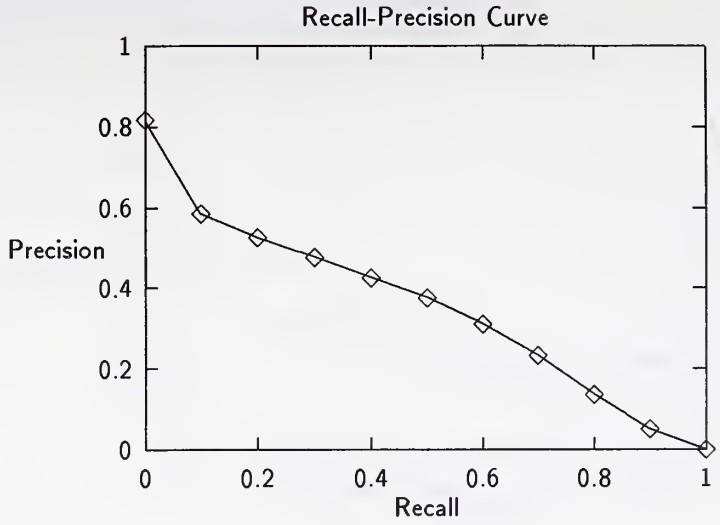
Average precision over all relevant docs	
non-interpolated	0.3397

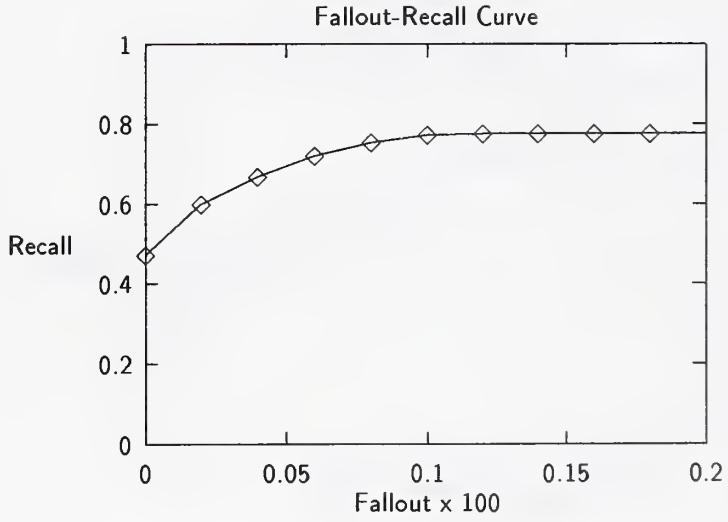
Document Level Averages	
	Precision
At 5 docs	0.5960
At 10 docs	0.5920
At 15 docs	0.5773
At 20 docs	0.5630
At 30 docs	0.5453
At 100 docs	0.4692
At 200 docs	0.3979
At 500 docs	0.2598
At 1000 docs	0.1613

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3931

Fallout Level Averages	
Fallout	Recall
0.00000	0.4714
0.00020	0.5999
0.00040	0.6680
0.00060	0.7207
0.00080	0.7543
0.00100	0.7720
0.00120	0.7757
0.00140	0.7757
0.00160	0.7757
0.00180	0.7757
0.00200	0.7757







Summary Statistics	
Run Number	pircs3–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	7456

Recall Level Averages	
Recall	Precision
0.00	0.7538
0.10	0.5529
0.20	0.4727
0.30	0.4088
0.40	0.3531
0.50	0.2994
0.60	0.2311
0.70	0.1677
0.80	0.1094
0.90	0.0551
1.00	0.0145

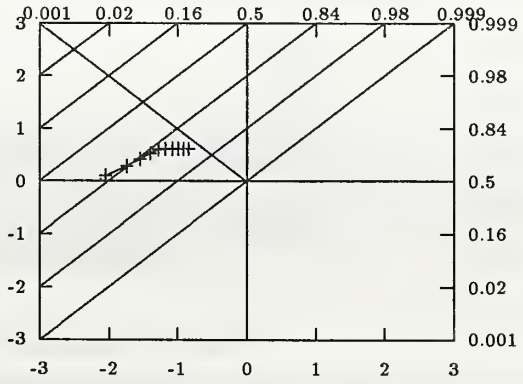
Average precision over all relevant docs	
non-interpolated	0.2949

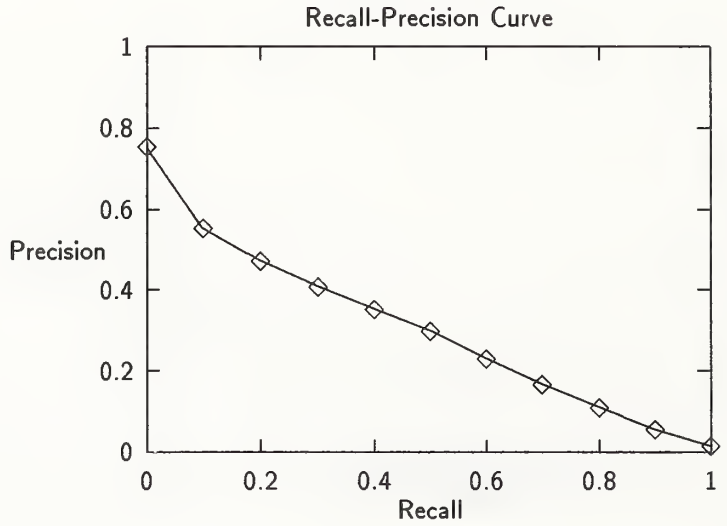
Document Level Averages	
	Precision
At 5 docs	0.5640
At 10 docs	0.5560
At 15 docs	0.5467
At 20 docs	0.5460
At 30 docs	0.5200
At 100 docs	0.4466
At 200 docs	0.3660
At 500 docs	0.2342
At 1000 docs	0.1491

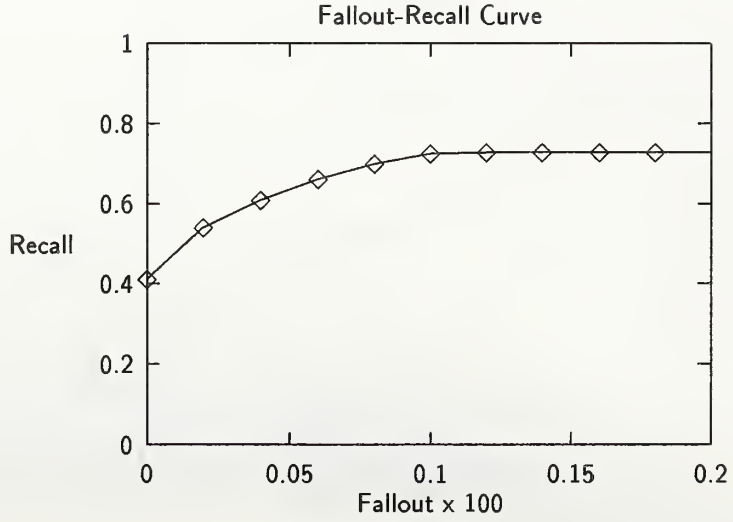
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3437

Fallout Level Averages	
Fallout	Recall
0.00000	0.4118
0.00020	0.5411
0.00040	0.6092
0.00060	0.6608
0.00080	0.6990
0.00100	0.7245
0.00120	0.7288
0.00140	0.7288
0.00160	0.7288
0.00180	0.7288
0.00200	0.7288

NORMAL DEVIATE - Fallout-Recall







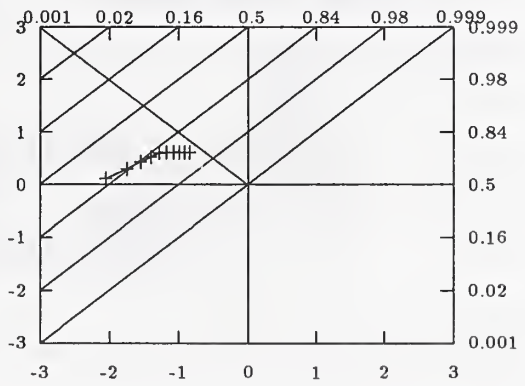
Summary Statistics	
Run Number	pircs4–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	7464

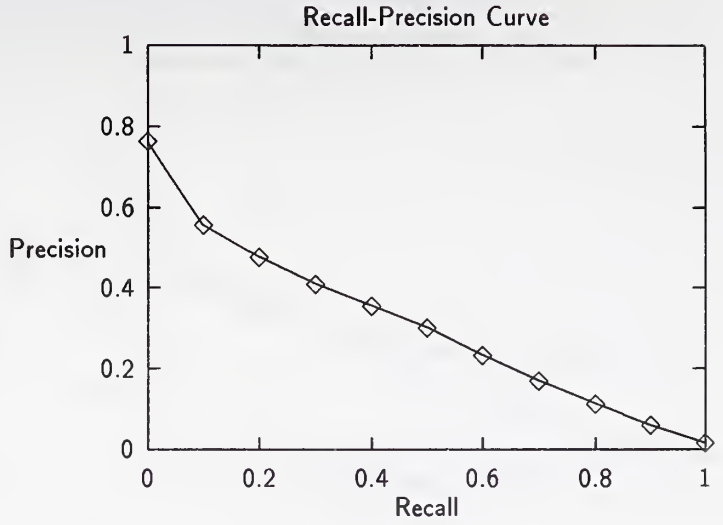
Recall Level Averages	
Recall	Precision
0.00	0.7627
0.10	0.5569
0.20	0.4773
0.30	0.4108
0.40	0.3554
0.50	0.3017
0.60	0.2343
0.70	0.1702
0.80	0.1128
0.90	0.0602
1.00	0.0154
Average precision over all relevant docs	
non-interpolated	0.2981

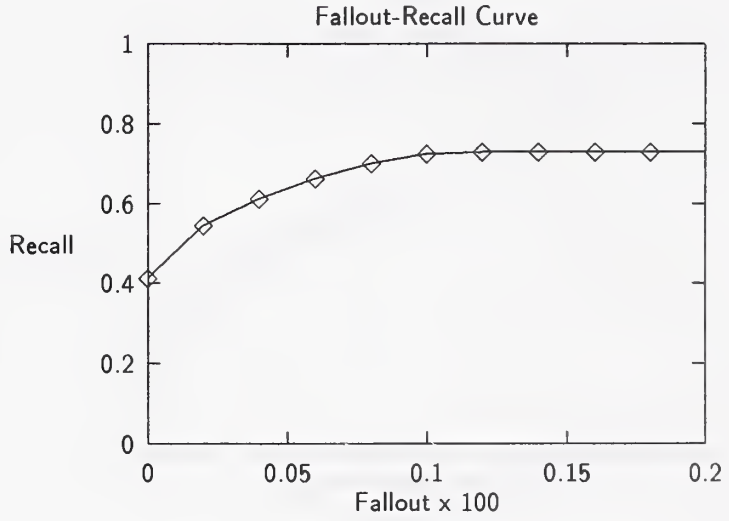
Document Level Averages	
	Precision
At 5 docs	0.5760
At 10 docs	0.5760
At 15 docs	0.5507
At 20 docs	0.5460
At 30 docs	0.5200
At 100 docs	0.4494
At 200 docs	0.3680
At 500 docs	0.2348
At 1000 docs	0.1493
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3467

Fallout Level Averages	
Fallout	Recall
0.00000	0.4140
0.00020	0.5449
0.00040	0.6128
0.00060	0.6637
0.00080	0.7008
0.00100	0.7244
0.00120	0.7293
0.00140	0.7293
0.00160	0.7293
0.00180	0.7293
0.00200	0.7293

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	VTcms2–full, manual
Num of Queries	50

Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	7676

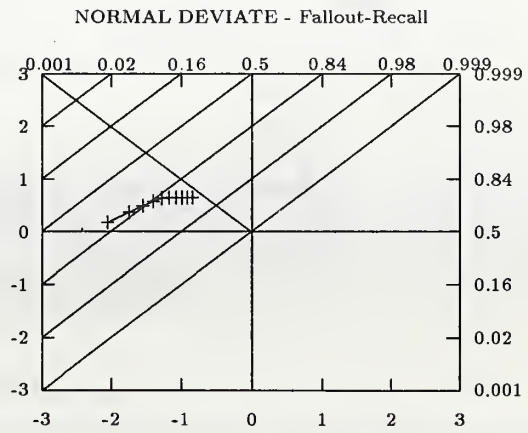
Recall Level Averages	
Recall	Precision
0.00	0.8693
0.10	0.6115
0.20	0.5409
0.30	0.4591
0.40	0.3834
0.50	0.3172
0.60	0.2451
0.70	0.1659
0.80	0.0909
0.90	0.0364
1.00	0.0009

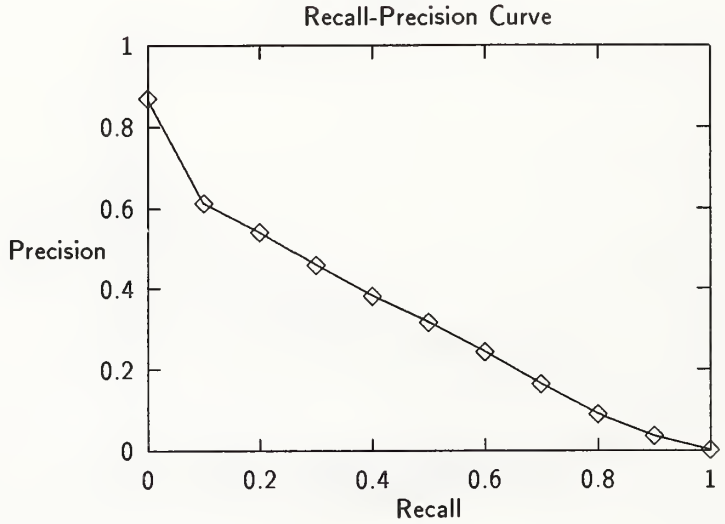
Average precision over all relevant docs	
non-interpolated	0.3200

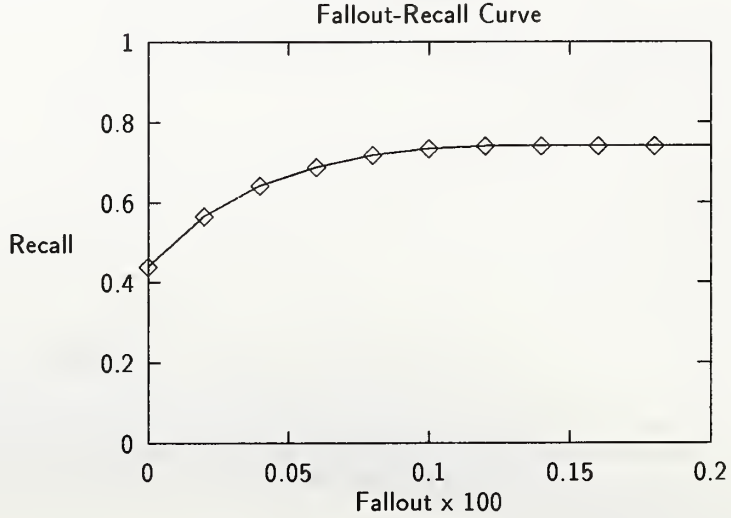
Document Level Averages	
	Precision
At 5 docs	0.6800
At 10 docs	0.6280
At 15 docs	0.6240
At 20 docs	0.6130
At 30 docs	0.5820
At 100 docs	0.4790
At 200 docs	0.3835
At 500 docs	0.2430
At 1000 docs	0.1535

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3708

Fallout Level Averages	
Fallout	Recall
0.00000	0.4410
0.00020	0.5666
0.00040	0.6420
0.00060	0.6877
0.00080	0.7179
0.00100	0.7347
0.00120	0.7414
0.00140	0.7414
0.00160	0.7414
0.00180	0.7414
0.00200	0.7414





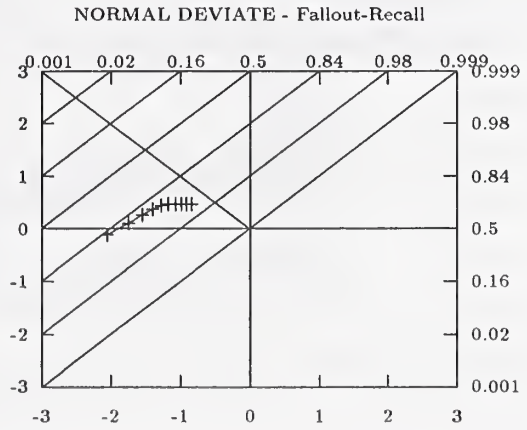


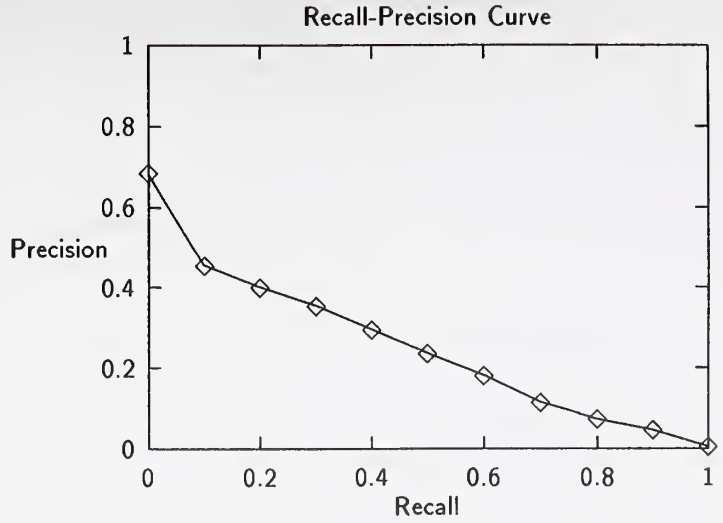
Summary Statistics	
Run Number	CnQst1–full, manual
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	6494

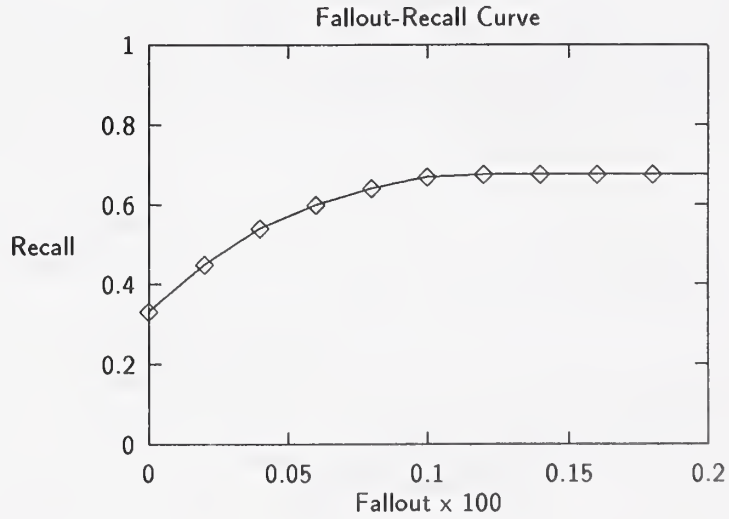
Recall Level Averages	
Recall	Precision
0.00	0.6831
0.10	0.4552
0.20	0.4010
0.30	0.3538
0.40	0.2957
0.50	0.2367
0.60	0.1807
0.70	0.1147
0.80	0.0725
0.90	0.0454
1.00	0.0054
Average precision over all relevant docs	
non-interpolated	0.2343

Document Level Averages	
	Precision
At 5 docs	0.4480
At 10 docs	0.4360
At 15 docs	0.4427
At 20 docs	0.4310
At 30 docs	0.4393
At 100 docs	0.3954
At 200 docs	0.2997
At 500 docs	0.1962
At 1000 docs	0.1299
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2869

Fallout Level Averages	
Fallout	Recall
0.00000	0.3320
0.00020	0.4506
0.00040	0.5406
0.00060	0.5993
0.00080	0.6400
0.00100	0.6696
0.00120	0.6763
0.00140	0.6763
0.00160	0.6763
0.00180	0.6763
0.00200	0.6763







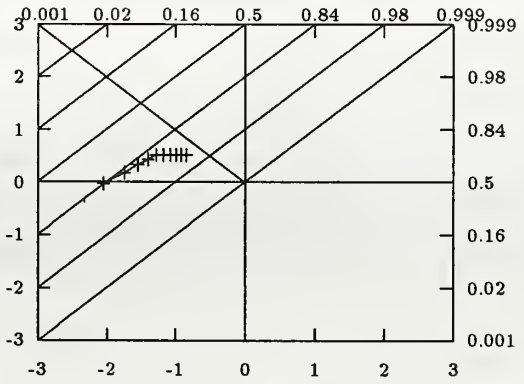
Summary Statistics	
Run Number	CnQst2–full, manual
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	6788

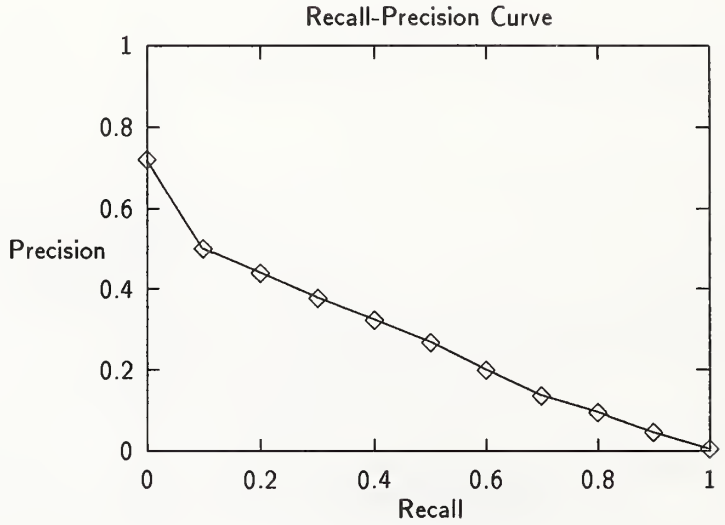
Recall Level Averages	
Recall	Precision
0.00	0.7202
0.10	0.5005
0.20	0.4411
0.30	0.3787
0.40	0.3242
0.50	0.2681
0.60	0.2010
0.70	0.1373
0.80	0.0947
0.90	0.0465
1.00	0.0045
Average precision over all relevant docs	
non-interpolated	0.2633

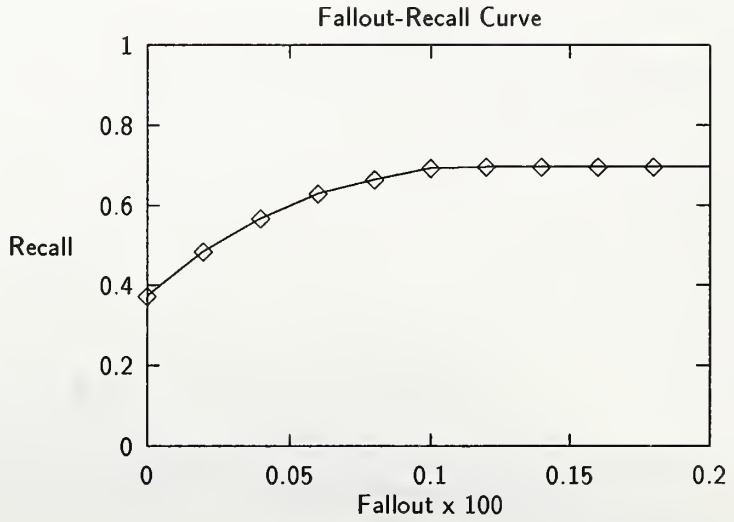
Document Level Averages	
	Precision
At 5 docs	0.5080
At 10 docs	0.4880
At 15 docs	0.4920
At 20 docs	0.4960
At 30 docs	0.4793
At 100 docs	0.4178
At 200 docs	0.3256
At 500 docs	0.2102
At 1000 docs	0.1358
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3140

Fallout Level Averages	
Fallout	Recall
0.00000	0.3737
0.00020	0.4851
0.00040	0.5676
0.00060	0.6290
0.00080	0.6653
0.00100	0.6931
0.00120	0.6960
0.00140	0.6960
0.00160	0.6960
0.00180	0.6960
0.00200	0.6960

NORMAL DEVIATE - Fallout-Recall







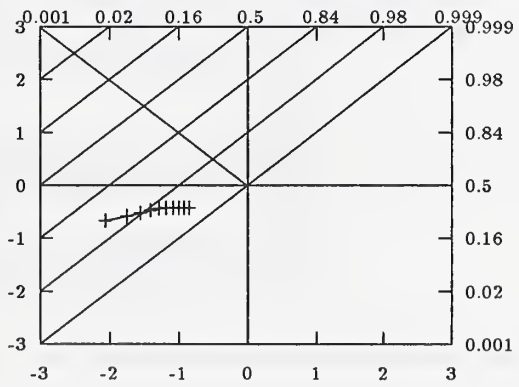
Summary Statistics	
Run Number	prceo1–full, manual
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	3323

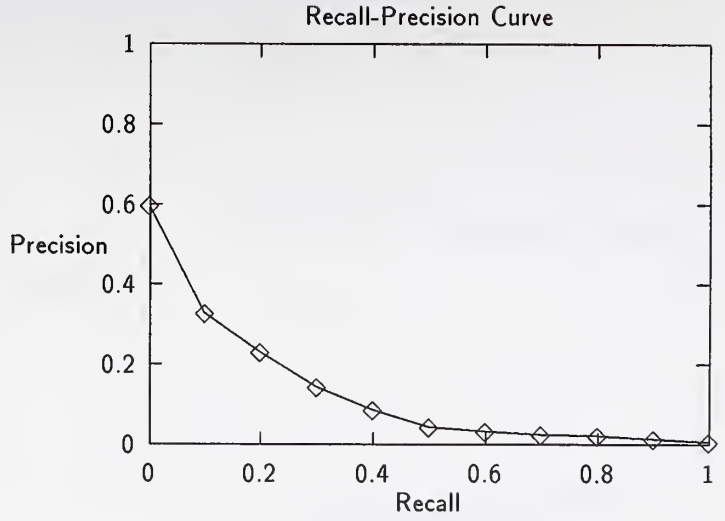
Recall Level Averages	
Recall	Precision
0.00	0.5963
0.10	0.3270
0.20	0.2306
0.30	0.1430
0.40	0.0866
0.50	0.0425
0.60	0.0323
0.70	0.0246
0.80	0.0209
0.90	0.0131
1.00	0.0041
Average precision over all relevant docs	
non-interpolated	0.1120

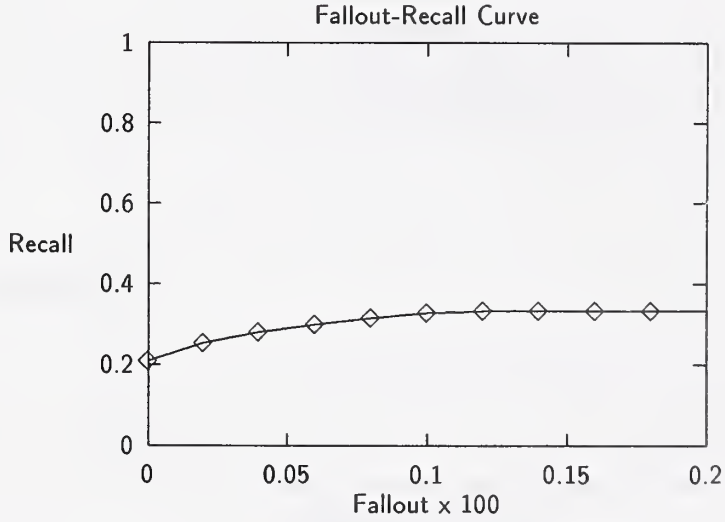
Document Level Averages	
	Precision
At 5 docs	0.4120
At 10 docs	0.4000
At 15 docs	0.3920
At 20 docs	0.3740
At 30 docs	0.3527
At 100 docs	0.2722
At 200 docs	0.1974
At 500 docs	0.1090
At 1000 docs	0.0665
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1809

Fallout Level Averages	
Fallout	Recall
0.00000	0.2103
0.00020	0.2537
0.00040	0.2811
0.00060	0.3006
0.00080	0.3166
0.00100	0.3294
0.00120	0.3335
0.00140	0.3335
0.00160	0.3335
0.00180	0.3335
0.00200	0.3335

NORMAL DEVIATE - Fallout-Recall







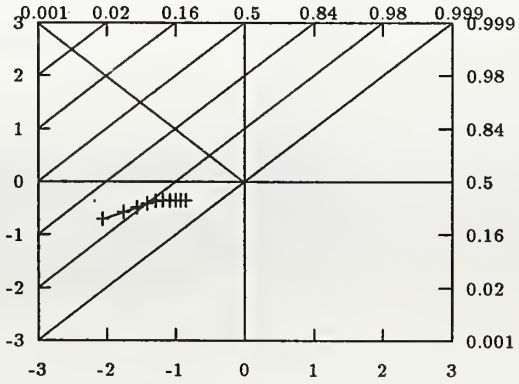
Summary Statistics	
Run Number	prceo2–full, manual
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10785
Rel_ret:	3561

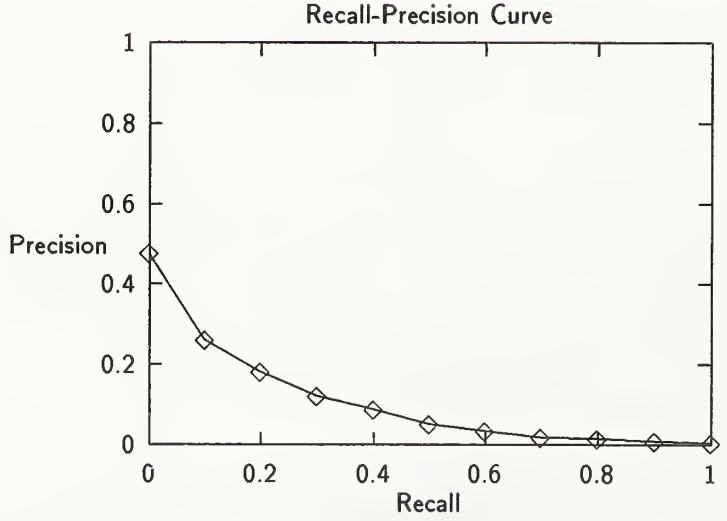
Recall Level Averages	
Recall	Precision
0.00	0.4768
0.10	0.2613
0.20	0.1807
0.30	0.1202
0.40	0.0875
0.50	0.0504
0.60	0.0328
0.70	0.0169
0.80	0.0142
0.90	0.0077
1.00	0.0031
Average precision over all relevant docs	
non-interpolated	0.0904

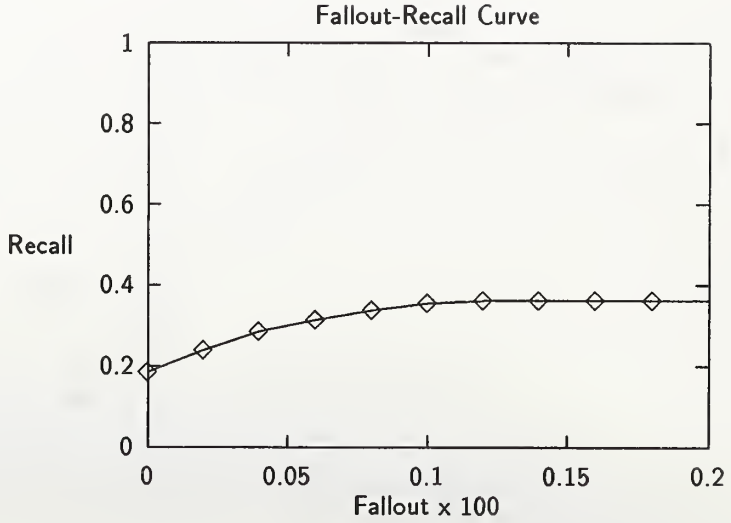
Document Level Averages	
	Precision
At 5 docs	0.2120
At 10 docs	0.2480
At 15 docs	0.2707
At 20 docs	0.2760
At 30 docs	0.2753
At 100 docs	0.2418
At 200 docs	0.1756
At 500 docs	0.1086
At 1000 docs	0.0712
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1703

Fallout Level Averages	
Fallout	Recall
0.00000	0.1861
0.00020	0.2402
0.00040	0.2860
0.00060	0.3147
0.00080	0.3387
0.00100	0.3564
0.00120	0.3623
0.00140	0.3623
0.00160	0.3623
0.00180	0.3623
0.00200	0.3623

NORMAL DEVIATE - Fallout-Recall







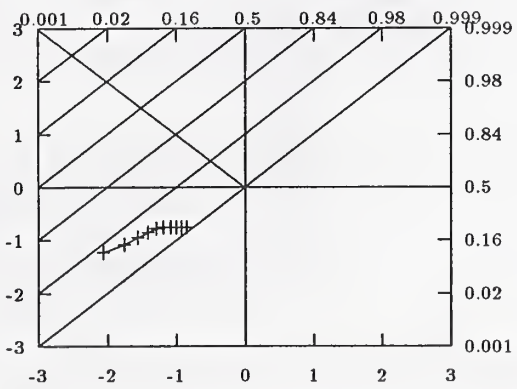
Summary Statistics	
Run Number	UREKA2–full, manual
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50042
Relevant:	10785
Rel_ret:	2690

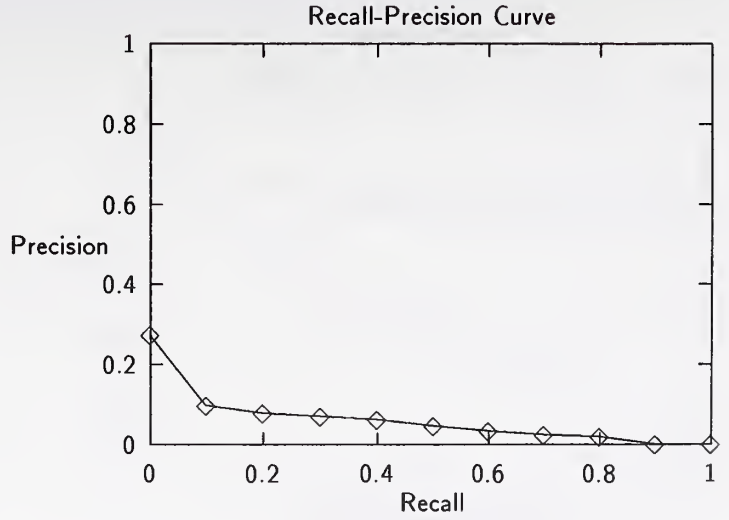
Recall Level Averages	
Recall	Precision
0.00	0.2740
0.10	0.0977
0.20	0.0774
0.30	0.0701
0.40	0.0617
0.50	0.0467
0.60	0.0338
0.70	0.0242
0.80	0.0187
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0502

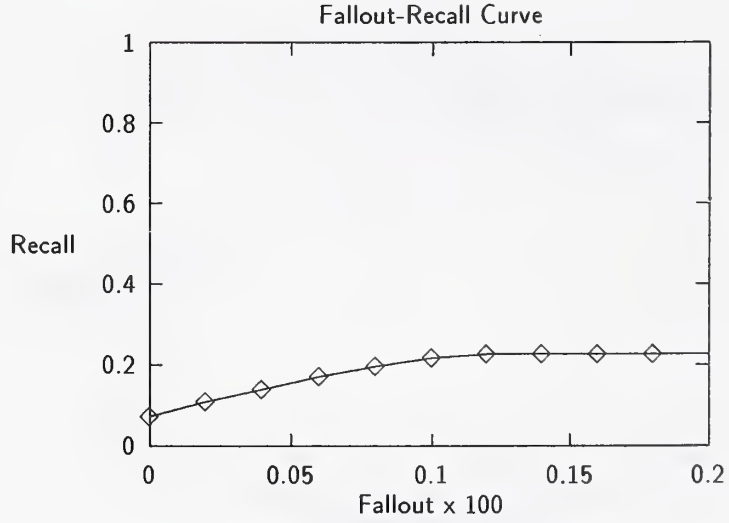
Document Level Averages	
	Precision
At 5 docs	0.1360
At 10 docs	0.1140
At 15 docs	0.1067
At 20 docs	0.1100
At 30 docs	0.1033
At 100 docs	0.0912
At 200 docs	0.0831
At 500 docs	0.0664
At 1000 docs	0.0538
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0793

Fallout Level Averages	
Fallout	Recall
0.00000	0.0733
0.00020	0.1100
0.00040	0.1403
0.00060	0.1713
0.00080	0.1970
0.00100	0.2173
0.00120	0.2271
0.00140	0.2271
0.00160	0.2271
0.00180	0.2271
0.00200	0.2271

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	UREKA3–full, manual
Num of Queries	50

Total number of documents over all queries	
Retrieved:	48914
Relevant:	10785
Rel_ret:	3282

Recall Level Averages	
Recall	Precision
0.00	0.4228
0.10	0.1880
0.20	0.1385
0.30	0.1045
0.40	0.0840
0.50	0.0643
0.60	0.0445
0.70	0.0365
0.80	0.0183
0.90	0.0000
1.00	0.0000

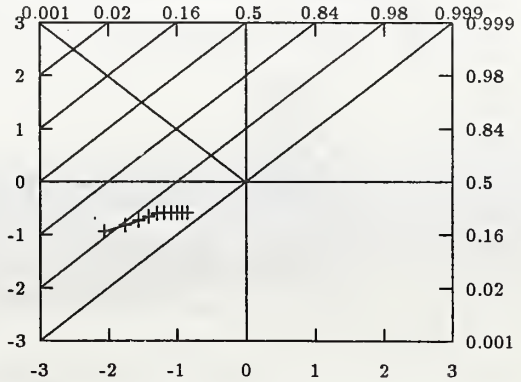
Average precision over all relevant docs	
non-interpolated	0.0819

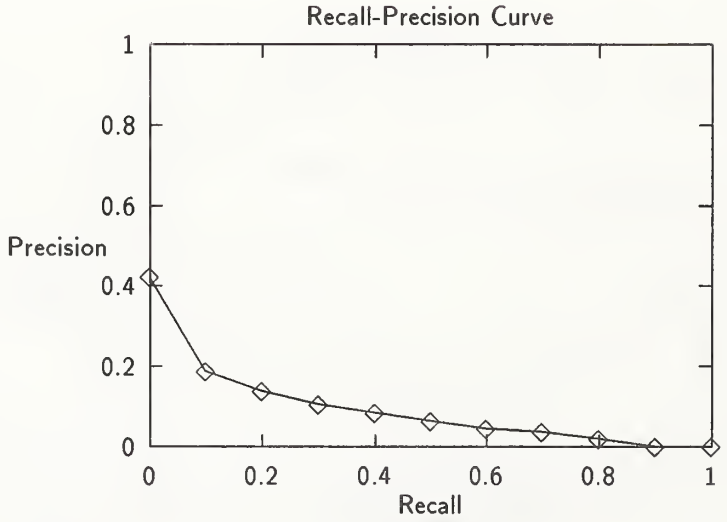
Document Level Averages	
	Precision
At 5 docs	0.2520
At 10 docs	0.2320
At 15 docs	0.2267
At 20 docs	0.2200
At 30 docs	0.2073
At 100 docs	0.1688
At 200 docs	0.1383
At 500 docs	0.0954
At 1000 docs	0.0656

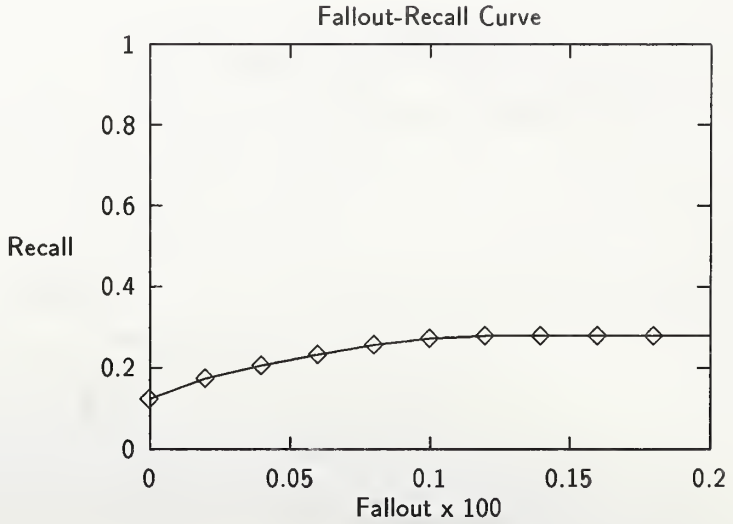
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1231

Fallout Level Averages	
Fallout	Recall
0.00000	0.1242
0.00020	0.1744
0.00040	0.2072
0.00060	0.2337
0.00080	0.2571
0.00100	0.2740
0.00120	0.2805
0.00140	0.2805
0.00160	0.2805
0.00180	0.2805
0.00200	0.2805

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	uclaa1–category B, automatic
Num of Queries	50

Total number of documents over all queries	
Retrieved:	50000
Relevant:	3929
Rel_ret:	3204

Recall Level Averages	
Recall	Precision
0.00	0.8126
0.10	0.6075
0.20	0.5292
0.30	0.4385
0.40	0.3861
0.50	0.3210
0.60	0.2767
0.70	0.2284
0.80	0.1646
0.90	0.0938
1.00	0.0163

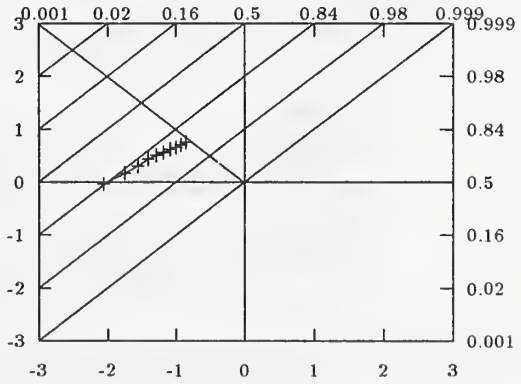
Average precision over all relevant docs	
non-interpolated	0.3345

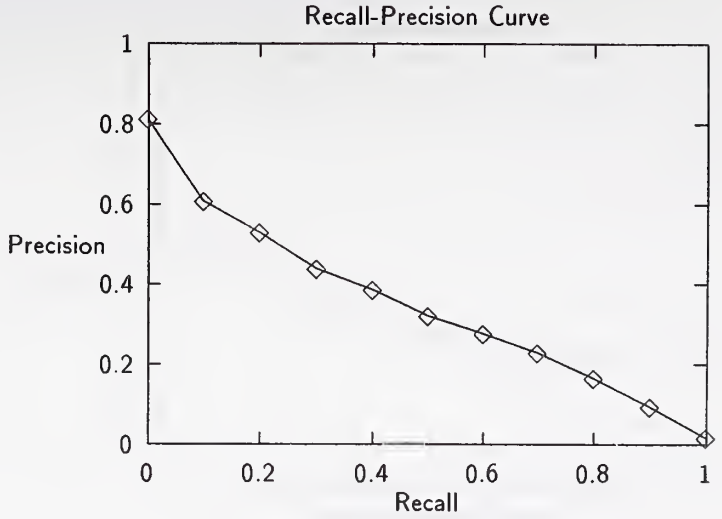
Document Level Averages	
	Precision
At 5 docs	0.5840
At 10 docs	0.5380
At 15 docs	0.4973
At 20 docs	0.4660
At 30 docs	0.4333
At 100 docs	0.3098
At 200 docs	0.2216
At 500 docs	0.1140
At 1000 docs	0.0641

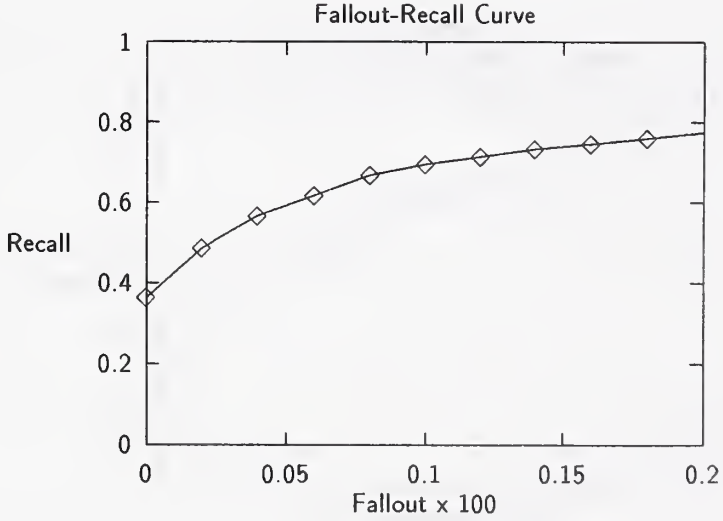
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3629

Fallout Level Averages	
Fallout	Recall
0.00000	0.3638
0.00020	0.4872
0.00040	0.5666
0.00060	0.6174
0.00080	0.6677
0.00100	0.6944
0.00120	0.7136
0.00140	0.7335
0.00160	0.7451
0.00180	0.7604
0.00200	0.7748

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	uclaa2–category B, automatic
Num of Queries	50

Total number of documents over all queries	
Retrieved:	50000
Relevant:	3929
Rel_ret:	3059

Recall Level Averages	
Recall	Precision
0.00	0.7464
0.10	0.5880
0.20	0.5024
0.30	0.4058
0.40	0.3429
0.50	0.2698
0.60	0.2168
0.70	0.1803
0.80	0.1209
0.90	0.0695
1.00	0.0098

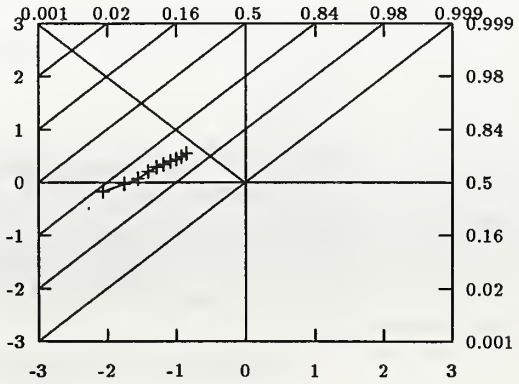
Average precision over all relevant docs	
non-interpolated	0.2957

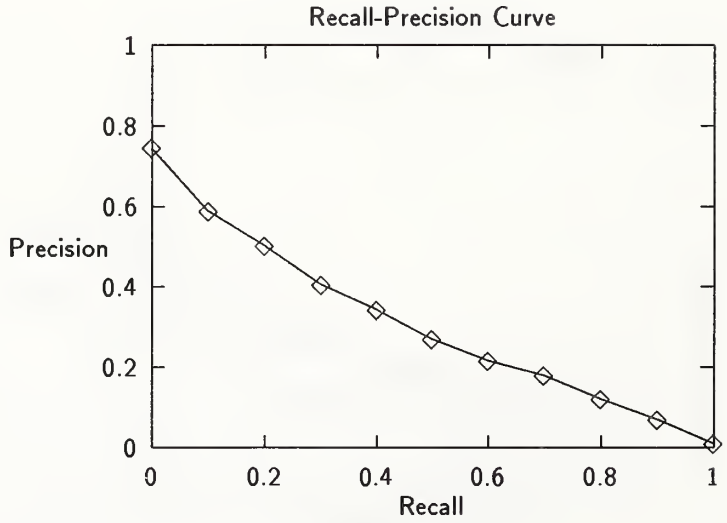
Document Level Averages	
	Precision
At 5 docs	0.5440
At 10 docs	0.5180
At 15 docs	0.4920
At 20 docs	0.4650
At 30 docs	0.4207
At 100 docs	0.2760
At 200 docs	0.1958
At 500 docs	0.1057
At 1000 docs	0.0612

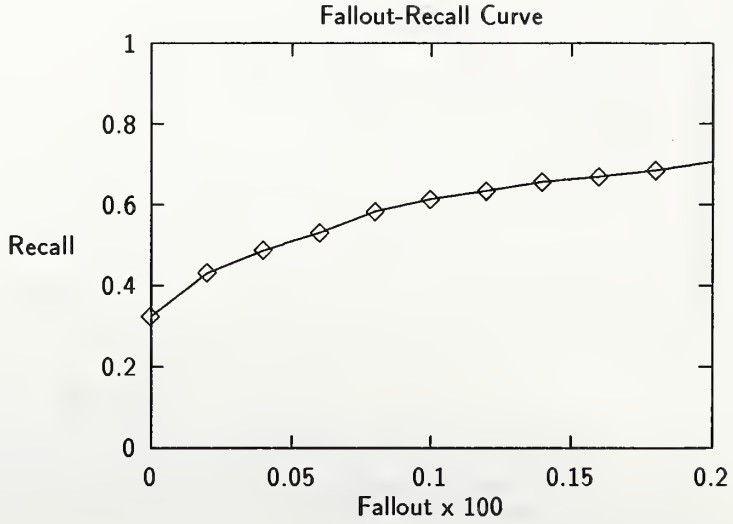
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3289

Fallout Level Averages	
Fallout	Recall
0.00000	0.3245
0.00020	0.4324
0.00040	0.4885
0.00060	0.5313
0.00080	0.5838
0.00100	0.6135
0.00120	0.6339
0.00140	0.6561
0.00160	0.6698
0.00180	0.6852
0.00200	0.7066

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	uclaf1–category B, feedback, frozen evaluation
Num of Queries	50

Total number of documents over all queries	
Retrieved:	50473
Relevant:	3929
Rel_ret:	3047

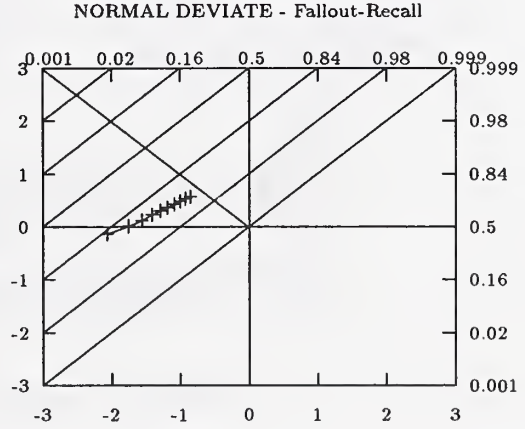
Recall Level Averages	
Recall	Precision
0.00	0.8125
0.10	0.5957
0.20	0.5218
0.30	0.4113
0.40	0.3484
0.50	0.2813
0.60	0.2289
0.70	0.1915
0.80	0.1351
0.90	0.0764
1.00	0.0105

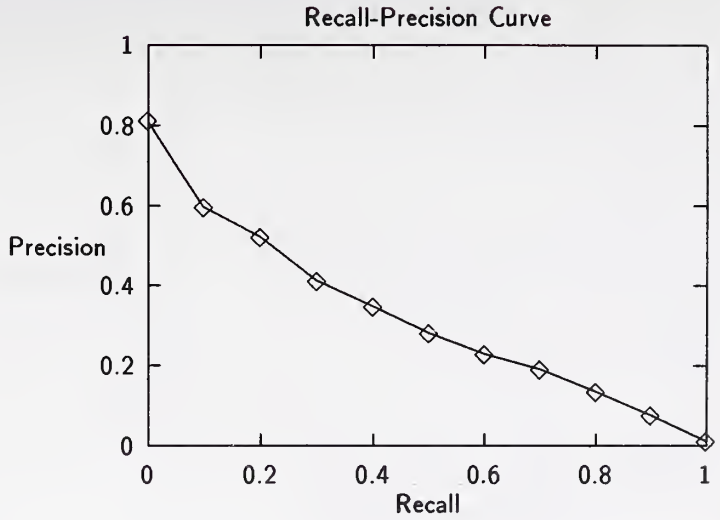
Average precision over all relevant docs	
non-interpolated	0.3091

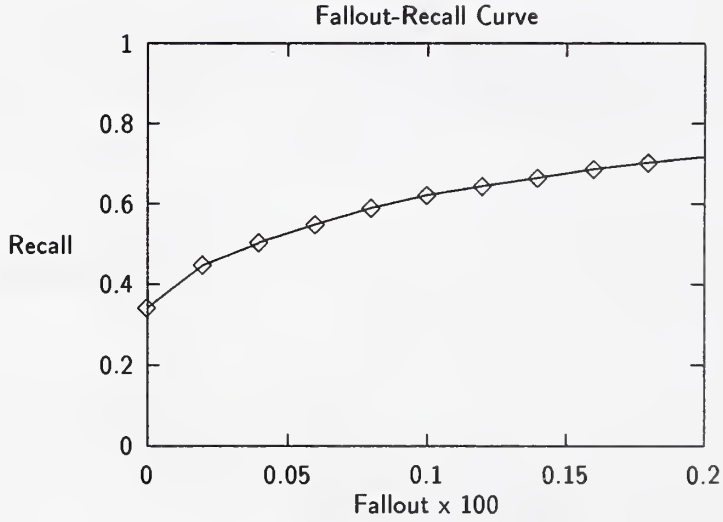
Document Level Averages	
	Precision
At 5 docs	0.5880
At 10 docs	0.5220
At 15 docs	0.4893
At 20 docs	0.4640
At 30 docs	0.4360
At 100 docs	0.2884
At 200 docs	0.1992
At 500 docs	0.1058
At 1000 docs	0.0609

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3459

Fallout Level Averages	
Fallout	Recall
0.00000	0.3425
0.00020	0.4483
0.00040	0.5041
0.00060	0.5489
0.00080	0.5898
0.00100	0.6216
0.00120	0.6431
0.00140	0.6653
0.00160	0.6864
0.00180	0.7030
0.00200	0.7172







Summary Statistics	
Run Number	uicah–full, automatic
Num of Queries	50

Total number of documents over all queries	
Retrieved:	45108
Relevant:	10785
Rel_ret:	4421

Recall Level Averages	
Recall	Precision
0.00	0.5665
0.10	0.3272
0.20	0.2527
0.30	0.1937
0.40	0.1413
0.50	0.0712
0.60	0.0393
0.70	0.0069
0.80	0.0000
0.90	0.0000
1.00	0.0000

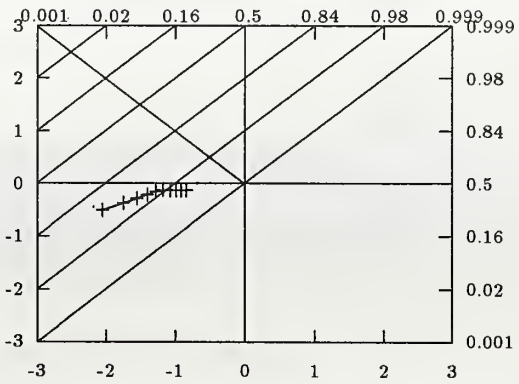
Average precision over all relevant docs	
non-interpolated	0.1200

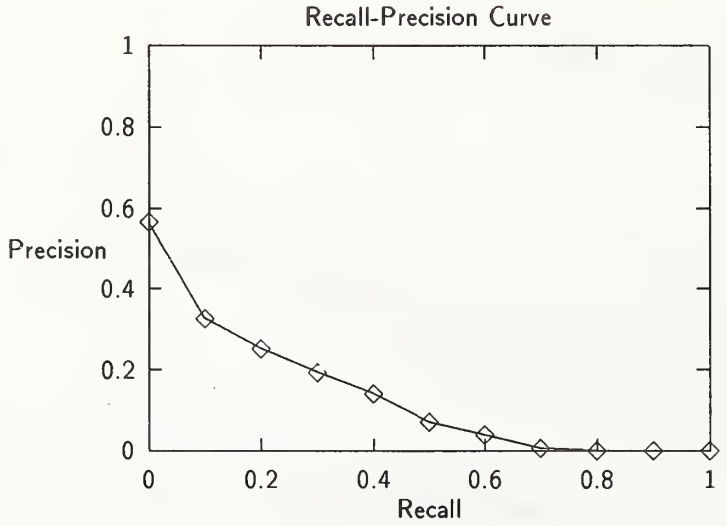
Document Level Averages	
	Precision
At 5 docs	0.3280
At 10 docs	0.3340
At 15 docs	0.3280
At 20 docs	0.3380
At 30 docs	0.3233
At 100 docs	0.2784
At 200 docs	0.2107
At 500 docs	0.1392
At 1000 docs	0.0884

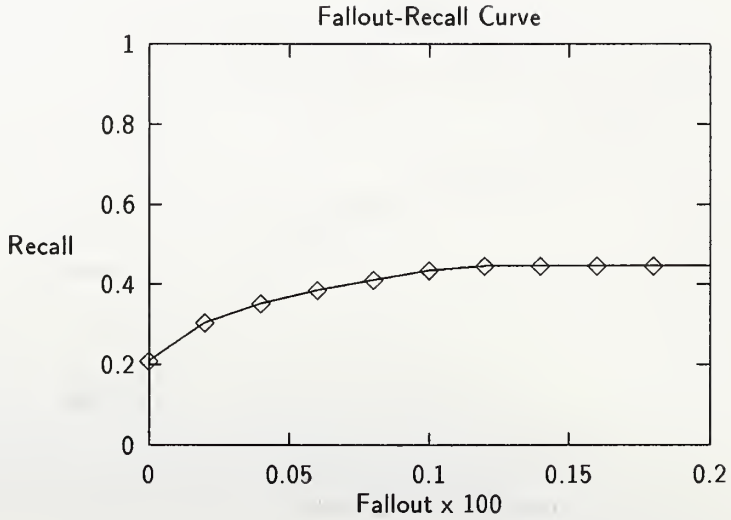
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1950

Fallout Level Averages	
Fallout	Recall
0.00000	0.2095
0.00020	0.3057
0.00040	0.3532
0.00060	0.3860
0.00080	0.4125
0.00100	0.4359
0.00120	0.4467
0.00140	0.4467
0.00160	0.4467
0.00180	0.4467
0.00200	0.4467

NORMAL DEVIATE - Fallout-Recall





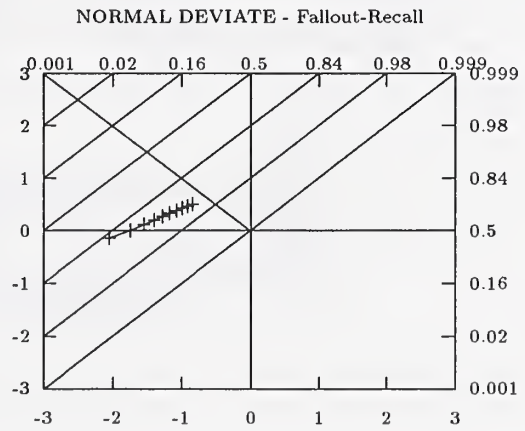


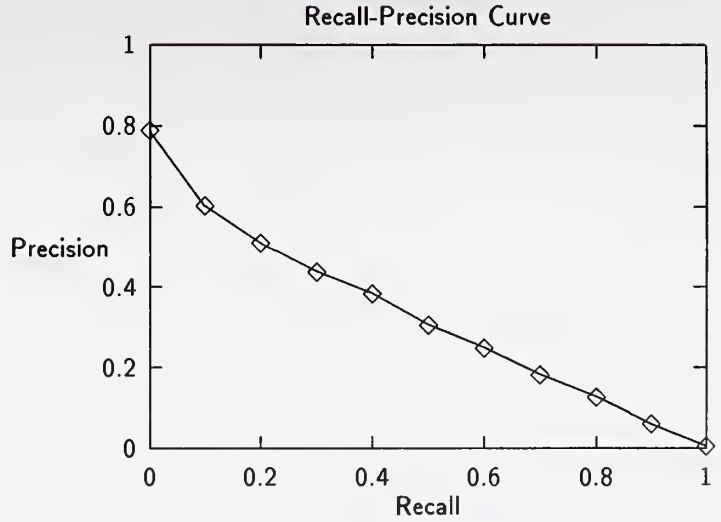
Summary Statistics	
Run Number	cityr1–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10489
Rel_ret:	6801

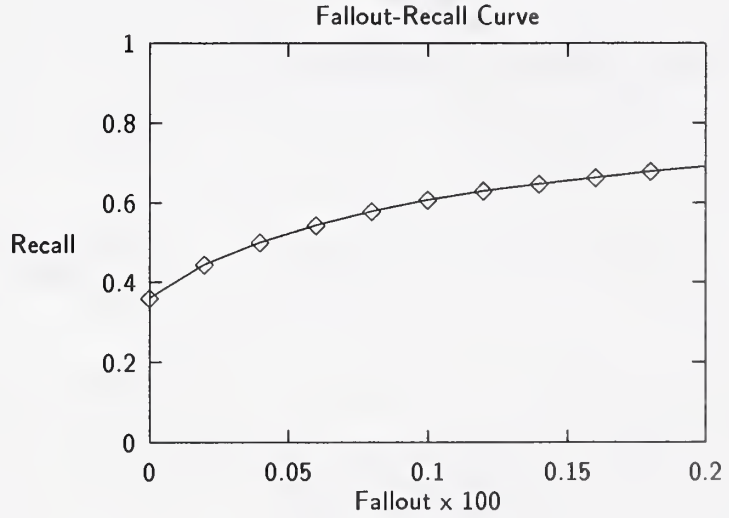
Recall Level Averages	
Recall	Precision
0.00	0.7888
0.10	0.6028
0.20	0.5101
0.30	0.4380
0.40	0.3851
0.50	0.3070
0.60	0.2494
0.70	0.1826
0.80	0.1269
0.90	0.0606
1.00	0.0048
Average precision over all relevant docs	
non-interpolated	0.3149

Document Level Averages	
	Precision
At 5 docs	0.6280
At 10 docs	0.5940
At 15 docs	0.5867
At 20 docs	0.5690
At 30 docs	0.5333
At 100 docs	0.4316
At 200 docs	0.3385
At 500 docs	0.2154
At 1000 docs	0.1360
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3607

Fallout Level Averages	
Fallout	Recall
0.00000	0.3610
0.00020	0.4444
0.00040	0.5009
0.00060	0.5433
0.00080	0.5791
0.00100	0.6074
0.00120	0.6294
0.00140	0.6476
0.00160	0.6636
0.00180	0.6784
0.00200	0.6918







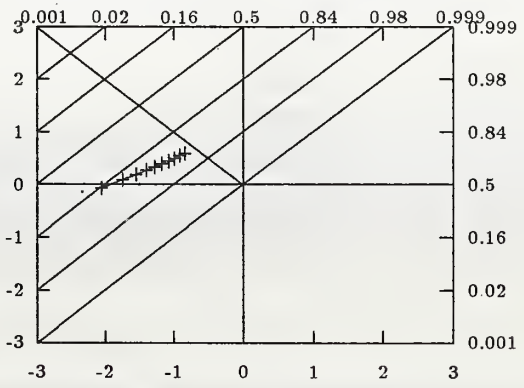
Summary Statistics	
Run Number	cityr2–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10489
Rel_ret:	7134

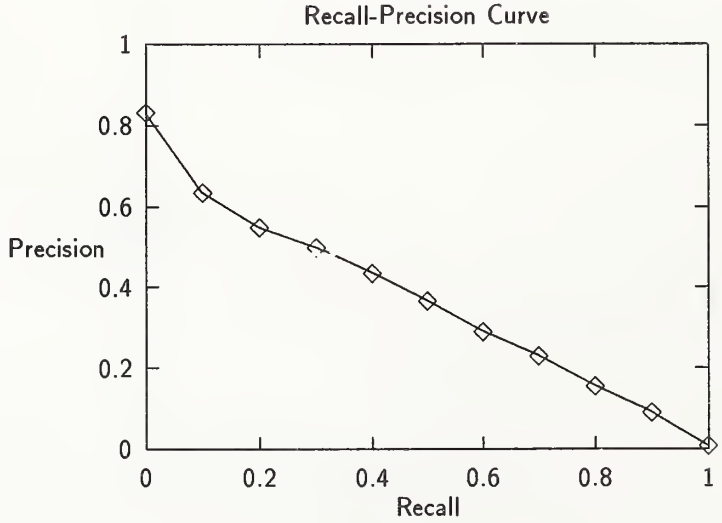
Recall Level Averages	
Recall	Precision
0.00	0.8313
0.10	0.6347
0.20	0.5491
0.30	0.4990
0.40	0.4356
0.50	0.3678
0.60	0.2908
0.70	0.2305
0.80	0.1551
0.90	0.0900
1.00	0.0085
Average precision over all relevant docs	
non-interpolated	0.3562

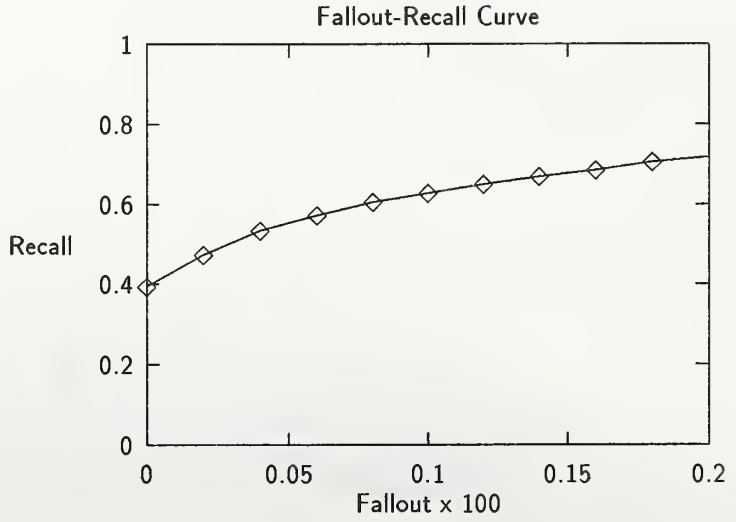
Document Level Averages	
	Precision
At 5 docs	0.6920
At 10 docs	0.6500
At 15 docs	0.6213
At 20 docs	0.6060
At 30 docs	0.5613
At 100 docs	0.4494
At 200 docs	0.3588
At 500 docs	0.2261
At 1000 docs	0.1427
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3877

Fallout Level Averages	
Fallout	Recall
0.00000	0.3949
0.00020	0.4731
0.00040	0.5337
0.00060	0.5722
0.00080	0.6050
0.00100	0.6278
0.00120	0.6510
0.00140	0.6696
0.00160	0.6851
0.00180	0.7065
0.00200	0.7181

NORMAL DEVIATE - Fallout-Recall







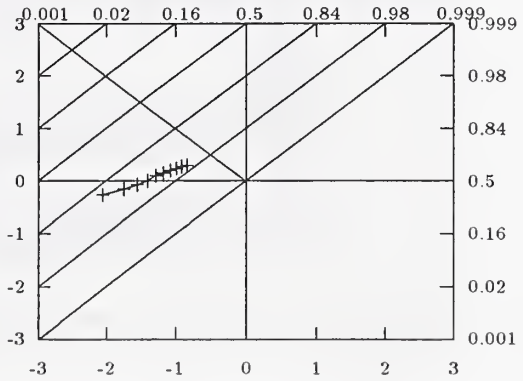
Summary Statistics	
Run Number	rutcombx–full, manual
Num of Queries	50
Total number of documents over all queries	
Retrieved:	49983
Relevant:	10489
Rel_ret:	6244

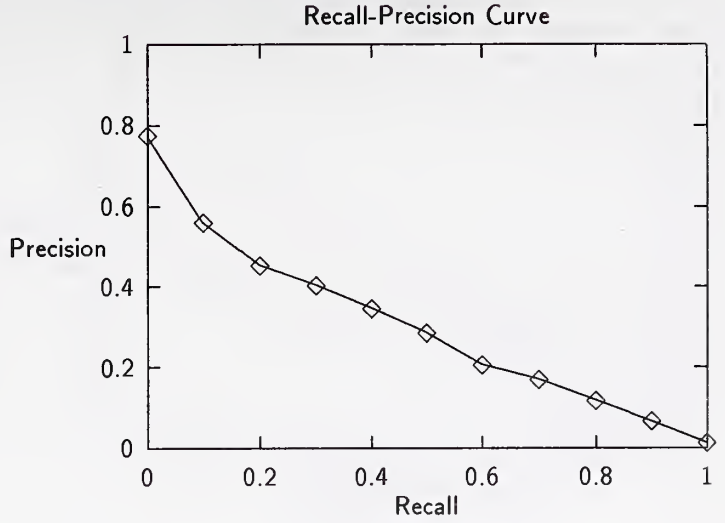
Recall Level Averages	
Recall	Precision
0.00	0.7739
0.10	0.5601
0.20	0.4537
0.30	0.4042
0.40	0.3458
0.50	0.2860
0.60	0.2074
0.70	0.1708
0.80	0.1180
0.90	0.0668
1.00	0.0132
Average precision over all relevant docs	
non-interpolated	0.2905

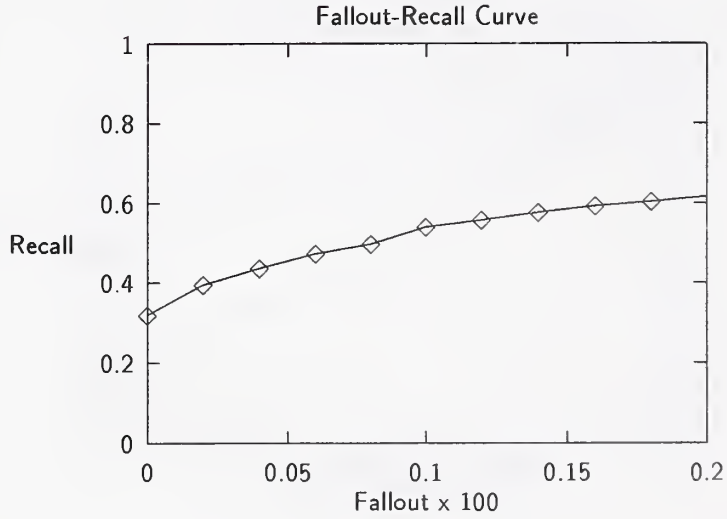
Document Level Averages	
	Precision
At 5 docs	0.6000
At 10 docs	0.5640
At 15 docs	0.5427
At 20 docs	0.5300
At 30 docs	0.5000
At 100 docs	0.3880
At 200 docs	0.3023
At 500 docs	0.1930
At 1000 docs	0.1249
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3375

Fallout Level Averages	
Fallout	Recall
0.00000	0.3188
0.00020	0.3966
0.00040	0.4374
0.00060	0.4733
0.00080	0.4980
0.00100	0.5403
0.00120	0.5587
0.00140	0.5765
0.00160	0.5927
0.00180	0.6044
0.00200	0.6169

NORMAL DEVIATE - Fallout-Recall







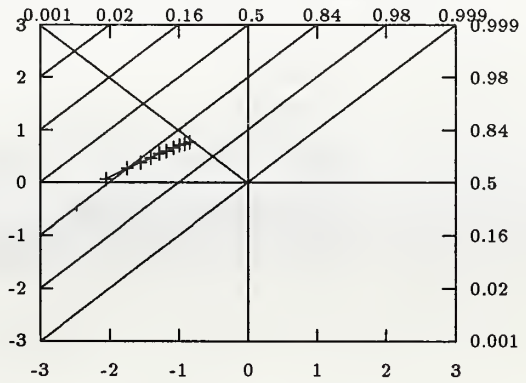
Summary Statistics	
Run Number	crnlR1–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10489
Rel_ret:	7764

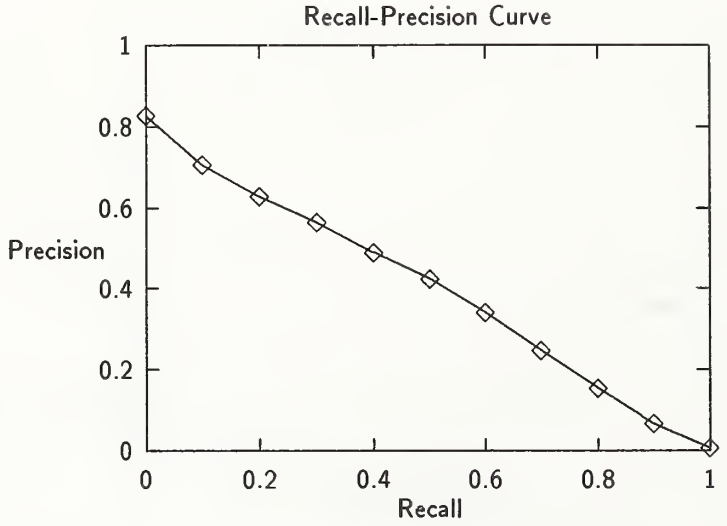
Recall Level Averages	
Recall	Precision
0.00	0.8268
0.10	0.7060
0.20	0.6282
0.30	0.5636
0.40	0.4900
0.50	0.4246
0.60	0.3416
0.70	0.2483
0.80	0.1543
0.90	0.0675
1.00	0.0062
Average precision over all relevant docs	
non-interpolated	0.3952

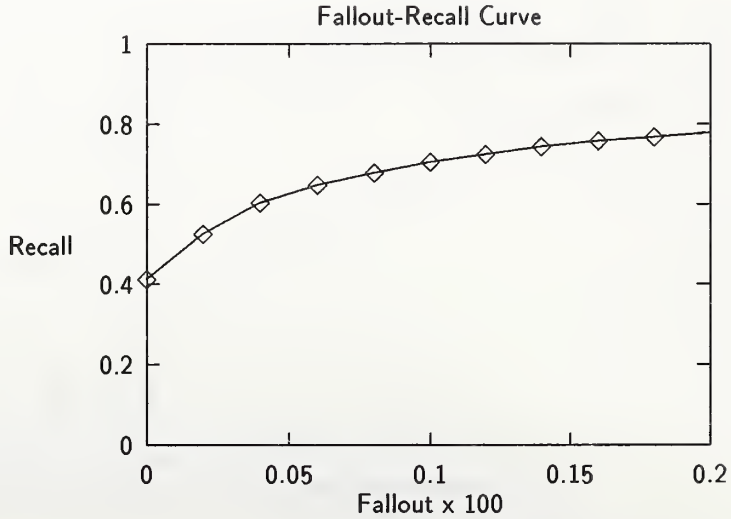
Document Level Averages	
	Precision
At 5 docs	0.6800
At 10 docs	0.6780
At 15 docs	0.6640
At 20 docs	0.6560
At 30 docs	0.6247
At 100 docs	0.4950
At 200 docs	0.3974
At 500 docs	0.2526
At 1000 docs	0.1553
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4273

Fallout Level Averages	
Fallout	Recall
0.00000	0.4126
0.00020	0.5260
0.00040	0.6041
0.00060	0.6485
0.00080	0.6786
0.00100	0.7056
0.00120	0.7247
0.00140	0.7433
0.00160	0.7588
0.00180	0.7685
0.00200	0.7794

NORMAL DEVIATE - Fallout-Recall







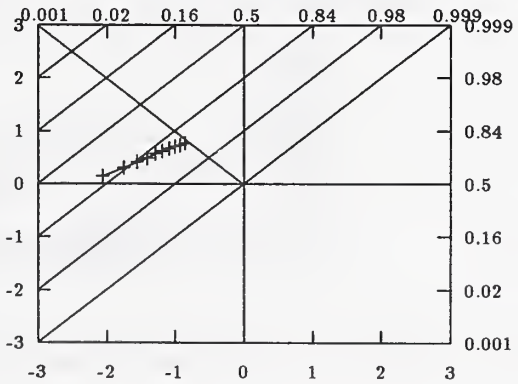
Summary Statistics	
Run Number	crnlC1–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10489
Rel_ret:	7808

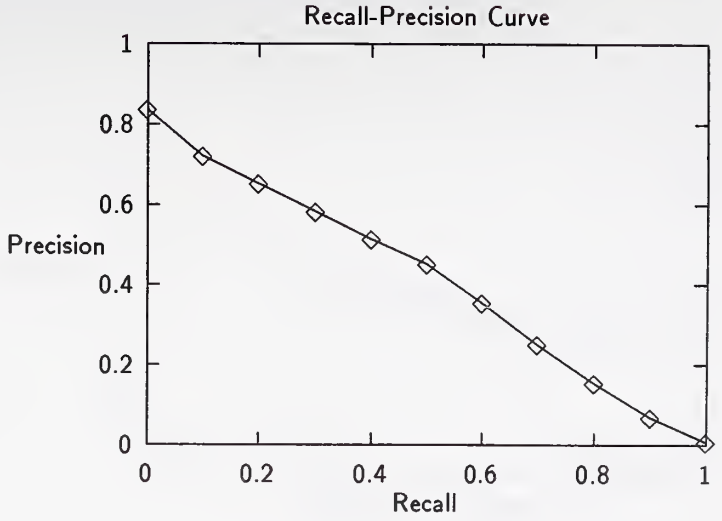
Recall Level Averages	
Recall	Precision
0.00	0.8355
0.10	0.7202
0.20	0.6516
0.30	0.5816
0.40	0.5142
0.50	0.4518
0.60	0.3542
0.70	0.2519
0.80	0.1547
0.90	0.0677
1.00	0.0062
Average precision over all relevant docs	
non-interpolated	0.4091

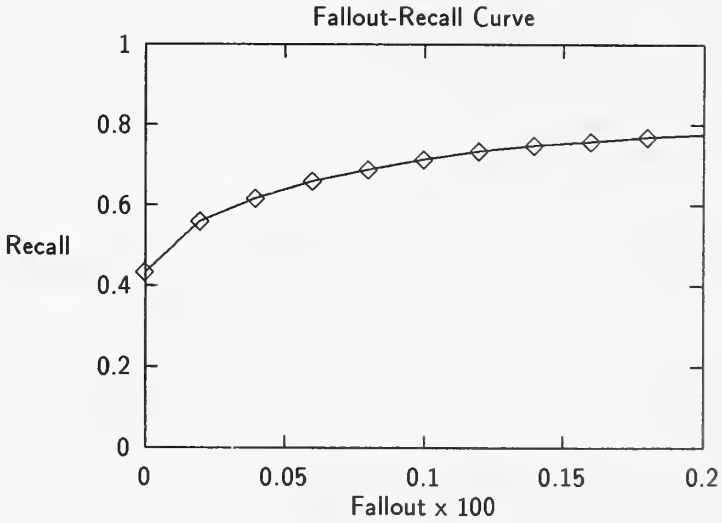
Document Level Averages	
	Precision
At 5 docs	0.7240
At 10 docs	0.7000
At 15 docs	0.6840
At 20 docs	0.6660
At 30 docs	0.6240
At 100 docs	0.5156
At 200 docs	0.4104
At 500 docs	0.2570
At 1000 docs	0.1562
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4367

Fallout Level Averages	
Fallout	Recall
0.00000	0.4336
0.00020	0.5602
0.00040	0.6170
0.00060	0.6592
0.00080	0.6886
0.00100	0.7144
0.00120	0.7338
0.00140	0.7489
0.00160	0.7588
0.00180	0.7687
0.00200	0.7763

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	Brkly4–full, automatic
Num of Queries	50

Total number of documents over all queries	
Retrieved:	50000
Relevant:	10489
Rel_ret:	6339

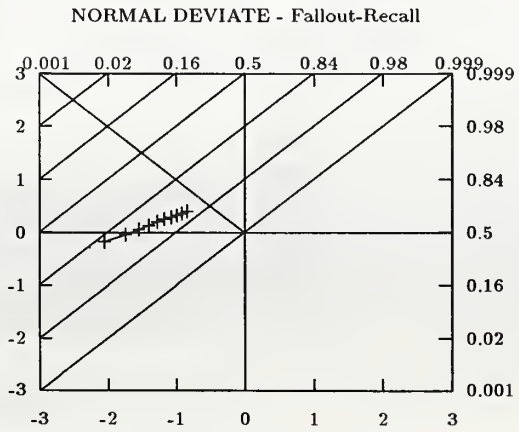
Recall Level Averages	
Recall	Precision
0.00	0.7606
0.10	0.5512
0.20	0.4543
0.30	0.4004
0.40	0.3522
0.50	0.2797
0.60	0.2032
0.70	0.1665
0.80	0.1166
0.90	0.0631
1.00	0.0147

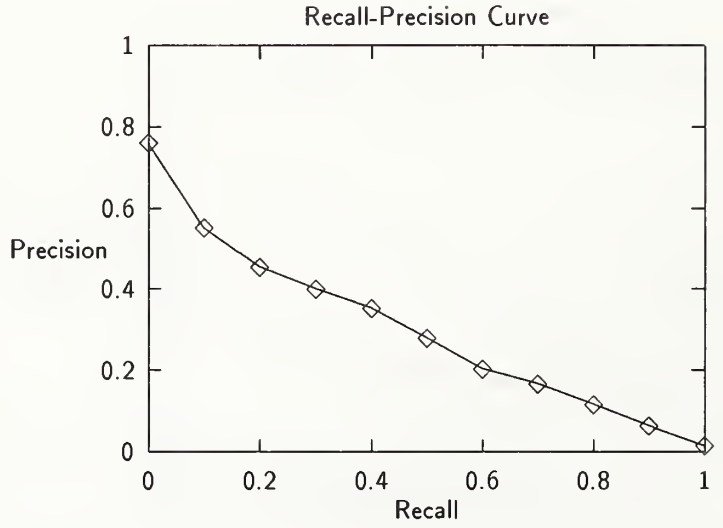
Average precision over all relevant docs	
non-interpolated	0.2905

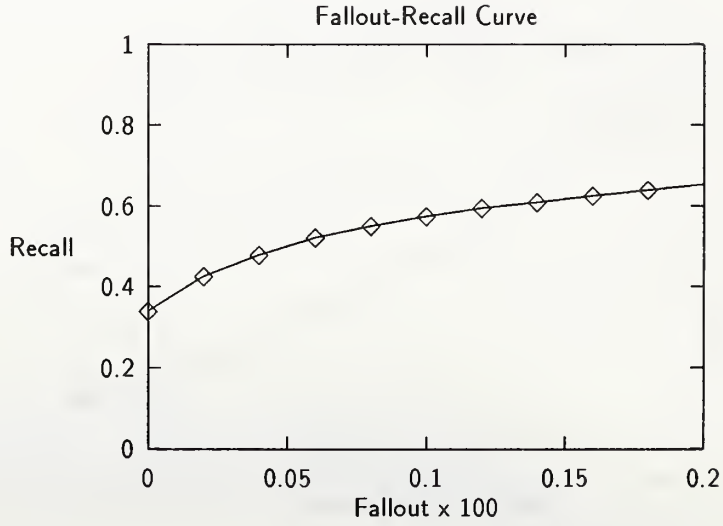
Document Level Averages	
	Precision
At 5 docs	0.6080
At 10 docs	0.5920
At 15 docs	0.5707
At 20 docs	0.5580
At 30 docs	0.5160
At 100 docs	0.3934
At 200 docs	0.3117
At 500 docs	0.1984
At 1000 docs	0.1268

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3332

Fallout Level Averages	
Fallout	Recall
0.00000	0.3395
0.00020	0.4263
0.00040	0.4791
0.00060	0.5212
0.00080	0.5500
0.00100	0.5747
0.00120	0.5939
0.00140	0.6096
0.00160	0.6255
0.00180	0.6388
0.00200	0.6530





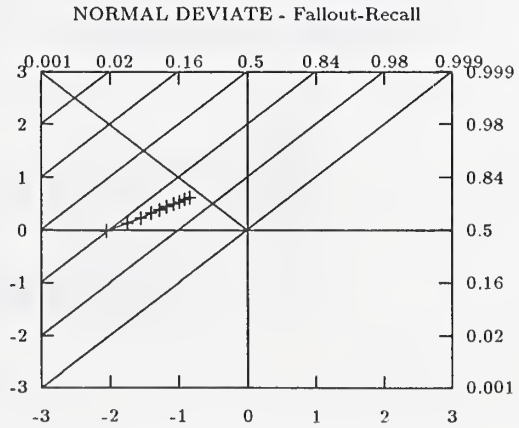


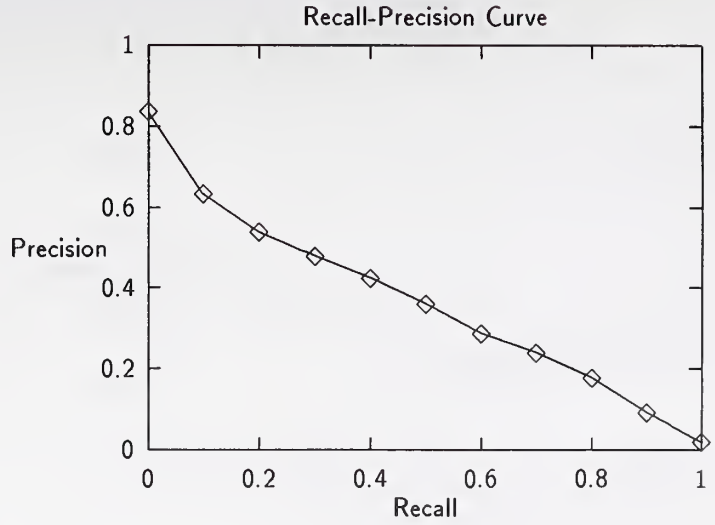
Summary Statistics	
Run Number	Brkly5–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10489
Rel_ret:	7237

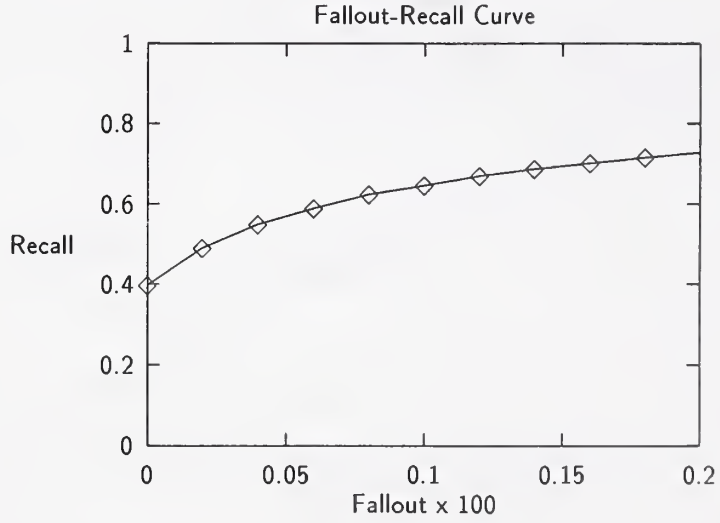
Recall Level Averages	
Recall	Precision
0.00	0.8361
0.10	0.6331
0.20	0.5384
0.30	0.4781
0.40	0.4237
0.50	0.3611
0.60	0.2872
0.70	0.2400
0.80	0.1774
0.90	0.0928
1.00	0.0190
Average precision over all relevant docs	
non-interpolated	0.3538

Document Level Averages	
	Precision
At 5 docs	0.6600
At 10 docs	0.6360
At 15 docs	0.6160
At 20 docs	0.6040
At 30 docs	0.5753
At 100 docs	0.4608
At 200 docs	0.3650
At 500 docs	0.2310
At 1000 docs	0.1447
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3852

Fallout Level Averages	
Fallout	Recall
0.00000	0.3969
0.00020	0.4892
0.00040	0.5489
0.00060	0.5881
0.00080	0.6227
0.00100	0.6458
0.00120	0.6690
0.00140	0.6864
0.00160	0.7007
0.00180	0.7158
0.00200	0.7277







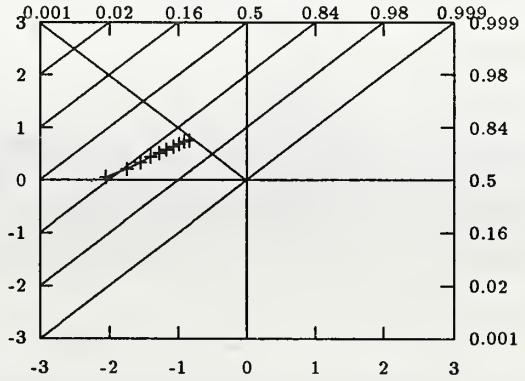
Summary Statistics	
Run Number	dortP1–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10489
Rel_ret:	7748

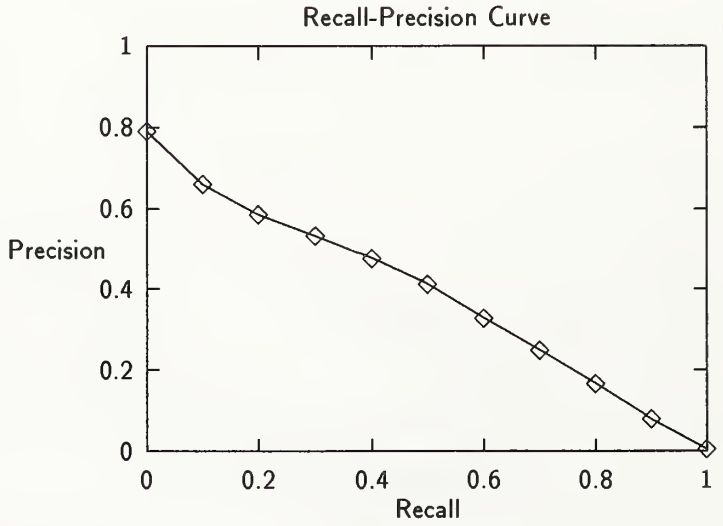
Recall Level Averages	
Recall	Precision
0.00	0.7899
0.10	0.6598
0.20	0.5844
0.30	0.5331
0.40	0.4765
0.50	0.4134
0.60	0.3289
0.70	0.2502
0.80	0.1665
0.90	0.0801
1.00	0.0041
Average precision over all relevant docs	
non-interpolated	0.3800

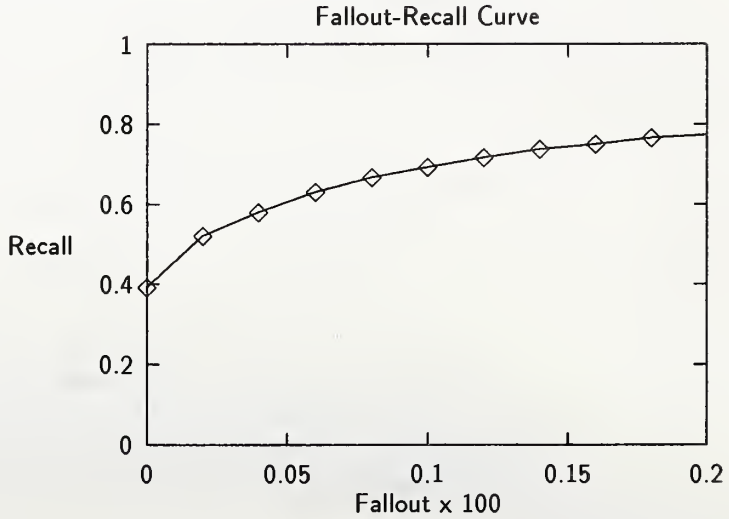
Document Level Averages	
	Precision
At 5 docs	0.6600
At 10 docs	0.6380
At 15 docs	0.6387
At 20 docs	0.6300
At 30 docs	0.6080
At 100 docs	0.4816
At 200 docs	0.3904
At 500 docs	0.2500
At 1000 docs	0.1550
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4195

Fallout Level Averages	
Fallout	Recall
0.00000	0.3933
0.00020	0.5213
0.00040	0.5807
0.00060	0.6304
0.00080	0.6679
0.00100	0.6933
0.00120	0.7171
0.00140	0.7370
0.00160	0.7511
0.00180	0.7666
0.00200	0.7743

NORMAL DEVIATE - Fallout-Recall







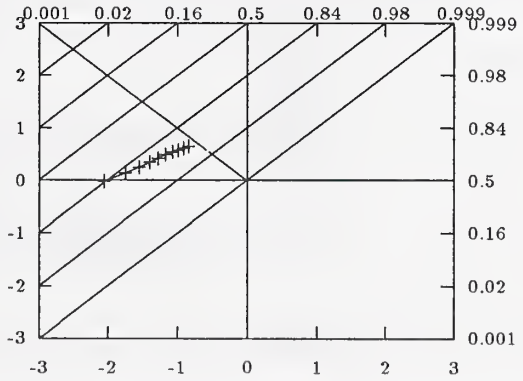
Summary Statistics	
Run Number	dortV1–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10489
Rel_ret:	7386

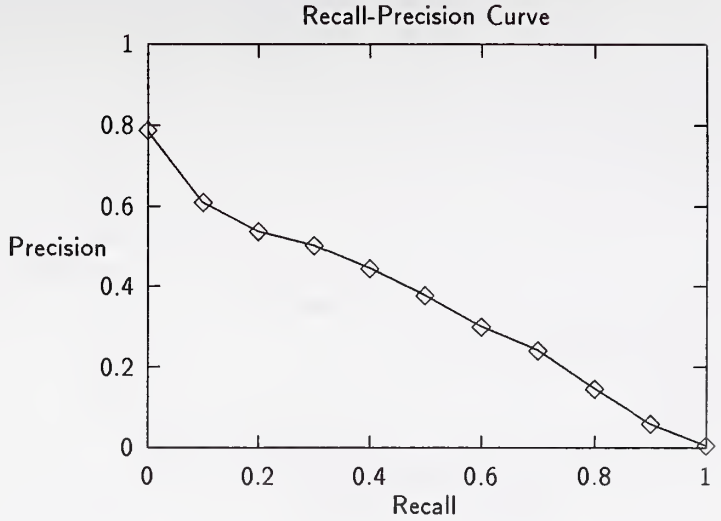
Recall Level Averages	
Recall	Precision
0.00	0.7864
0.10	0.6082
0.20	0.5373
0.30	0.5023
0.40	0.4446
0.50	0.3788
0.60	0.2997
0.70	0.2415
0.80	0.1467
0.90	0.0590
1.00	0.0049
Average precision over all relevant docs	
non-interpolated	0.3516

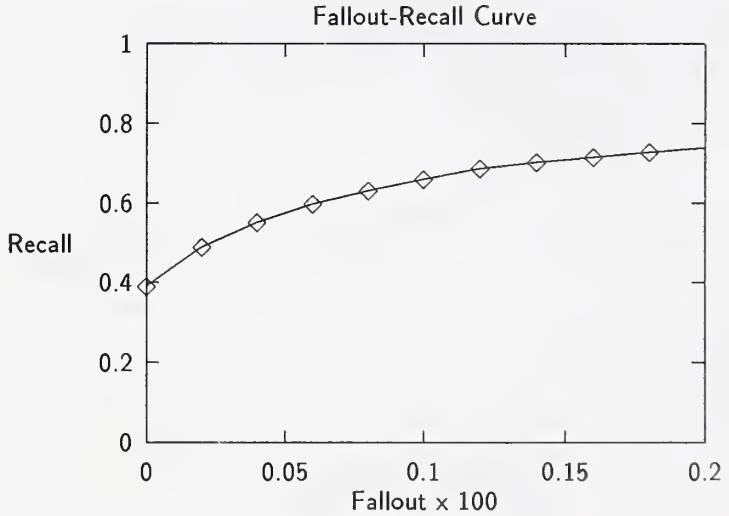
Document Level Averages	
	Precision
At 5 docs	0.6280
At 10 docs	0.5980
At 15 docs	0.5960
At 20 docs	0.5790
At 30 docs	0.5653
At 100 docs	0.4520
At 200 docs	0.3661
At 500 docs	0.2352
At 1000 docs	0.1477
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3947

Fallout Level Averages	
Fallout	Recall
0.00000	0.3912
0.00020	0.4894
0.00040	0.5521
0.00060	0.5974
0.00080	0.6316
0.00100	0.6602
0.00120	0.6863
0.00140	0.7020
0.00160	0.7152
0.00180	0.7279
0.00200	0.7389

NORMAL DEVIATE - Fallout-Recall







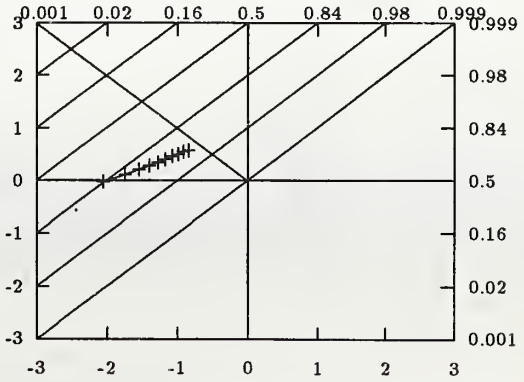
Summary Statistics	
Run Number	INQ003–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10489
Rel_ret:	7161

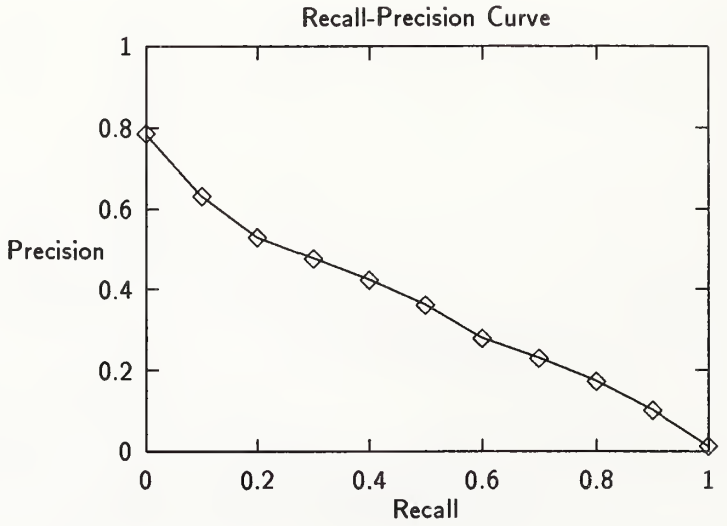
Recall Level Averages	
Recall	Precision
0.00	0.7852
0.10	0.6315
0.20	0.5296
0.30	0.4762
0.40	0.4246
0.50	0.3627
0.60	0.2795
0.70	0.2309
0.80	0.1735
0.90	0.1016
1.00	0.0122
Average precision over all relevant docs	
non-interpolated	0.3538

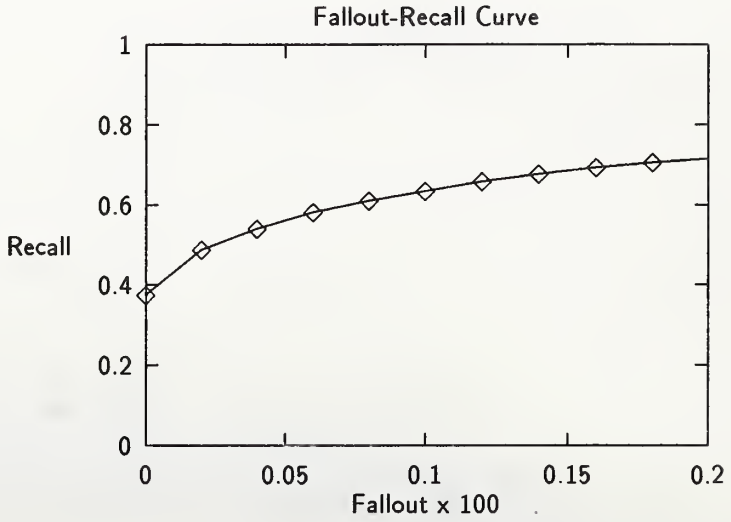
Document Level Averages	
	Precision
At 5 docs	0.6440
At 10 docs	0.6320
At 15 docs	0.6027
At 20 docs	0.5820
At 30 docs	0.5593
At 100 docs	0.4452
At 200 docs	0.3493
At 500 docs	0.2295
At 1000 docs	0.1432
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3893

Fallout Level Averages	
Fallout	Recall
0.00000	0.3748
0.00020	0.4880
0.00040	0.5412
0.00060	0.5823
0.00080	0.6106
0.00100	0.6347
0.00120	0.6576
0.00140	0.6780
0.00160	0.6926
0.00180	0.7063
0.00200	0.7155

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	INQ004–full, manual
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10489
Rel_ret:	7386

Recall Level Averages	
Recall	Precision
0.00	0.8489
0.10	0.6450
0.20	0.5327
0.30	0.4821
0.40	0.4304
0.50	0.3633
0.60	0.3054
0.70	0.2325
0.80	0.1762
0.90	0.1054
1.00	0.0151

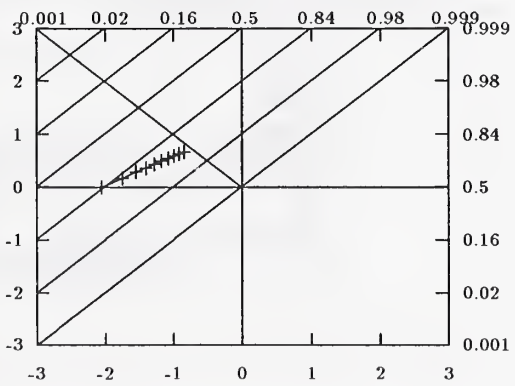
Average precision over all relevant docs	
non-interpolated	0.3612

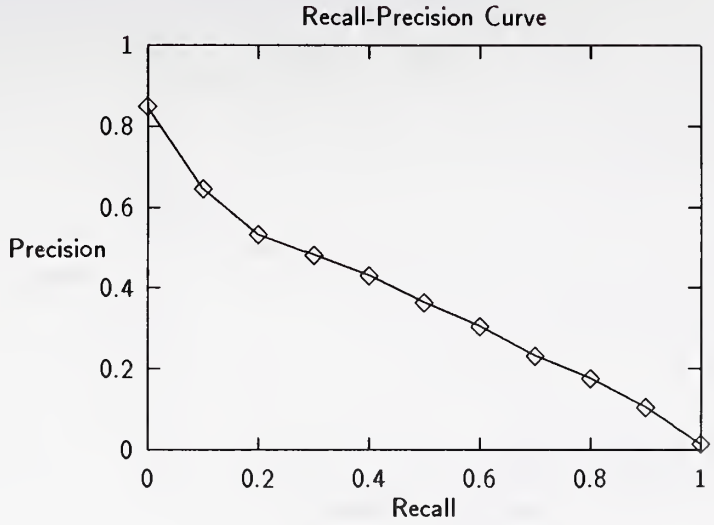
Document Level Averages	
	Precision
At 5 docs	0.6680
At 10 docs	0.6580
At 15 docs	0.6200
At 20 docs	0.5940
At 30 docs	0.5747
At 100 docs	0.4474
At 200 docs	0.3547
At 500 docs	0.2360
At 1000 docs	0.1477

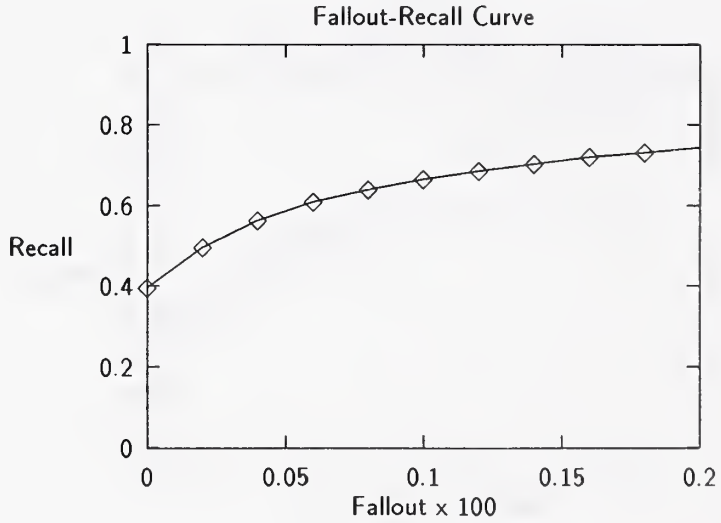
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3951

Fallout Level Averages	
Fallout	Recall
0.00000	0.3956
0.00020	0.4966
0.00040	0.5621
0.00060	0.6083
0.00080	0.6396
0.00100	0.6649
0.00120	0.6860
0.00140	0.7025
0.00160	0.7203
0.00180	0.7313
0.00200	0.7435

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	HNCrt1–full, automatic
Num of Queries	50

Total number of documents over all queries	
Retrieved:	49970
Relevant:	10489
Rel_ret:	5833

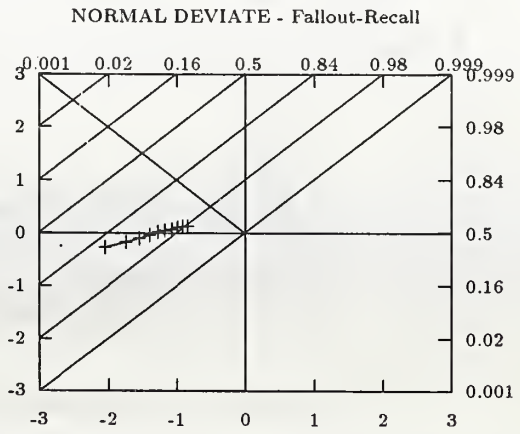
Recall Level Averages	
Recall	Precision
0.00	0.7867
0.10	0.5873
0.20	0.5083
0.30	0.4187
0.40	0.3371
0.50	0.2691
0.60	0.1962
0.70	0.1130
0.80	0.0451
0.90	0.0180
1.00	0.0023

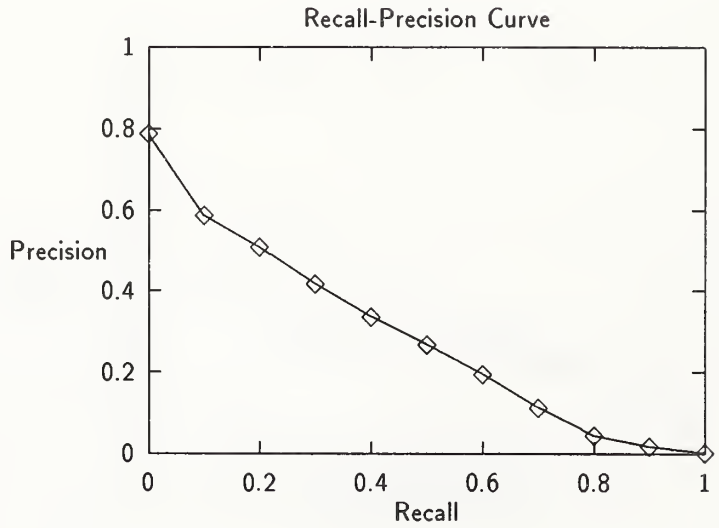
Average precision over all relevant docs	
non-interpolated	0.2810

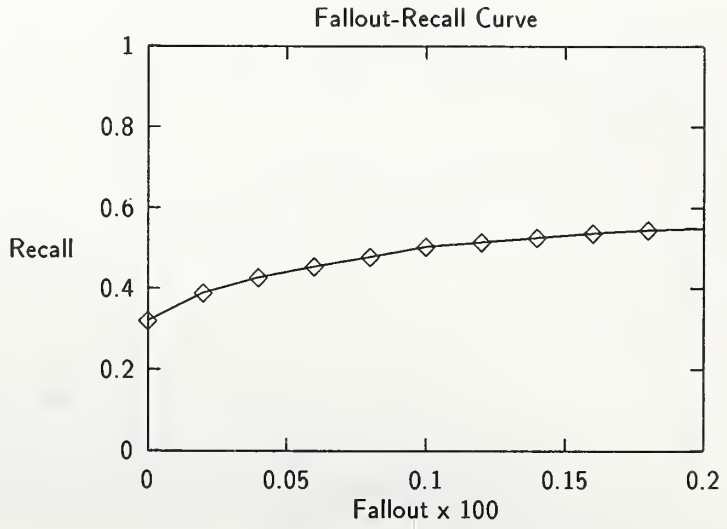
Document Level Averages	
	Precision
At 5 docs	0.6120
At 10 docs	0.6020
At 15 docs	0.5853
At 20 docs	0.5740
At 30 docs	0.5480
At 100 docs	0.4228
At 200 docs	0.3223
At 500 docs	0.1924
At 1000 docs	0.1167

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3383

Fallout Level Averages	
Fallout	Recall
0.00000	0.3211
0.00020	0.3893
0.00040	0.4274
0.00060	0.4554
0.00080	0.4791
0.00100	0.5037
0.00120	0.5152
0.00140	0.5260
0.00160	0.5373
0.00180	0.5458
0.00200	0.5504







Summary Statistics	
Run Number	HNCrt2–full, automatic
Num of Queries	50

Total number of documents over all queries	
Retrieved:	50000
Relevant:	10489
Rel_ret:	6407

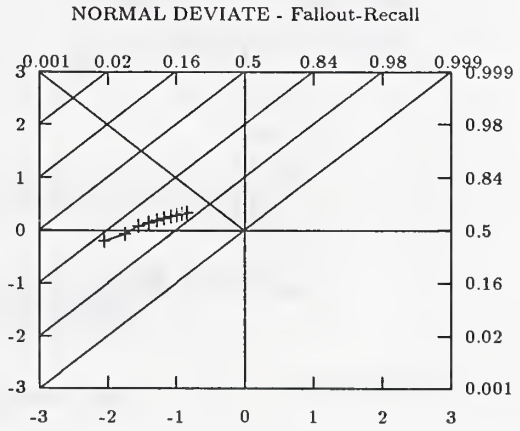
Recall Level Averages	
Recall	Precision
0.00	0.8028
0.10	0.5734
0.20	0.5058
0.30	0.4376
0.40	0.3734
0.50	0.2933
0.60	0.2190
0.70	0.1373
0.80	0.0535
0.90	0.0137
1.00	0.0029

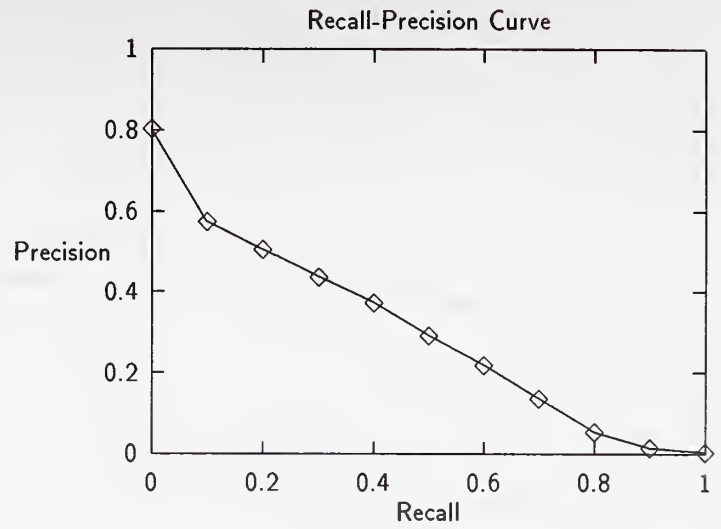
Average precision over all relevant docs	
non-interpolated	0.2927

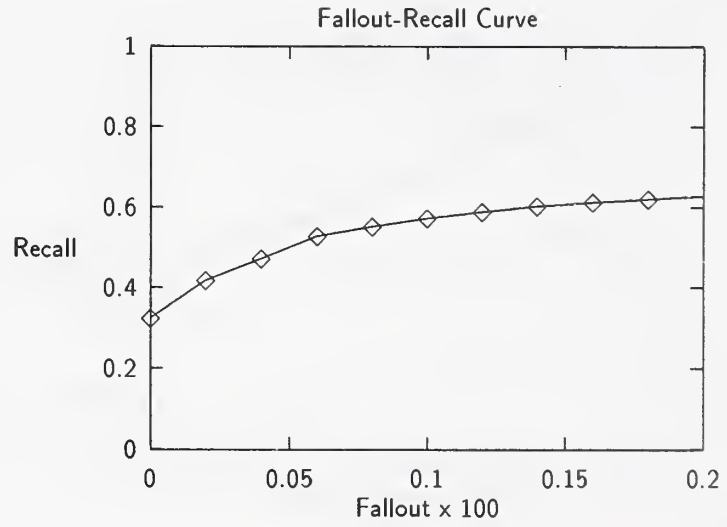
Document Level Averages	
	Precision
At 5 docs	0.6200
At 10 docs	0.6100
At 15 docs	0.5827
At 20 docs	0.5560
At 30 docs	0.5300
At 100 docs	0.4090
At 200 docs	0.3296
At 500 docs	0.2095
At 1000 docs	0.1281

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3638

Fallout Level Averages	
Fallout	Recall
0.00000	0.3238
0.00020	0.4185
0.00040	0.4718
0.00060	0.5285
0.00080	0.5523
0.00100	0.5731
0.00120	0.5884
0.00140	0.6024
0.00160	0.6117
0.00180	0.6200
0.00200	0.6281







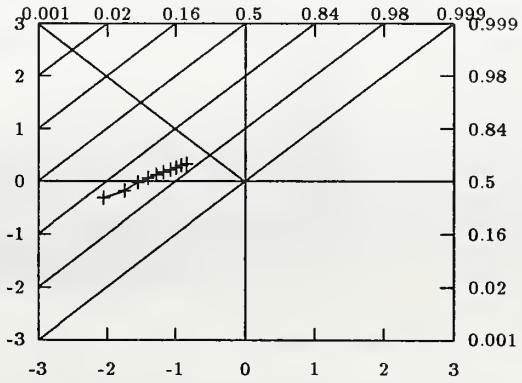
Summary Statistics	
Run Number	TMC6–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	46075
Relevant:	10489
Rel_ret:	6067

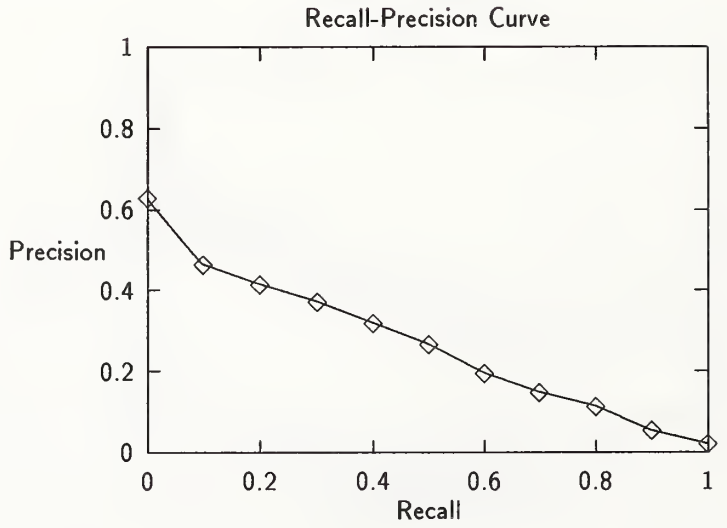
Recall Level Averages	
Recall	Precision
0.00	0.6276
0.10	0.4638
0.20	0.4142
0.30	0.3724
0.40	0.3194
0.50	0.2670
0.60	0.1950
0.70	0.1482
0.80	0.1123
0.90	0.0547
1.00	0.0211
Average precision over all relevant docs	
non-interpolated	0.2553

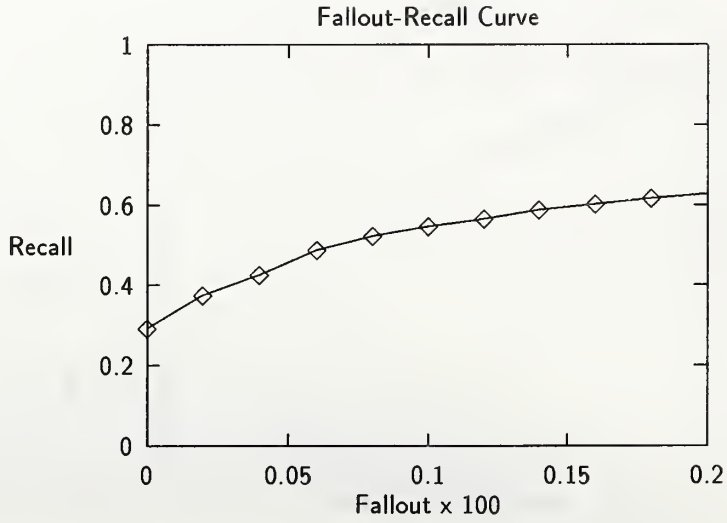
Document Level Averages	
	Precision
At 5 docs	0.4320
At 10 docs	0.4180
At 15 docs	0.4187
At 20 docs	0.4200
At 30 docs	0.4060
At 100 docs	0.3396
At 200 docs	0.2839
At 500 docs	0.1868
At 1000 docs	0.1213
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2991

Fallout Level Averages	
Fallout	Recall
0.00000	0.2924
0.00020	0.3747
0.00040	0.4267
0.00060	0.4887
0.00080	0.5237
0.00100	0.5472
0.00120	0.5666
0.00140	0.5883
0.00160	0.6021
0.00180	0.6172
0.00200	0.6274

NORMAL DEVIATE - Fallout-Recall







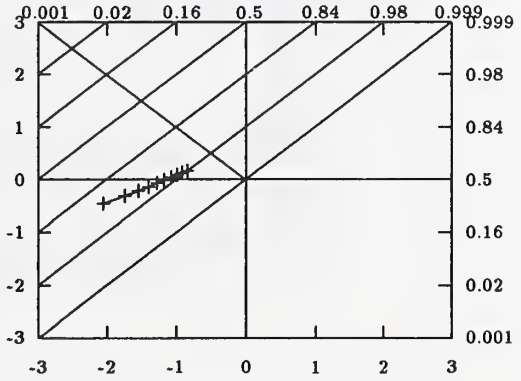
Summary Statistics	
Run Number	TMC7–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	46722
Relevant:	10489
Rel_ret:	5712

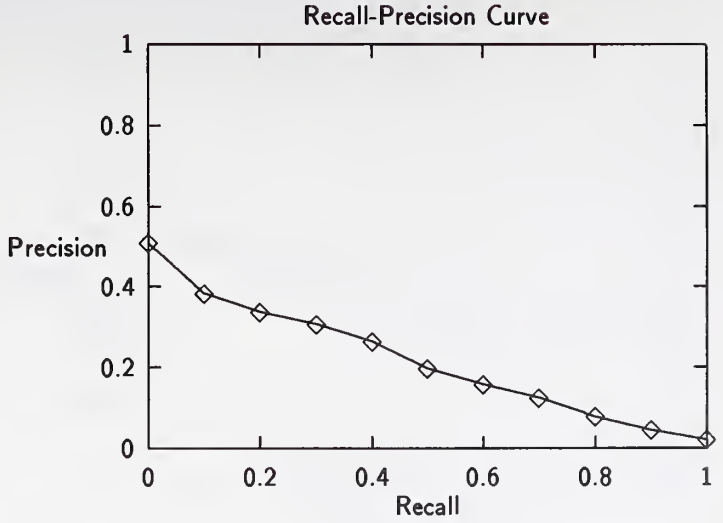
Recall Level Averages	
Recall	Precision
0.00	0.5085
0.10	0.3828
0.20	0.3378
0.30	0.3064
0.40	0.2632
0.50	0.1964
0.60	0.1581
0.70	0.1240
0.80	0.0784
0.90	0.0453
1.00	0.0202
Average precision over all relevant docs	
non-interpolated	0.2045

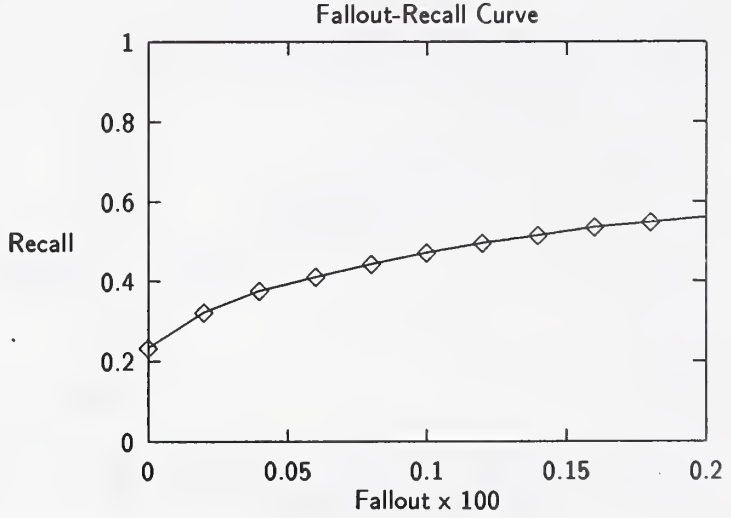
Document Level Averages	
	Precision
At 5 docs	0.3120
At 10 docs	0.3360
At 15 docs	0.3347
At 20 docs	0.3360
At 30 docs	0.3287
At 100 docs	0.2920
At 200 docs	0.2440
At 500 docs	0.1688
At 1000 docs	0.1142
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2564

Fallout Level Averages	
Fallout	Recall
0.00000	0.2333
0.00020	0.3224
0.00040	0.3774
0.00060	0.4123
0.00080	0.4433
0.00100	0.4727
0.00120	0.4967
0.00140	0.5158
0.00160	0.5360
0.00180	0.5482
0.00200	0.5617

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	lsir1–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10489
Rel_ret:	6522

Recall Level Averages	
Recall	Precision
0.00	0.7569
0.10	0.5281
0.20	0.4302
0.30	0.3779
0.40	0.3222
0.50	0.2446
0.60	0.1935
0.70	0.1474
0.80	0.0847
0.90	0.0295
1.00	0.0043

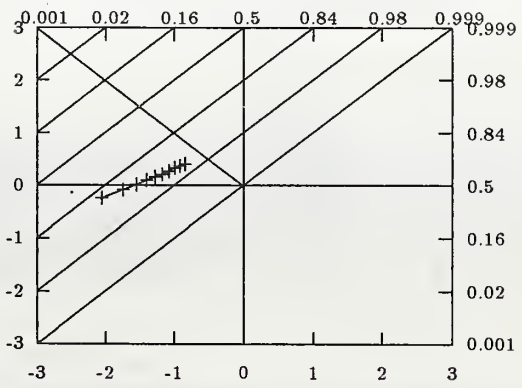
Average precision over all relevant docs	
non-interpolated	0.2662

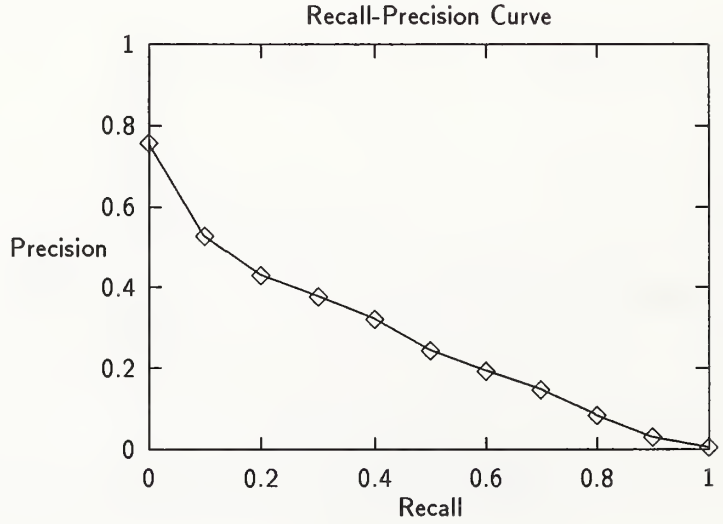
Document Level Averages	
	Precision
At 5 docs	0.6120
At 10 docs	0.5480
At 15 docs	0.5387
At 20 docs	0.5110
At 30 docs	0.4880
At 100 docs	0.3798
At 200 docs	0.3037
At 500 docs	0.2011
At 1000 docs	0.1304

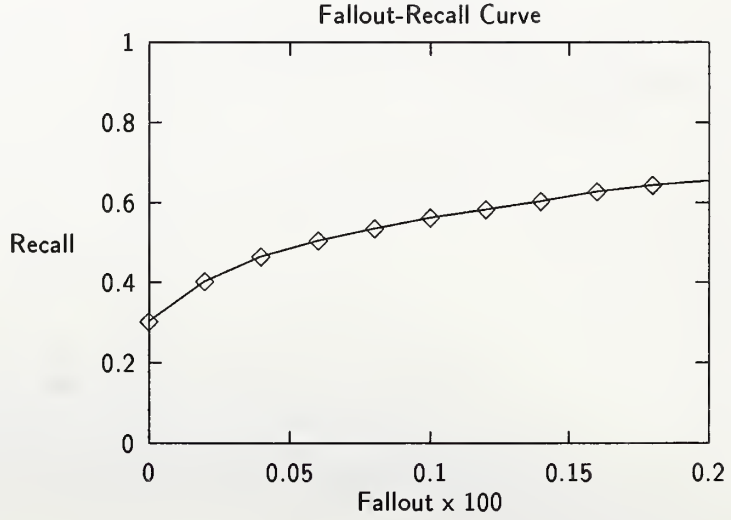
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3050

Fallout Level Averages	
Fallout	Recall
0.00000	0.3042
0.00020	0.4031
0.00040	0.4660
0.00060	0.5058
0.00080	0.5365
0.00100	0.5635
0.00120	0.5838
0.00140	0.6034
0.00160	0.6276
0.00180	0.6431
0.00200	0.6554

NORMAL DEVIATE - Fallout-Recall





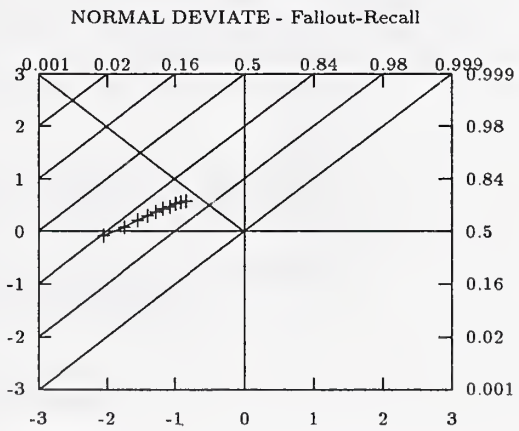


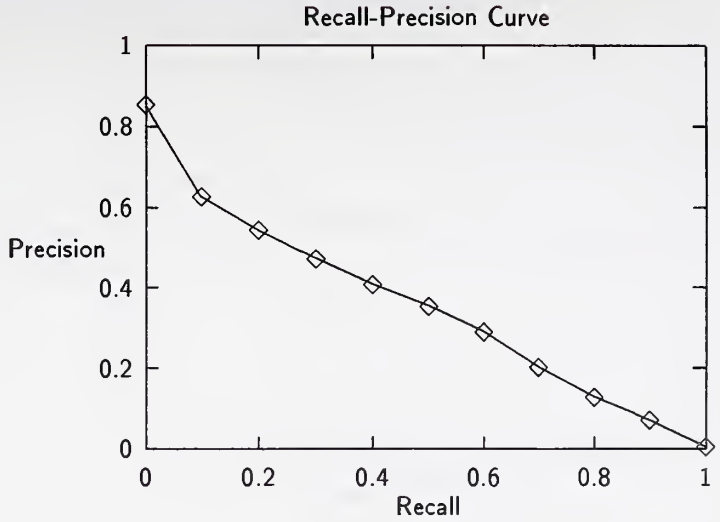
Summary Statistics	
Run Number	lsir2–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10489
Rel_ret:	7155

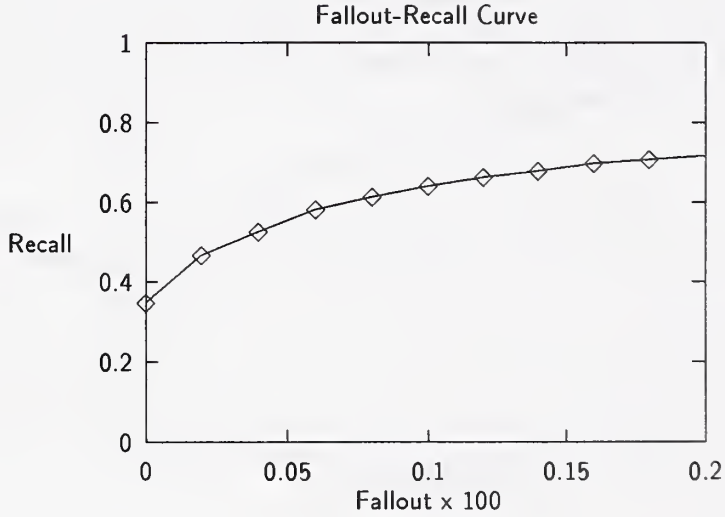
Recall Level Averages	
Recall	Precision
0.00	0.8533
0.10	0.6270
0.20	0.5435
0.30	0.4722
0.40	0.4080
0.50	0.3546
0.60	0.2913
0.70	0.2032
0.80	0.1282
0.90	0.0717
1.00	0.0044
Average precision over all relevant docs	
non-interpolated	0.3442

Document Level Averages	
	Precision
At 5 docs	0.6960
At 10 docs	0.6660
At 15 docs	0.6253
At 20 docs	0.5960
At 30 docs	0.5740
At 100 docs	0.4524
At 200 docs	0.3562
At 500 docs	0.2274
At 1000 docs	0.1431
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3804

Fallout Level Averages	
Fallout	Recall
0.00000	0.3478
0.00020	0.4676
0.00040	0.5269
0.00060	0.5817
0.00080	0.6136
0.00100	0.6407
0.00120	0.6632
0.00140	0.6792
0.00160	0.6973
0.00180	0.7079
0.00200	0.7174







Summary Statistics	
Run Number	nyuir1–category B, automatic
Num of Queries	50

Total number of documents over all queries	
Retrieved:	50000
Relevant:	2064
Rel_ret:	1390

Recall Level Averages	
Recall	Precision
0.00	0.5400
0.10	0.3937
0.20	0.3423
0.30	0.2572
0.40	0.2263
0.50	0.2032
0.60	0.1674
0.70	0.1295
0.80	0.0905
0.90	0.0442
1.00	0.0284

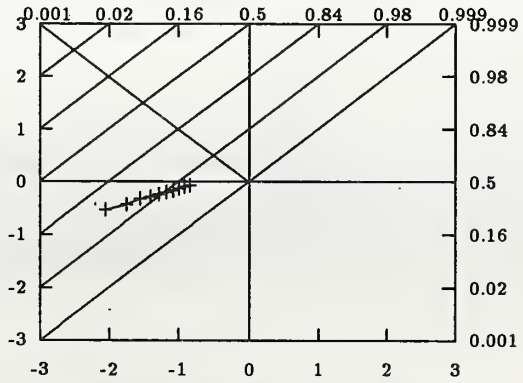
Average precision over all relevant docs	
non-interpolated	0.2038

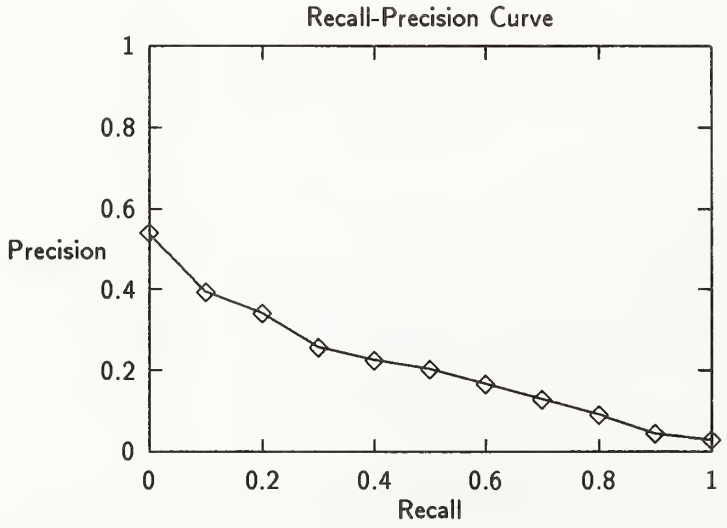
Document Level Averages	
	Precision
At 5 docs	0.3360
At 10 docs	0.3240
At 15 docs	0.2933
At 20 docs	0.2790
At 30 docs	0.2407
At 100 docs	0.1412
At 200 docs	0.0939
At 500 docs	0.0489
At 1000 docs	0.0278

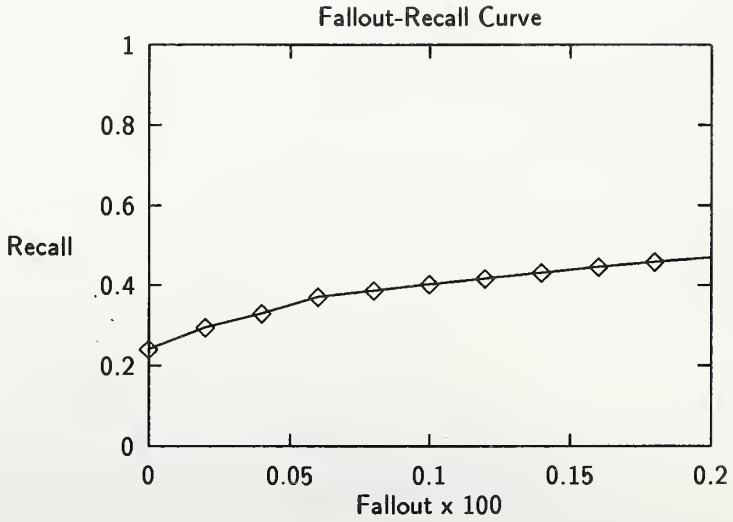
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2267

Fallout Level Averages	
Fallout	Recall
0.00000	0.2413
0.00020	0.2950
0.00040	0.3314
0.00060	0.3726
0.00080	0.3872
0.00100	0.4046
0.00120	0.4181
0.00140	0.4328
0.00160	0.4475
0.00180	0.4600
0.00200	0.4698

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	nyuir2–category B, automatic
Num of Queries	50

Total number of documents over all queries	
Retrieved:	50000
Relevant:	2064
Rel_ret:	1610

Recall Level Averages	
Recall	Precision
0.00	0.6435
0.10	0.4610
0.20	0.3705
0.30	0.3031
0.40	0.2637
0.50	0.2282
0.60	0.1934
0.70	0.1542
0.80	0.1002
0.90	0.0456
1.00	0.0186

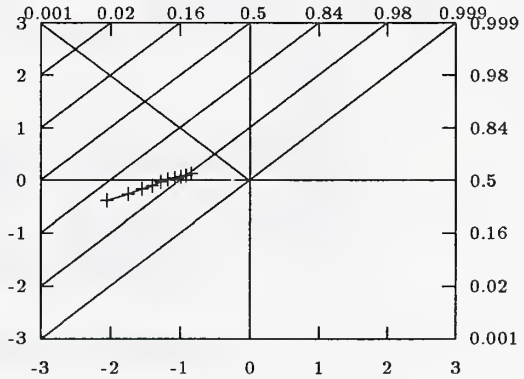
Average precision over all relevant docs	
non-interpolated	0.2337

Document Level Averages	
	Precision
At 5 docs	0.4280
At 10 docs	0.4000
At 15 docs	0.3613
At 20 docs	0.3260
At 30 docs	0.2740
At 100 docs	0.1708
At 200 docs	0.1078
At 500 docs	0.0575
At 1000 docs	0.0322

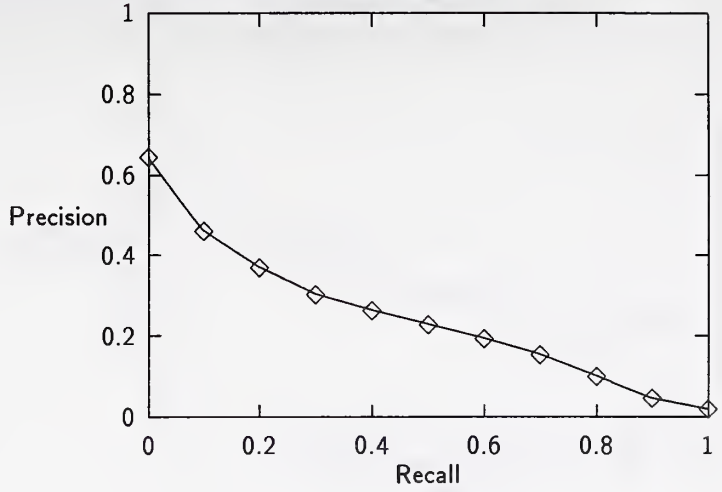
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2513

Fallout Level Averages	
Fallout	Recall
0.00000	0.2600
0.00020	0.3484
0.00040	0.3946
0.00060	0.4336
0.00080	0.4551
0.00100	0.4858
0.00120	0.5033
0.00140	0.5126
0.00160	0.5246
0.00180	0.5355
0.00200	0.5486

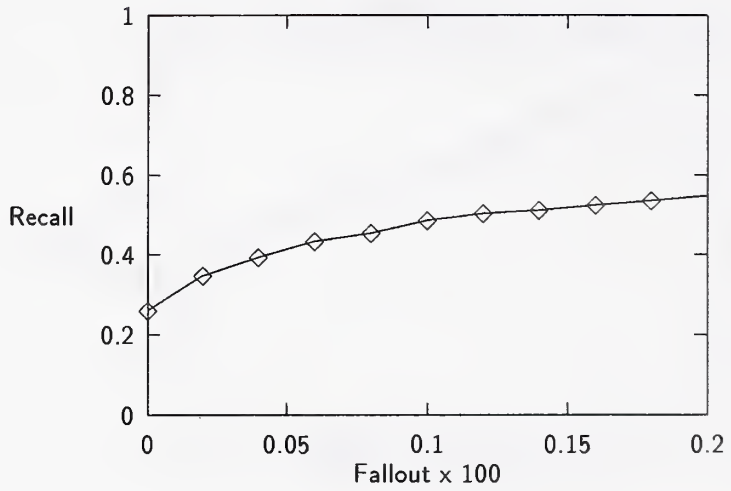
NORMAL DEVIATE - Fallout-Recall



Recall-Precision Curve



Fallout-Recall Curve



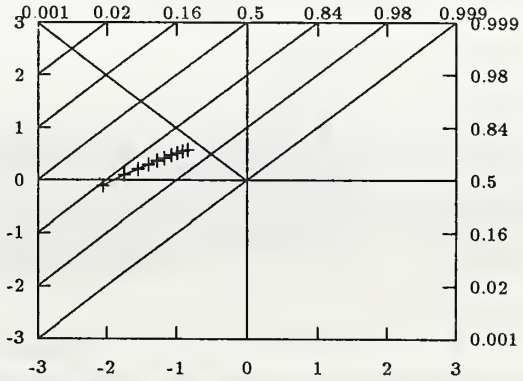
Summary Statistics	
Run Number	CLARTA–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10489
Rel_ret:	6811

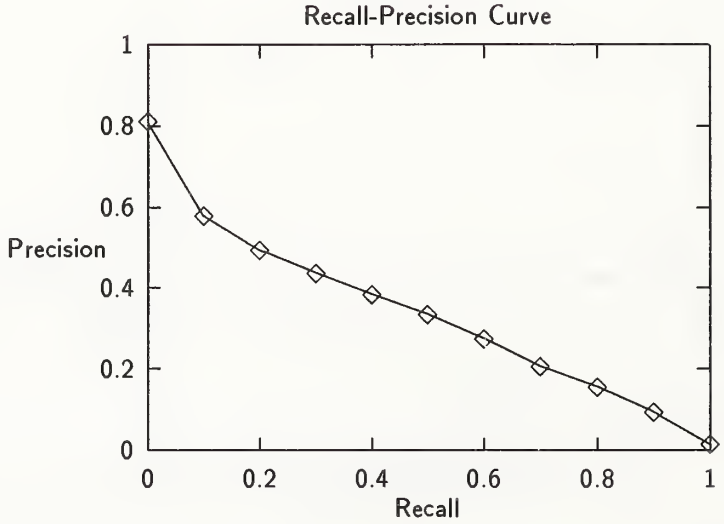
Recall Level Averages	
Recall	Precision
0.00	0.8101
0.10	0.5781
0.20	0.4941
0.30	0.4368
0.40	0.3851
0.50	0.3358
0.60	0.2749
0.70	0.2074
0.80	0.1561
0.90	0.0933
1.00	0.0141
Average precision over all relevant docs	
non-interpolated	0.3269

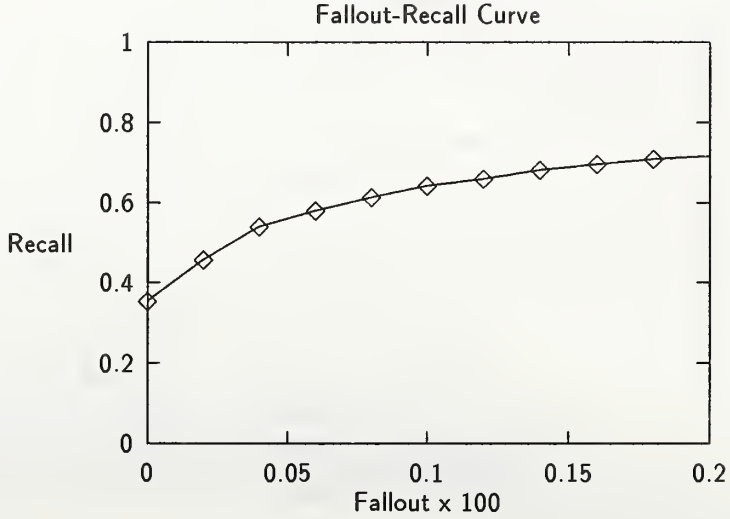
Document Level Averages	
	Precision
At 5 docs	0.6200
At 10 docs	0.6060
At 15 docs	0.5893
At 20 docs	0.5730
At 30 docs	0.5460
At 100 docs	0.4346
At 200 docs	0.3416
At 500 docs	0.2184
At 1000 docs	0.1362
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3646

Fallout Level Averages	
Fallout	Recall
0.00000	0.3545
0.00020	0.4571
0.00040	0.5408
0.00060	0.5801
0.00080	0.6132
0.00100	0.6422
0.00120	0.6603
0.00140	0.6822
0.00160	0.6959
0.00180	0.7096
0.00200	0.7172

NORMAL DEVIATE - Fallout-Recall





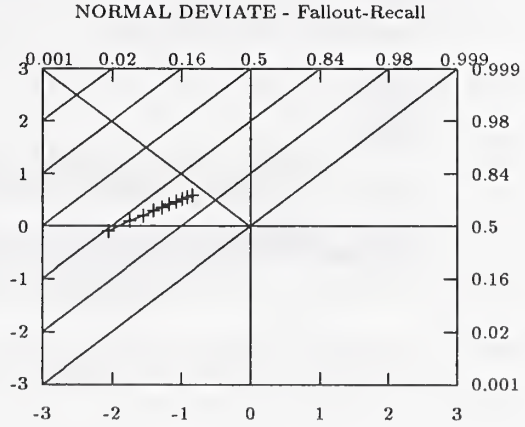


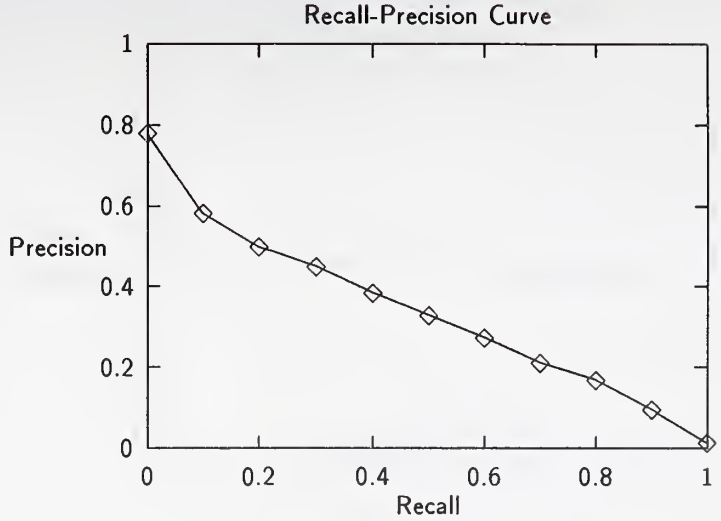
Summary Statistics	
Run Number	CLARTM–full, manual
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10489
Rel_ret:	6785

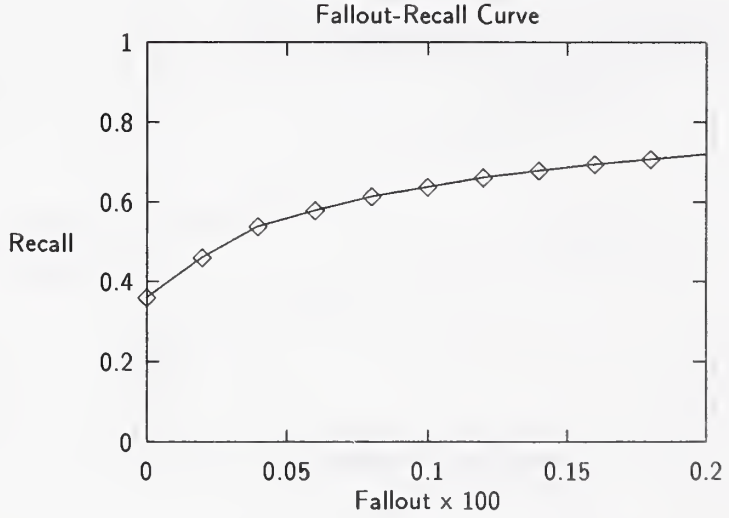
Recall Level Averages	
Recall	Precision
0.00	0.7791
0.10	0.5825
0.20	0.4987
0.30	0.4505
0.40	0.3848
0.50	0.3295
0.60	0.2739
0.70	0.2109
0.80	0.1683
0.90	0.0951
1.00	0.0126
Average precision over all relevant docs	
non-interpolated	0.3302

Document Level Averages	
	Precision
At 5 docs	0.6120
At 10 docs	0.5940
At 15 docs	0.5800
At 20 docs	0.5700
At 30 docs	0.5527
At 100 docs	0.4396
At 200 docs	0.3436
At 500 docs	0.2176
At 1000 docs	0.1357
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3642

Fallout Level Averages	
Fallout	Recall
0.00000	0.3602
0.00020	0.4615
0.00040	0.5383
0.00060	0.5783
0.00080	0.6143
0.00100	0.6371
0.00120	0.6611
0.00140	0.6781
0.00160	0.6946
0.00180	0.7071
0.00200	0.7199







Summary Statistics	
Run Number	idsra2–category B, partial topics, manual
Num of Queries	10

Total number of documents over all queries	
Retrieved:	10000
Relevant:	423
Rel_ret:	267

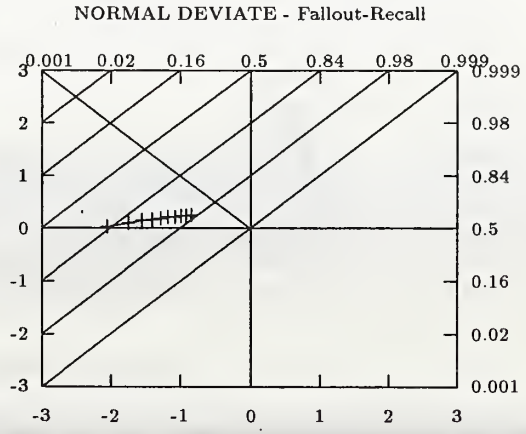
Recall Level Averages	
Recall	Precision
0.00	0.5279
0.10	0.4577
0.20	0.4183
0.30	0.3665
0.40	0.3159
0.50	0.2956
0.60	0.2425
0.70	0.1672
0.80	0.0971
0.90	0.0115
1.00	0.0000

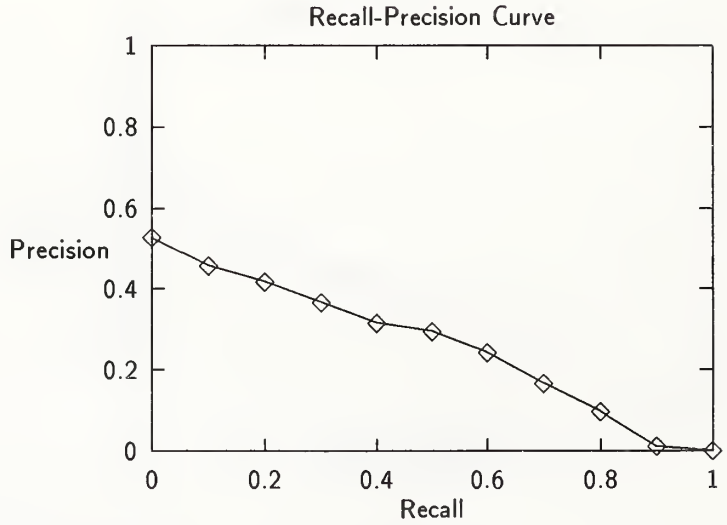
Average precision over all relevant docs	
non-interpolated	0.2504

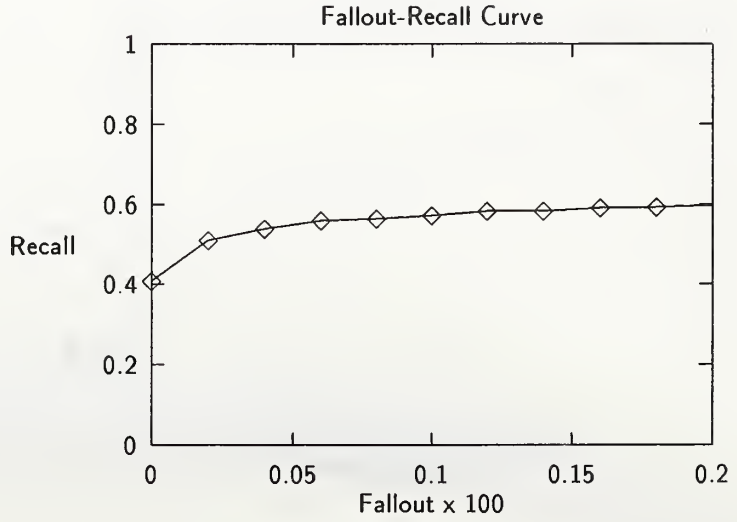
Document Level Averages	
	Precision
At 5 docs	0.4000
At 10 docs	0.4200
At 15 docs	0.3933
At 20 docs	0.3600
At 30 docs	0.3333
At 100 docs	0.1870
At 200 docs	0.1040
At 500 docs	0.0482
At 1000 docs	0.0267

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3085

Fallout Level Averages	
Fallout	Recall
0.00000	0.4079
0.00020	0.5099
0.00040	0.5397
0.00060	0.5591
0.00080	0.5642
0.00100	0.5717
0.00120	0.5834
0.00140	0.5834
0.00160	0.5920
0.00180	0.5927
0.00200	0.5979







Summary Statistics	
Run Number	schau1–full, automatic
Num of Queries	50

Total number of documents over all queries	
Retrieved:	50000
Relevant:	10489
Rel_ret:	5345

Recall Level Averages	
Recall	Precision
0.00	0.5925
0.10	0.4192
0.20	0.3678
0.30	0.3083
0.40	0.2453
0.50	0.1982
0.60	0.1319
0.70	0.0824
0.80	0.0485
0.90	0.0165
1.00	0.0023

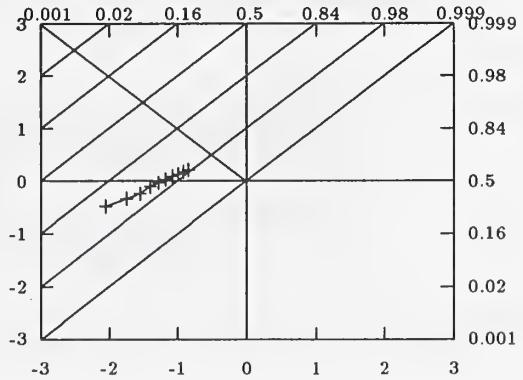
Average precision over all relevant docs	
non-interpolated	0.1994

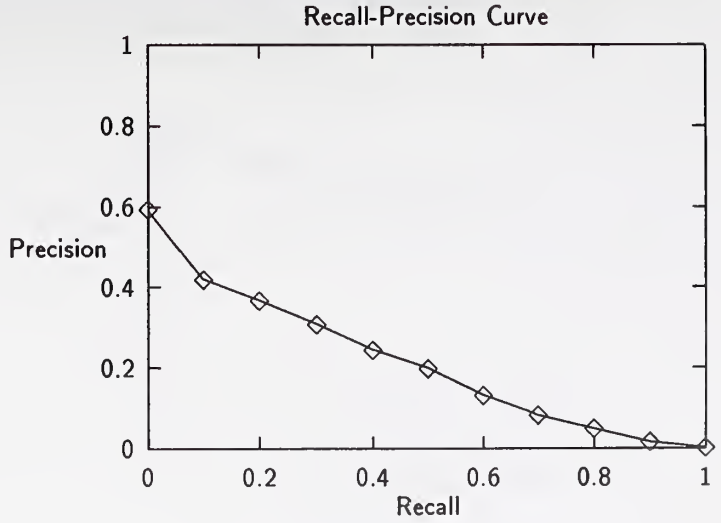
Document Level Averages	
	Precision
At 5 docs	0.4000
At 10 docs	0.3960
At 15 docs	0.3680
At 20 docs	0.3550
At 30 docs	0.3500
At 100 docs	0.2910
At 200 docs	0.2350
At 500 docs	0.1567
At 1000 docs	0.1069

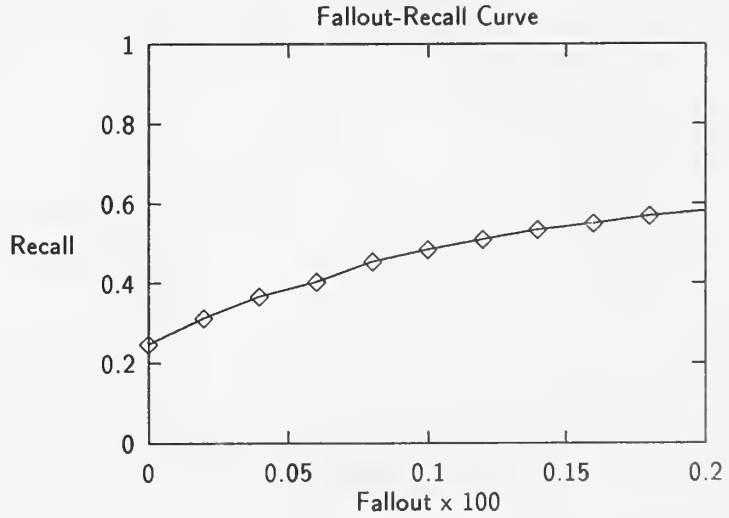
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2708

Fallout Level Averages	
Fallout	Recall
0.00000	0.2488
0.00020	0.3126
0.00040	0.3678
0.00060	0.4031
0.00080	0.4549
0.00100	0.4855
0.00120	0.5102
0.00140	0.5341
0.00160	0.5507
0.00180	0.5684
0.00200	0.5811

NORMAL DEVIATE - Fallout-Recall







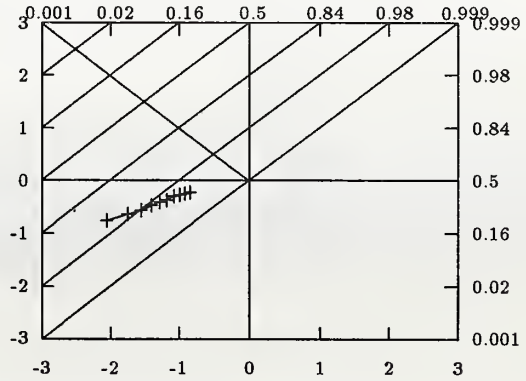
Summary Statistics	
Run Number	erimr1–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	47829
Relevant:	10489
Rel_ret:	4152

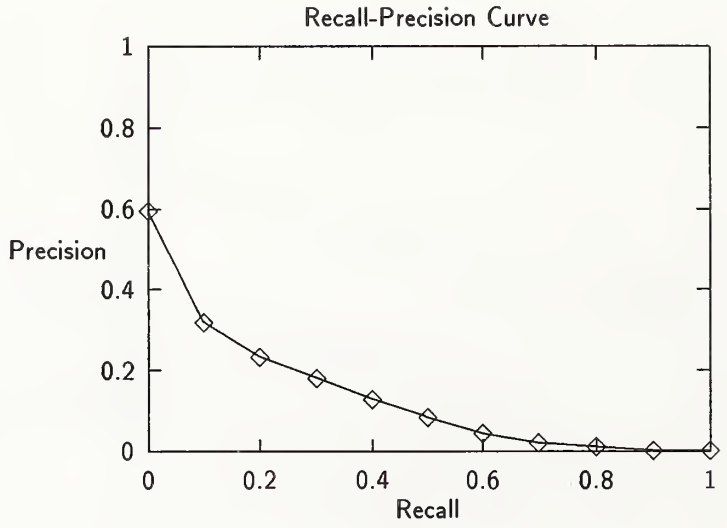
Recall Level Averages	
Recall	Precision
0.00	0.5948
0.10	0.3190
0.20	0.2334
0.30	0.1817
0.40	0.1280
0.50	0.0841
0.60	0.0441
0.70	0.0210
0.80	0.0115
0.90	0.0020
1.00	0.0020
Average precision over all relevant docs	
non-interpolated	0.1219

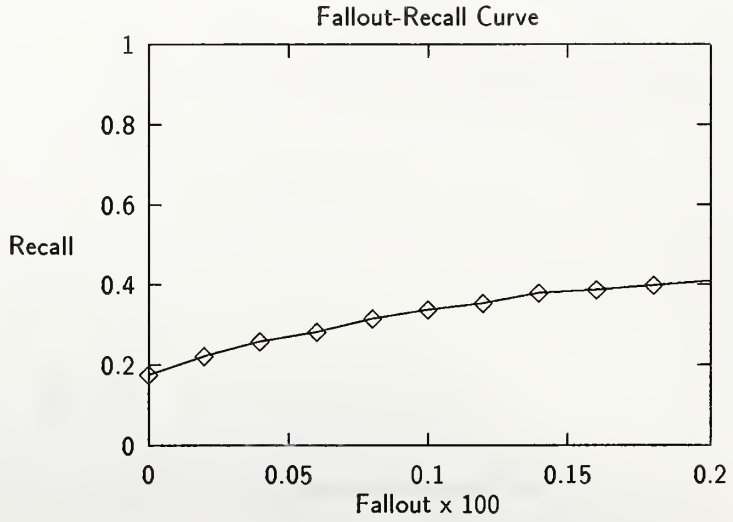
Document Level Averages	
	Precision
At 5 docs	0.3720
At 10 docs	0.3580
At 15 docs	0.3347
At 20 docs	0.3340
At 30 docs	0.3113
At 100 docs	0.2304
At 200 docs	0.1802
At 500 docs	0.1203
At 1000 docs	0.0830
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1814

Fallout Level Averages	
Fallout	Recall
0.00000	0.1771
0.00020	0.2227
0.00040	0.2587
0.00060	0.2834
0.00080	0.3169
0.00100	0.3383
0.00120	0.3545
0.00140	0.3794
0.00160	0.3886
0.00180	0.3995
0.00200	0.4104

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	erimr2–full, automatic
Num of Queries	50

Total number of documents over all queries	
Retrieved:	45274
Relevant:	10489
Rel_ret:	4475

Recall Level Averages	
Recall	Precision
0.00	0.6448
0.10	0.3426
0.20	0.2643
0.30	0.2098
0.40	0.1551
0.50	0.1116
0.60	0.0568
0.70	0.0300
0.80	0.0168
0.90	0.0074
1.00	0.0006

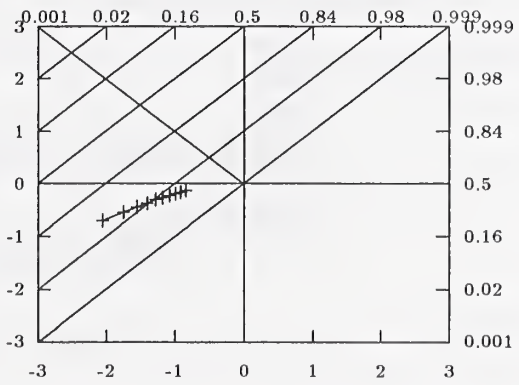
Average precision over all relevant docs	
non-interpolated	0.1415

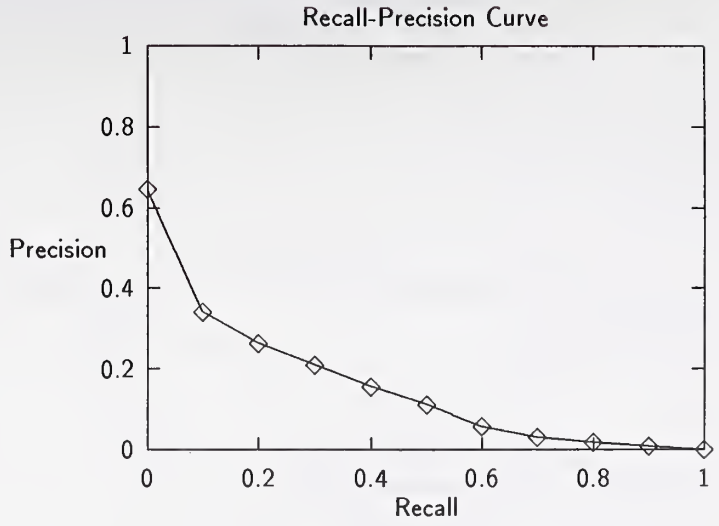
Document Level Averages	
	Precision
At 5 docs	0.4720
At 10 docs	0.4240
At 15 docs	0.3933
At 20 docs	0.3810
At 30 docs	0.3420
At 100 docs	0.2524
At 200 docs	0.2012
At 500 docs	0.1394
At 1000 docs	0.0895

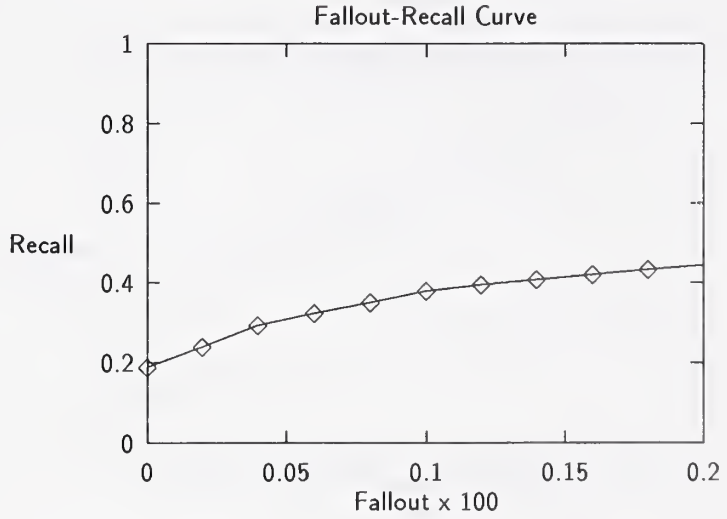
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2031

Fallout Level Averages	
Fallout	Recall
0.00000	0.1891
0.00020	0.2408
0.00040	0.2940
0.00060	0.3245
0.00080	0.3517
0.00100	0.3797
0.00120	0.3958
0.00140	0.4092
0.00160	0.4219
0.00180	0.4343
0.00200	0.4445

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	gecrd1–full, manual
Num of Queries	50

Total number of documents over all queries	
Retrieved:	44004
Relevant:	10489
Rel_ret:	6178

Recall Level Averages	
Recall	Precision
0.00	0.8172
0.10	0.6137
0.20	0.5320
0.30	0.4578
0.40	0.3988
0.50	0.3305
0.60	0.2798
0.70	0.2196
0.80	0.1393
0.90	0.0868
1.00	0.0130

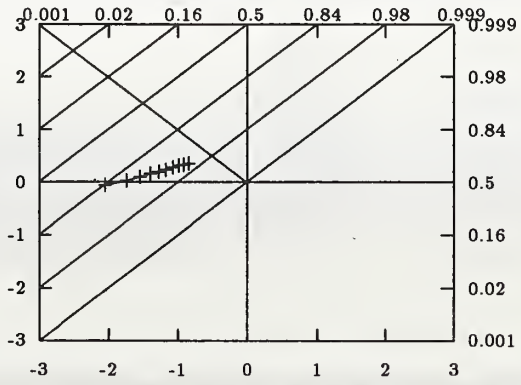
Average precision over all relevant docs	
non-interpolated	0.3308

Document Level Averages	
	Precision
At 5 docs	0.6080
At 10 docs	0.6060
At 15 docs	0.5827
At 20 docs	0.5660
At 30 docs	0.5440
At 100 docs	0.4528
At 200 docs	0.3364
At 500 docs	0.2045
At 1000 docs	0.1236

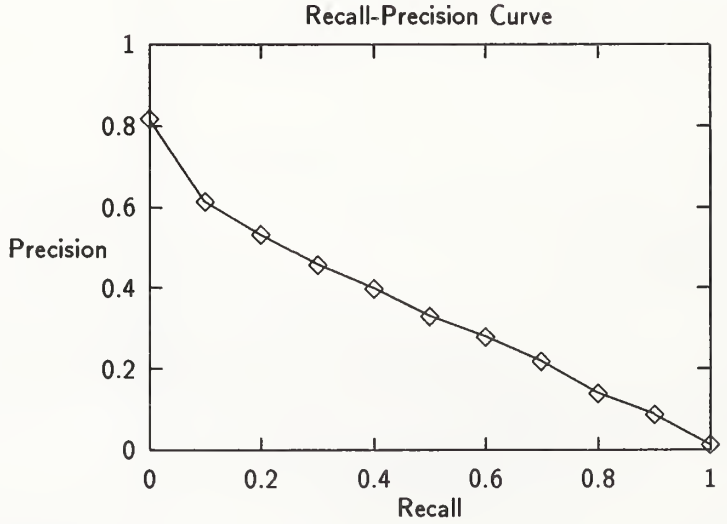
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3872

Fallout Level Averages	
Fallout	Recall
0.00000	0.3899
0.00020	0.4747
0.00040	0.5113
0.00060	0.5400
0.00080	0.5625
0.00100	0.5786
0.00120	0.5932
0.00140	0.6086
0.00160	0.6220
0.00180	0.6297
0.00200	0.6387

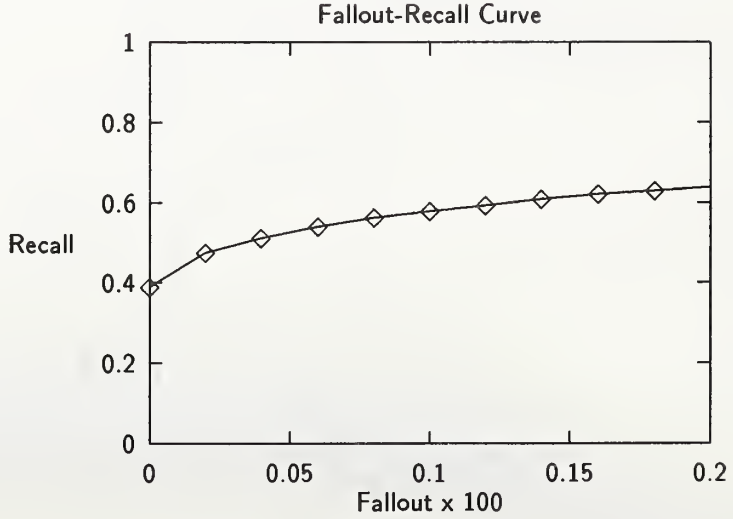
NORMAL DEVIATE - Fallout-Recall



Recall-Precision Curve



Fallout-Recall Curve



Summary Statistics	
Run Number	gecrd2–full, automatic
Num of Queries	50

Total number of documents over all queries	
Retrieved:	49267
Relevant:	10489
Rel_ret:	4485

Recall Level Averages	
Recall	Precision
0.00	0.6802
0.10	0.3796
0.20	0.3117
0.30	0.2581
0.40	0.2108
0.50	0.1600
0.60	0.1075
0.70	0.0611
0.80	0.0360
0.90	0.0150
1.00	0.0009

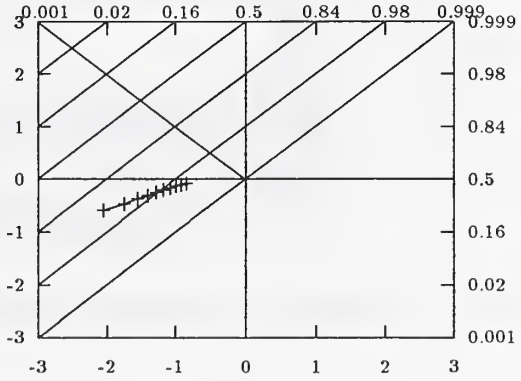
Average precision over all relevant docs	
non-interpolated	0.1772

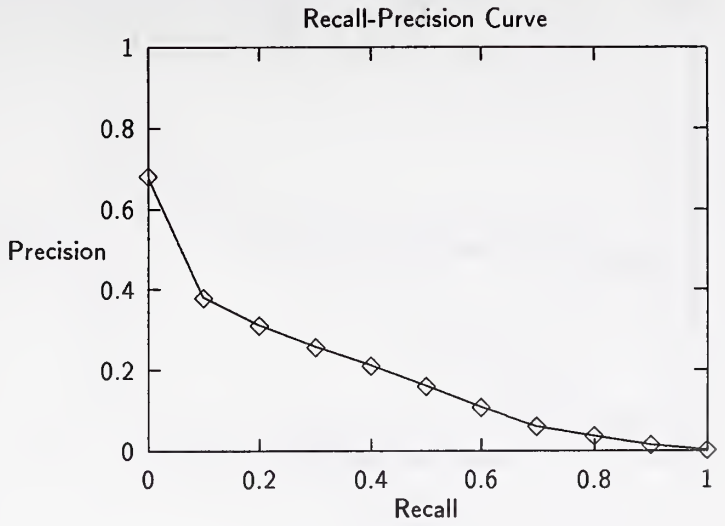
Document Level Averages	
	Precision
At 5 docs	0.4760
At 10 docs	0.4460
At 15 docs	0.4280
At 20 docs	0.4130
At 30 docs	0.3907
At 100 docs	0.2930
At 200 docs	0.2275
At 500 docs	0.1385
At 1000 docs	0.0897

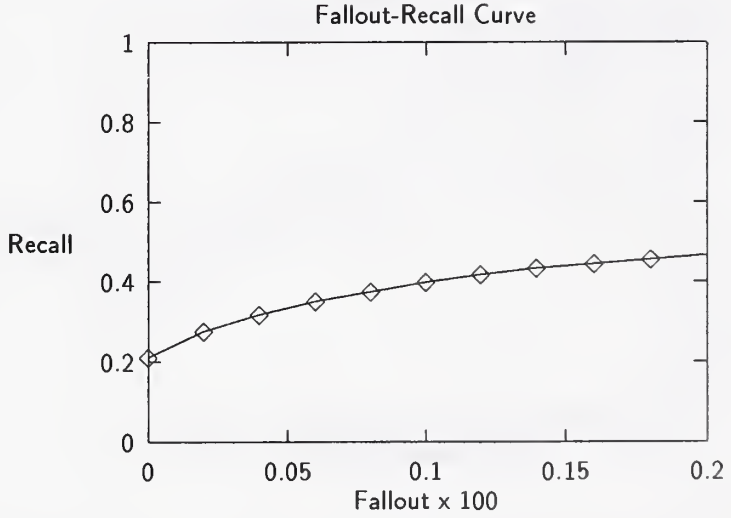
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2342

Fallout Level Averages	
Fallout	Recall
0.00000	0.2109
0.00020	0.2771
0.00040	0.3179
0.00060	0.3513
0.00080	0.3756
0.00100	0.3987
0.00120	0.4178
0.00140	0.4340
0.00160	0.4458
0.00180	0.4559
0.00200	0.4671

NORMAL DEVIATE - Fallout-Recall







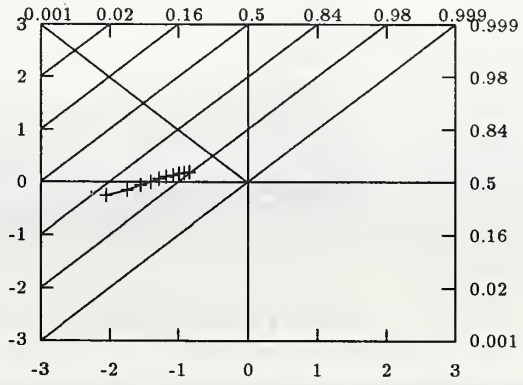
Summary Statistics	
Run Number	trw1–full, manual
Num of Queries	50
Total number of documents over all queries	
Retrieved:	36485
Relevant:	10489
Rel_ret:	5681

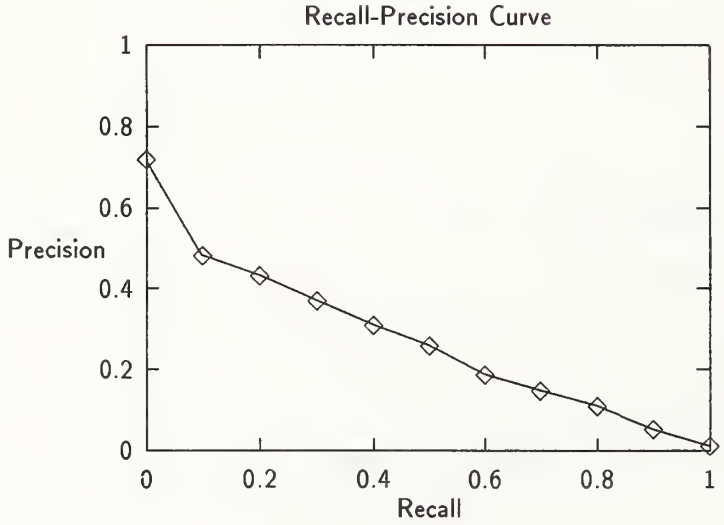
Recall Level Averages	
Recall	Precision
0.00	0.7181
0.10	0.4823
0.20	0.4319
0.30	0.3706
0.40	0.3093
0.50	0.2586
0.60	0.1878
0.70	0.1471
0.80	0.1094
0.90	0.0528
1.00	0.0113
Average precision over all relevant docs	
non-interpolated	0.2525

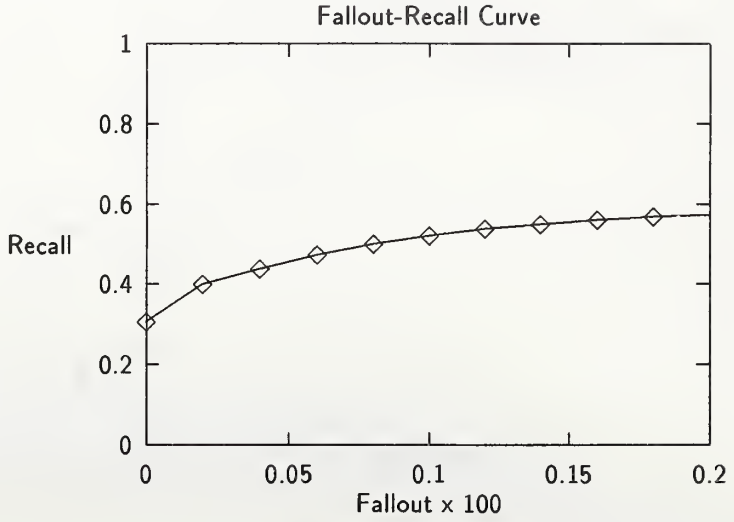
Document Level Averages	
	Precision
At 5 docs	0.4640
At 10 docs	0.4700
At 15 docs	0.4507
At 20 docs	0.4300
At 30 docs	0.4247
At 100 docs	0.3770
At 200 docs	0.2929
At 500 docs	0.1847
At 1000 docs	0.1136
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3220

Fallout Level Averages	
Fallout	Recall
0.00000	0.3071
0.00020	0.4006
0.00040	0.4389
0.00060	0.4745
0.00080	0.5004
0.00100	0.5218
0.00120	0.5389
0.00140	0.5507
0.00160	0.5614
0.00180	0.5685
0.00200	0.5743

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	trw2–full, manual
Num of Queries	50
Total number of documents over all queries	
Retrieved:	48432
Relevant:	10489
Rel_ret:	7383

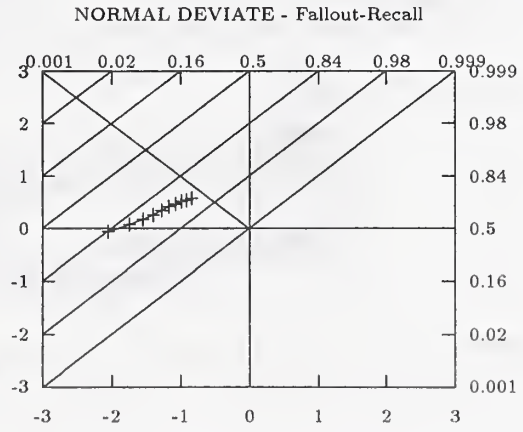
Recall Level Averages	
Recall	Precision
0.00	0.8320
0.10	0.6329
0.20	0.5431
0.30	0.4694
0.40	0.4096
0.50	0.3558
0.60	0.2848
0.70	0.2132
0.80	0.1499
0.90	0.0899
1.00	0.0210

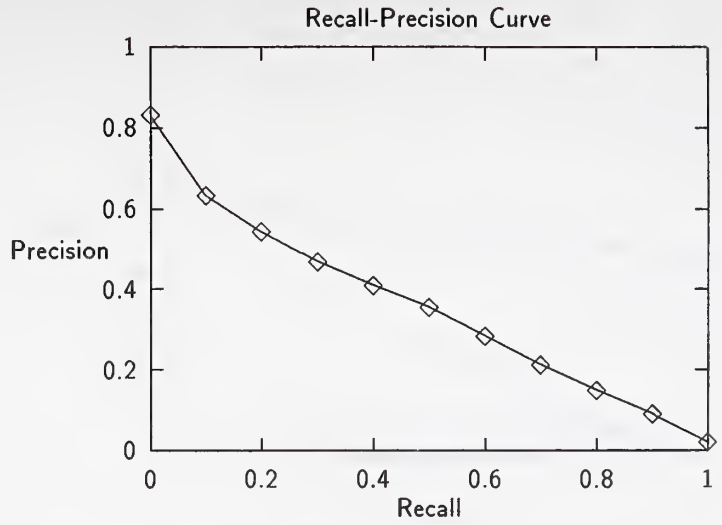
Average precision over all relevant docs	
non-interpolated	0.3459

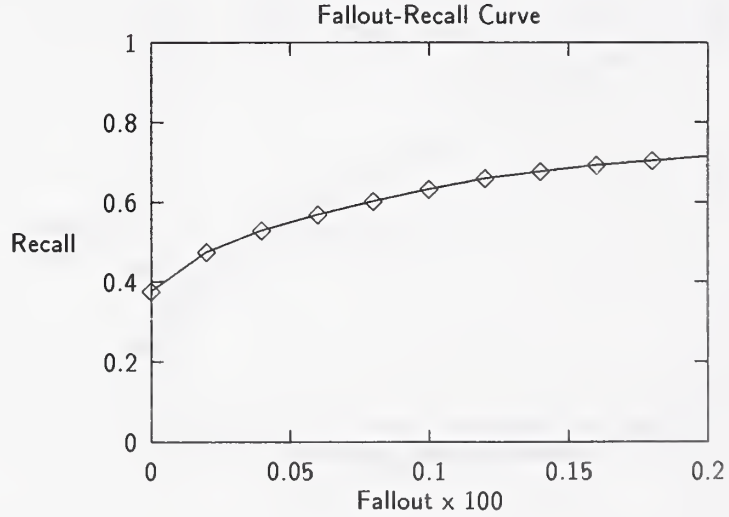
Document Level Averages	
	Precision
At 5 docs	0.6680
At 10 docs	0.6240
At 15 docs	0.6107
At 20 docs	0.5810
At 30 docs	0.5453
At 100 docs	0.4554
At 200 docs	0.3623
At 500 docs	0.2350
At 1000 docs	0.1477

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3807

Fallout Level Averages	
Fallout	Recall
0.00000	0.3768
0.00020	0.4753
0.00040	0.5289
0.00060	0.5694
0.00080	0.6029
0.00100	0.6321
0.00120	0.6599
0.00140	0.6770
0.00160	0.6930
0.00180	0.7035
0.00200	0.7147





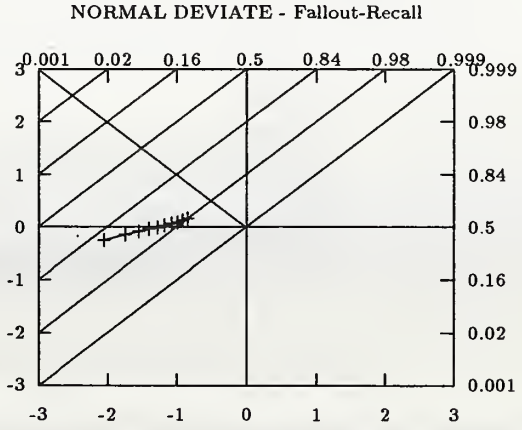


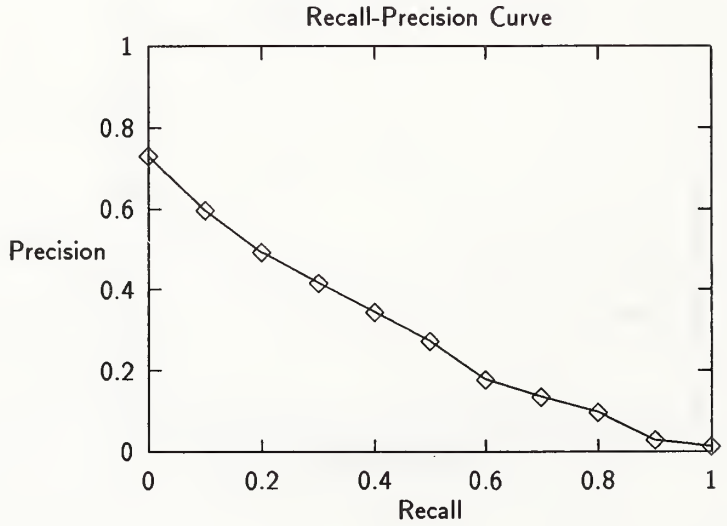
Summary Statistics	
Run Number	TOPIC2–full, manual
Num of Queries	50
Total number of documents over all queries	
Retrieved:	40869
Relevant:	10489
Rel_ret:	5600

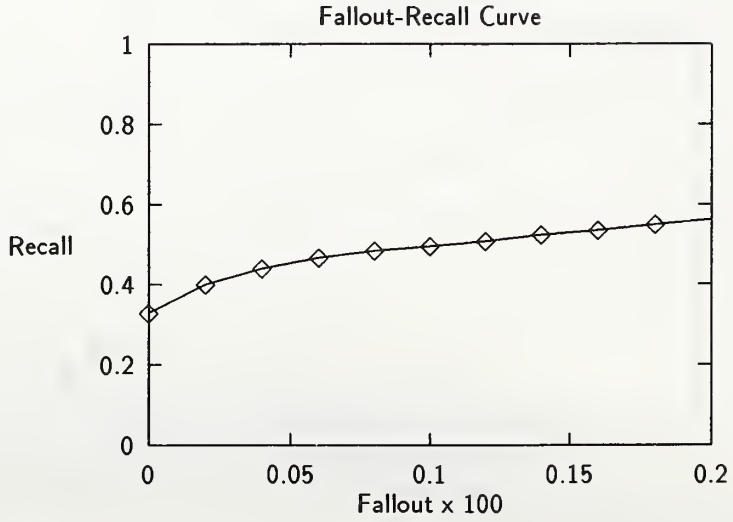
Recall Level Averages	
Recall	Precision
0.00	0.7300
0.10	0.5963
0.20	0.4923
0.30	0.4160
0.40	0.3450
0.50	0.2739
0.60	0.1783
0.70	0.1350
0.80	0.0963
0.90	0.0289
1.00	0.0120
Average precision over all relevant docs	
non-interpolated	0.2830

Document Level Averages	
	Precision
At 5 docs	0.6080
At 10 docs	0.5800
At 15 docs	0.5573
At 20 docs	0.5390
At 30 docs	0.5227
At 100 docs	0.4028
At 200 docs	0.3048
At 500 docs	0.1805
At 1000 docs	0.1120
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3418

Fallout Level Averages	
Fallout	Recall
0.00000	0.3293
0.00020	0.4000
0.00040	0.4399
0.00060	0.4677
0.00080	0.4843
0.00100	0.4966
0.00120	0.5094
0.00140	0.5243
0.00160	0.5355
0.00180	0.5500
0.00200	0.5631







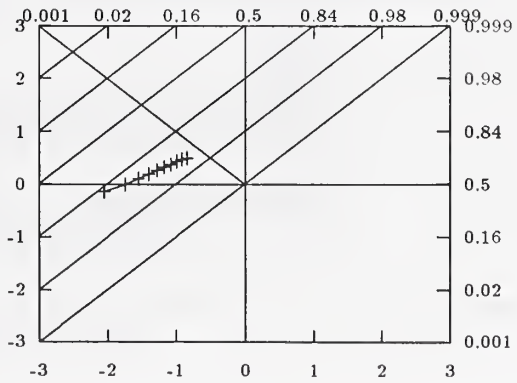
Summary Statistics	
Run Number	siems1–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10489
Rel_ret:	6666

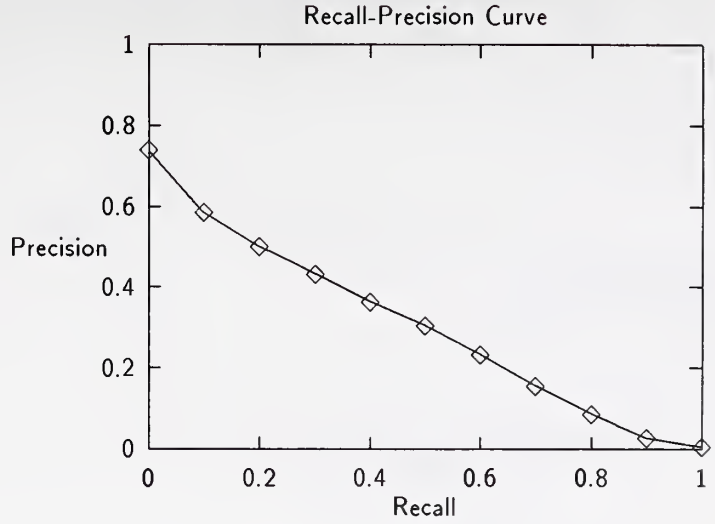
Recall Level Averages	
Recall	Precision
0.00	0.7397
0.10	0.5851
0.20	0.5002
0.30	0.4330
0.40	0.3637
0.50	0.3050
0.60	0.2333
0.70	0.1555
0.80	0.0860
0.90	0.0274
1.00	0.0042
Average precision over all relevant docs	
non-interpolated	0.2984

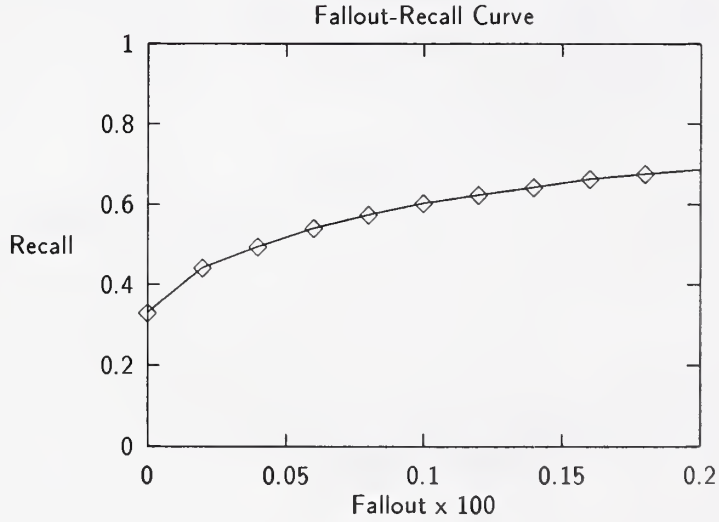
Document Level Averages	
	Precision
At 5 docs	0.6080
At 10 docs	0.5960
At 15 docs	0.5693
At 20 docs	0.5450
At 30 docs	0.5260
At 100 docs	0.4070
At 200 docs	0.3229
At 500 docs	0.2092
At 1000 docs	0.1333
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3482

Fallout Level Averages	
Fallout	Recall
0.00000	0.3305
0.00020	0.4414
0.00040	0.4944
0.00060	0.5411
0.00080	0.5737
0.00100	0.6022
0.00120	0.6232
0.00140	0.6427
0.00160	0.6628
0.00180	0.6759
0.00200	0.6865

NORMAL DEVIATE - Fallout-Recall







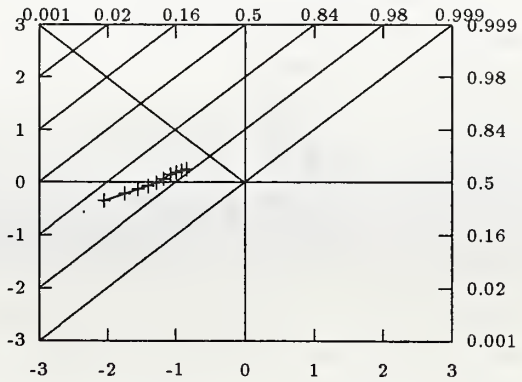
Summary Statistics	
Run Number	pircs1–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10489
Rel_ret:	5913

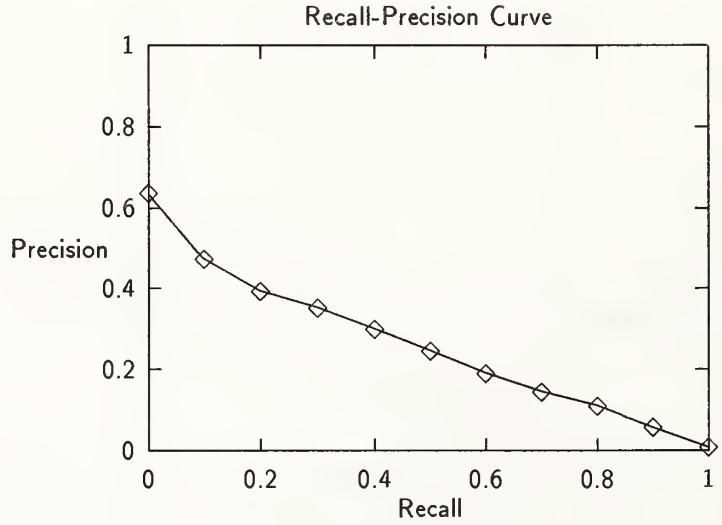
Recall Level Averages	
Recall	Precision
0.00	0.6360
0.10	0.4732
0.20	0.3947
0.30	0.3523
0.40	0.2999
0.50	0.2457
0.60	0.1911
0.70	0.1443
0.80	0.1094
0.90	0.0571
1.00	0.0087
Average precision over all relevant docs	
non-interpolated	0.2488

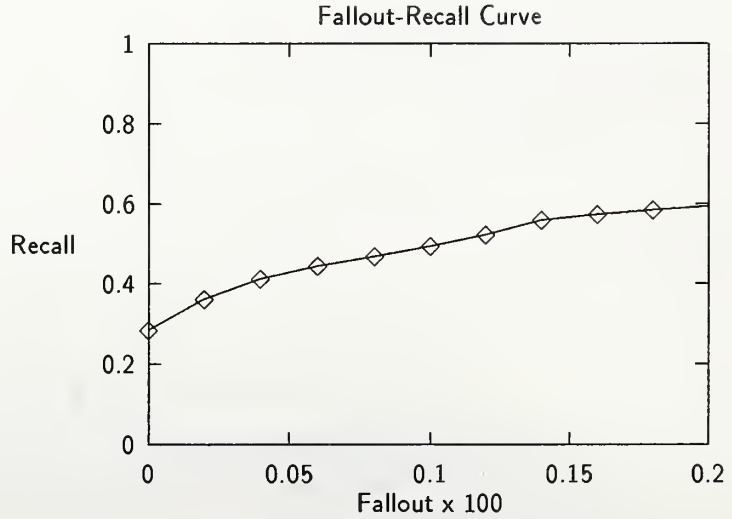
Document Level Averages	
	Precision
At 5 docs	0.4920
At 10 docs	0.4840
At 15 docs	0.4547
At 20 docs	0.4420
At 30 docs	0.4147
At 100 docs	0.3382
At 200 docs	0.2758
At 500 docs	0.1828
At 1000 docs	0.1183
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2950

Fallout Level Averages	
Fallout	Recall
0.00000	0.2840
0.00020	0.3622
0.00040	0.4133
0.00060	0.4448
0.00080	0.4690
0.00100	0.4951
0.00120	0.5229
0.00140	0.5603
0.00160	0.5746
0.00180	0.5853
0.00200	0.5942

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	pircs2–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10489
Rel_ret:	6135

Recall Level Averages	
Recall	Precision
0.00	0.6609
0.10	0.4959
0.20	0.4145
0.30	0.3735
0.40	0.3219
0.50	0.2638
0.60	0.2046
0.70	0.1586
0.80	0.1199
0.90	0.0656
1.00	0.0085

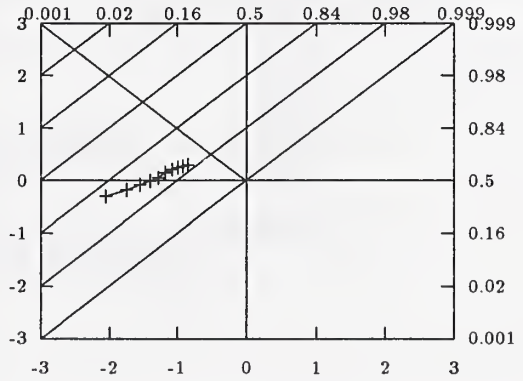
Average precision over all relevant docs	
non-interpolated	0.2664

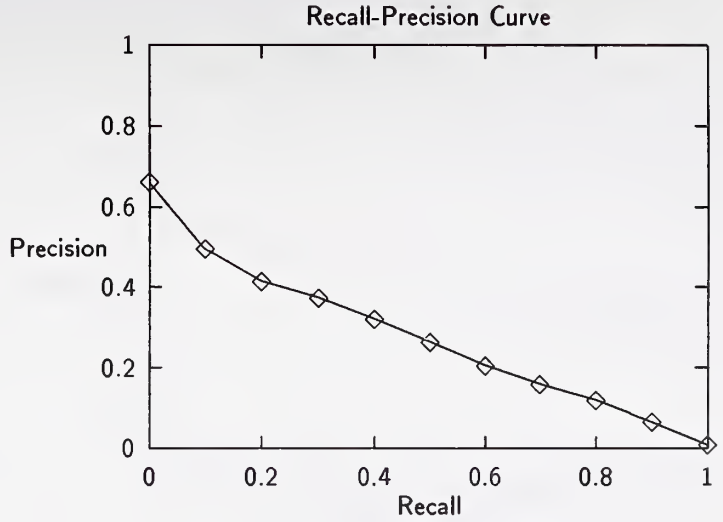
Document Level Averages	
	Precision
At 5 docs	0.5440
At 10 docs	0.5120
At 15 docs	0.4907
At 20 docs	0.4690
At 30 docs	0.4460
At 100 docs	0.3588
At 200 docs	0.2895
At 500 docs	0.1901
At 1000 docs	0.1227

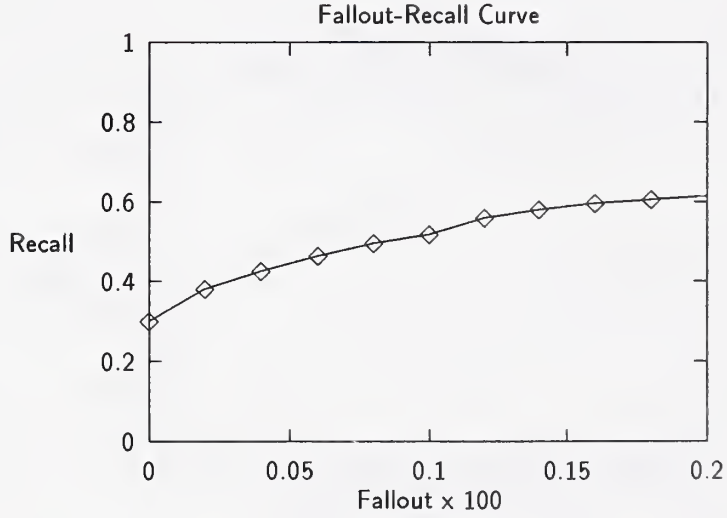
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3155

Fallout Level Averages	
Fallout	Recall
0.00000	0.3010
0.00020	0.3821
0.00040	0.4263
0.00060	0.4648
0.00080	0.4954
0.00100	0.5175
0.00120	0.5603
0.00140	0.5795
0.00160	0.5958
0.00180	0.6056
0.00200	0.6155

NORMAL DEVIATE - Fallout-Recall







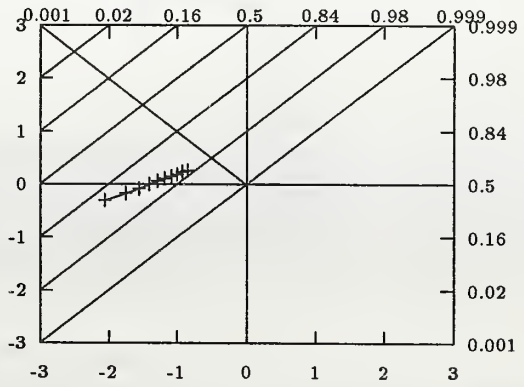
Summary Statistics	
Run Number	VTcms2–full, manual
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10489
Rel_ret:	5898

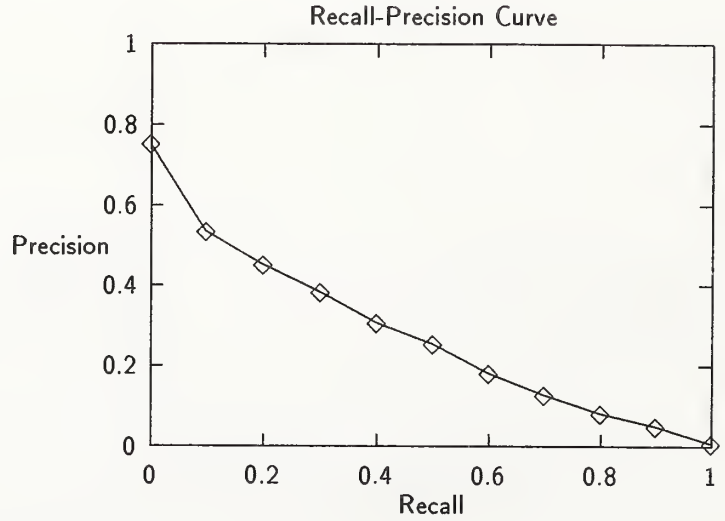
Recall Level Averages	
Recall	Precision
0.00	0.7524
0.10	0.5342
0.20	0.4510
0.30	0.3828
0.40	0.3063
0.50	0.2539
0.60	0.1817
0.70	0.1270
0.80	0.0809
0.90	0.0496
1.00	0.0060
Average precision over all relevant docs	
non-interpolated	0.2681

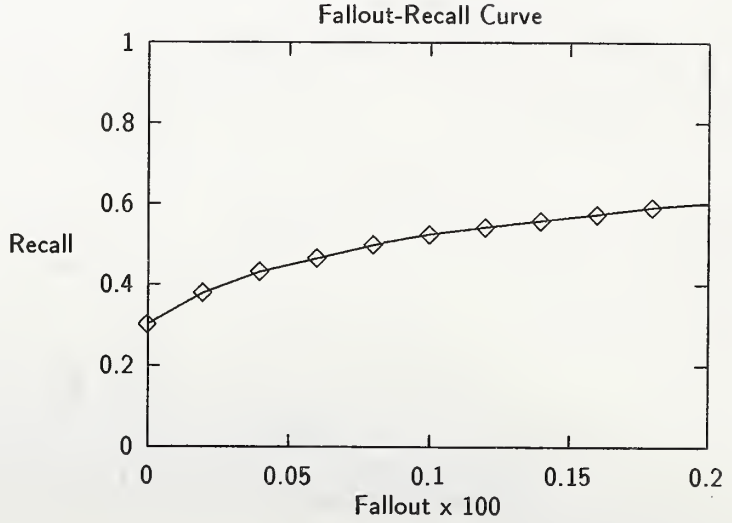
Document Level Averages	
	Precision
At 5 docs	0.6080
At 10 docs	0.5760
At 15 docs	0.5387
At 20 docs	0.5160
At 30 docs	0.4913
At 100 docs	0.3806
At 200 docs	0.2953
At 500 docs	0.1882
At 1000 docs	0.1180
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3143

Fallout Level Averages	
Fallout	Recall
0.00000	0.3015
0.00020	0.3795
0.00040	0.4320
0.00060	0.4660
0.00080	0.4986
0.00100	0.5250
0.00120	0.5421
0.00140	0.5577
0.00160	0.5745
0.00180	0.5916
0.00200	0.6021

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	ADS1–ap only, automatic
Num of Queries	50

Total number of documents over all queries	
Retrieved:	40423
Relevant:	5677
Rel_ret:	822

Recall Level Averages	
Recall	Precision
0.00	0.1620
0.10	0.0542
0.20	0.0468
0.30	0.0273
0.40	0.0201
0.50	0.0056
0.60	0.0056
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000

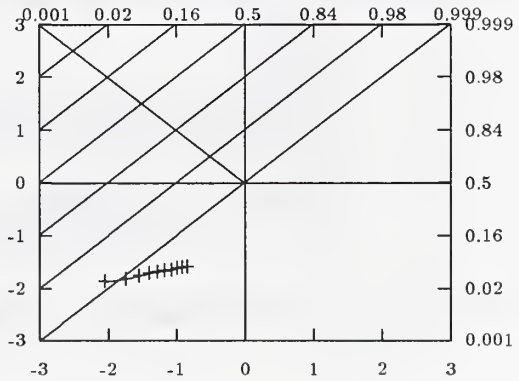
Average precision over all relevant docs	
non-interpolated	0.0195

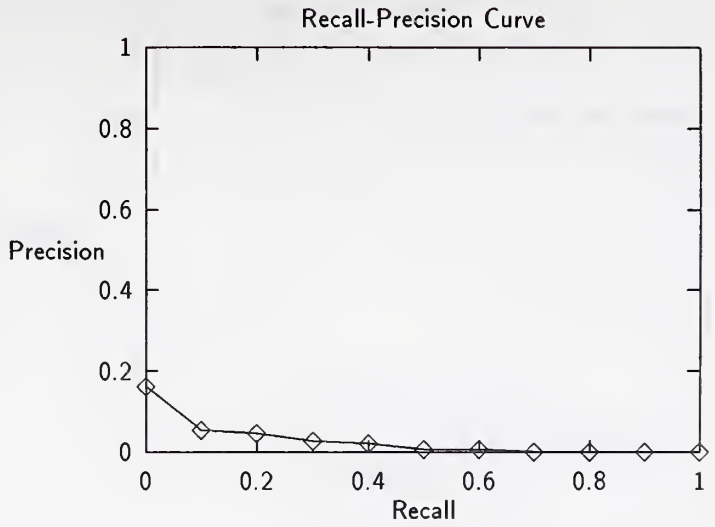
Document Level Averages	
	Precision
At 5 docs	0.0560
At 10 docs	0.0480
At 15 docs	0.0573
At 20 docs	0.0640
At 30 docs	0.0687
At 100 docs	0.0468
At 200 docs	0.0336
At 500 docs	0.0239
At 1000 docs	0.0164

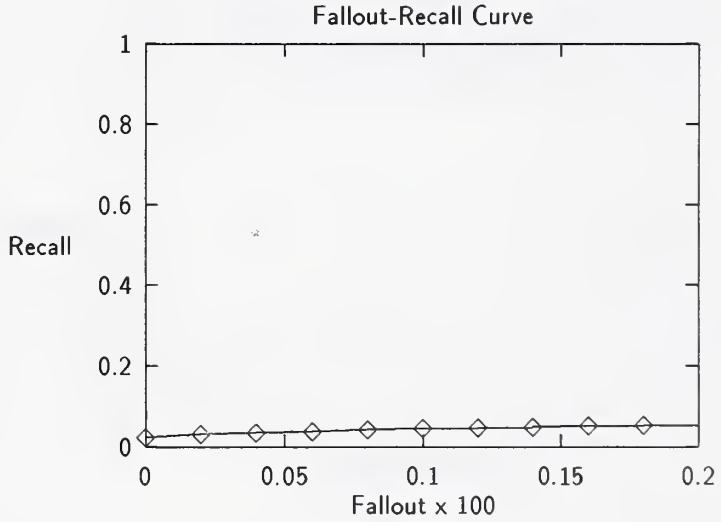
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0390

Fallout Level Averages	
Fallout	Recall
0.00000	0.0236
0.00020	0.0311
0.00040	0.0346
0.00060	0.0389
0.00080	0.0429
0.00100	0.0457
0.00120	0.0476
0.00140	0.0497
0.00160	0.0526
0.00180	0.0539
0.00200	0.0553

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	ADS2–ap only, automatic
Num of Queries	50

Total number of documents over all queries	
Retrieved:	33034
Relevant:	5677
Rel_ret:	1468

Recall Level Averages	
Recall	Precision
0.00	0.2799
0.10	0.1572
0.20	0.1304
0.30	0.1160
0.40	0.0885
0.50	0.0884
0.60	0.0714
0.70	0.0612
0.80	0.0362
0.90	0.0201
1.00	0.0196

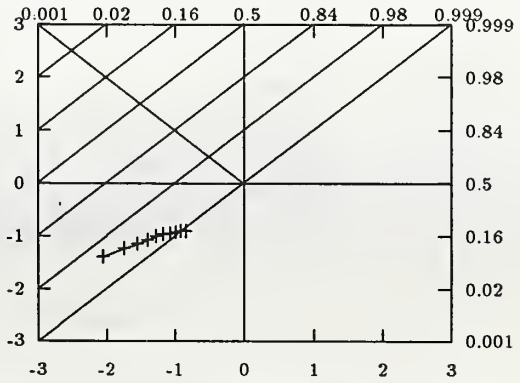
Average precision over all relevant docs	
non-interpolated	0.0821

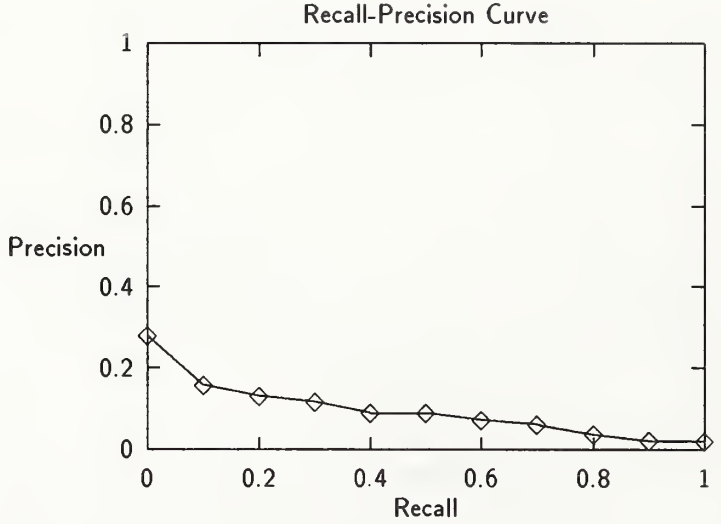
Document Level Averages	
	Precision
At 5 docs	0.1280
At 10 docs	0.1260
At 15 docs	0.1320
At 20 docs	0.1300
At 30 docs	0.1300
At 100 docs	0.1118
At 200 docs	0.0838
At 500 docs	0.0530
At 1000 docs	0.0294

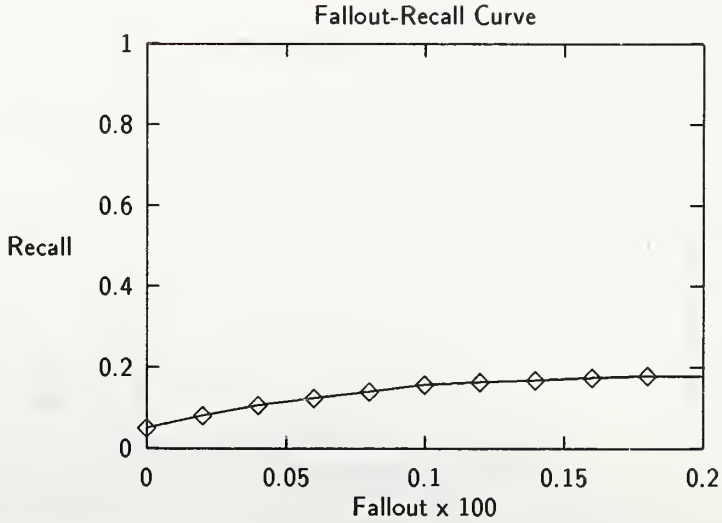
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1092

Fallout Level Averages	
Fallout	Recall
0.00000	0.0506
0.00020	0.0817
0.00040	0.1070
0.00060	0.1248
0.00080	0.1403
0.00100	0.1568
0.00120	0.1643
0.00140	0.1687
0.00160	0.1742
0.00180	0.1801
0.00200	0.1813

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	DalTx2–category B, manual
Num of Queries	50

Total number of documents over all queries	
Retrieved:	11987
Relevant:	2064
Rel_ret:	887

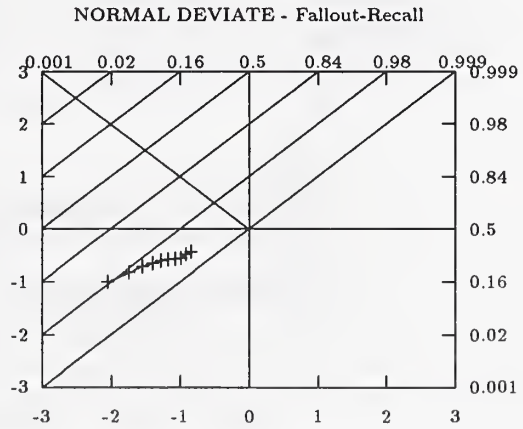
Recall Level Averages	
Recall	Precision
0.00	0.2984
0.10	0.2029
0.20	0.1661
0.30	0.1475
0.40	0.1056
0.50	0.0774
0.60	0.0691
0.70	0.0488
0.80	0.0117
0.90	0.0002
1.00	0.0002

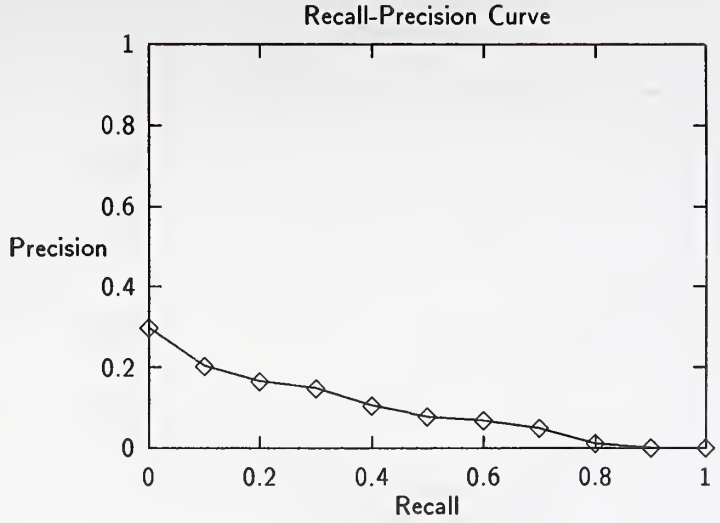
Average precision over all relevant docs	
non-interpolated	0.0893

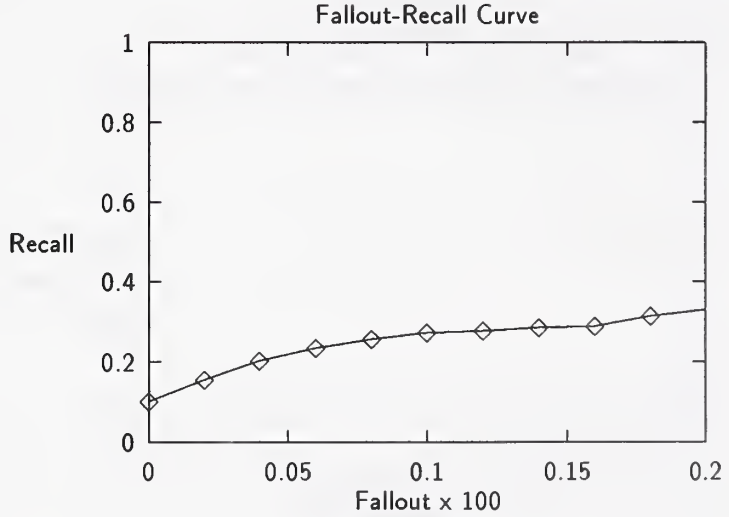
Document Level Averages	
	Precision
At 5 docs	0.1320
At 10 docs	0.1280
At 15 docs	0.1307
At 20 docs	0.1310
At 30 docs	0.1273
At 100 docs	0.1040
At 200 docs	0.0645
At 500 docs	0.0334
At 1000 docs	0.0174

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1199

Fallout Level Averages	
Fallout	Recall
0.00000	0.1023
0.00020	0.1559
0.00040	0.2034
0.00060	0.2353
0.00080	0.2576
0.00100	0.2735
0.00120	0.2781
0.00140	0.2854
0.00160	0.2890
0.00180	0.3151
0.00200	0.3305







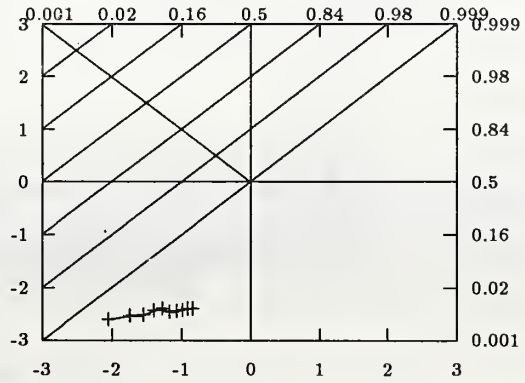
Summary Statistics	
Run Number	SEC–category B, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	19602
Relevant:	2064
Rel_ret:	37

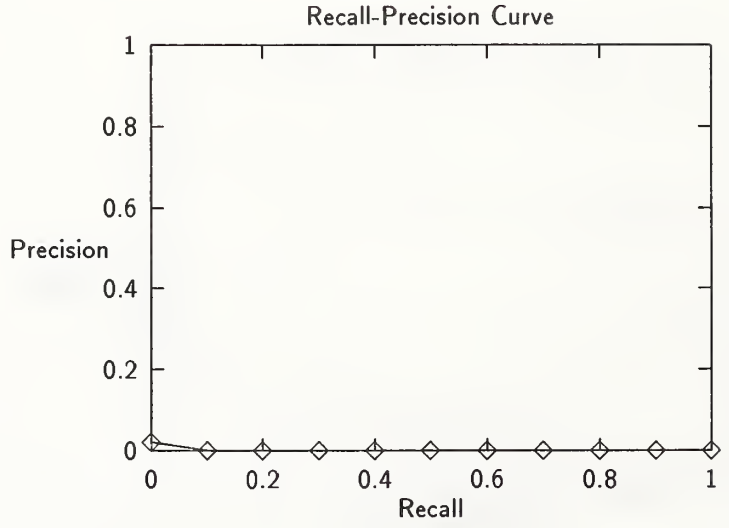
Recall Level Averages	
Recall	Precision
0.00	0.0213
0.10	0.0000
0.20	0.0000
0.30	0.0000
0.40	0.0000
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0005

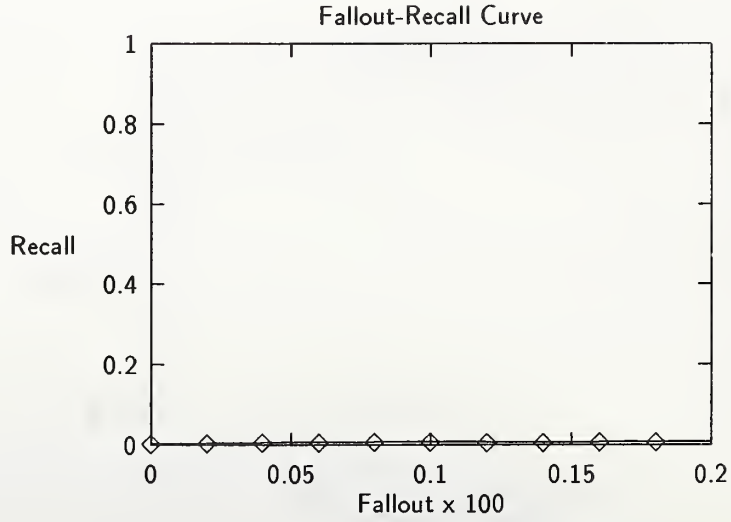
Document Level Averages	
	Precision
At 5 docs	0.0040
At 10 docs	0.0020
At 15 docs	0.0027
At 20 docs	0.0050
At 30 docs	0.0060
At 100 docs	0.0038
At 200 docs	0.0025
At 500 docs	0.0015
At 1000 docs	0.0007
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0042

Fallout Level Averages	
Fallout	Recall
0.00000	0.0030
0.00020	0.0046
0.00040	0.0056
0.00060	0.0057
0.00080	0.0073
0.00100	0.0080
0.00120	0.0069
0.00140	0.0071
0.00160	0.0076
0.00180	0.0080
0.00200	0.0082

NORMAL DEVIATE - Fallout-Recall







Summary Statistics	
Run Number	UCFIS1–category B, automatic
Num of Queries	50

Total number of documents over all queries	
Retrieved:	50000
Relevant:	2064
Rel_ret:	102

Recall Level Averages	
Recall	Precision
0.00	0.0891
0.10	0.0286
0.20	0.0062
0.30	0.0015
0.40	0.0015
0.50	0.0011
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000

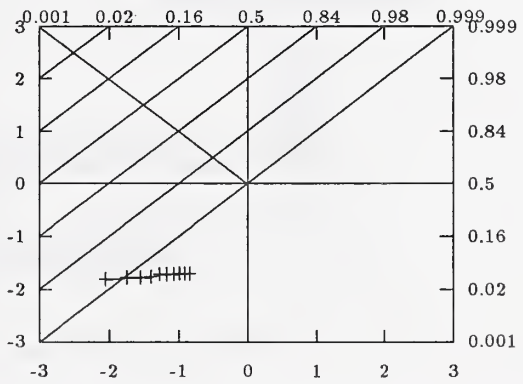
Average precision over all relevant docs	
non-interpolated	0.0060

Document Level Averages	
	Precision
At 5 docs	0.0400
At 10 docs	0.0260
At 15 docs	0.0187
At 20 docs	0.0170
At 30 docs	0.0127
At 100 docs	0.0070
At 200 docs	0.0039
At 500 docs	0.0024
At 1000 docs	0.0020

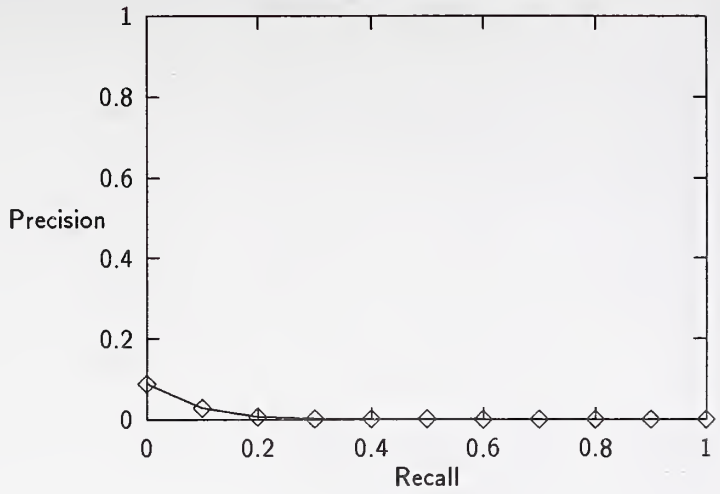
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0128

Fallout Level Averages	
Fallout	Recall
0.00000	0.0307
0.00020	0.0339
0.00040	0.0369
0.00060	0.0374
0.00080	0.0384
0.00100	0.0427
0.00120	0.0427
0.00140	0.0427
0.00160	0.0436
0.00180	0.0436
0.00200	0.0439

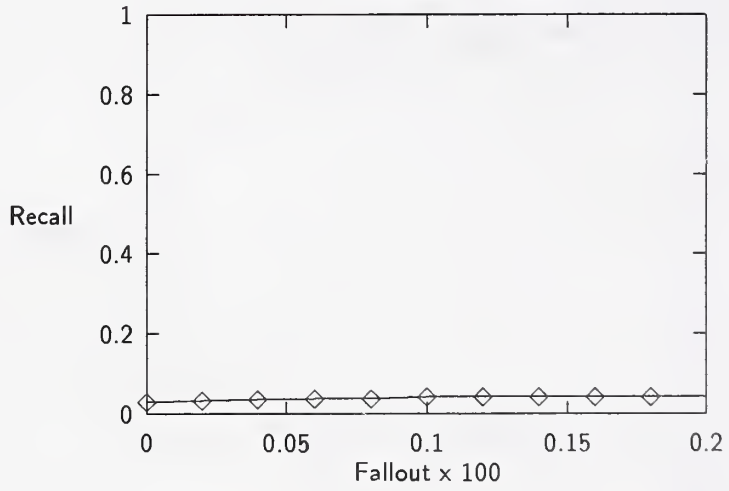
NORMAL DEVIATE - Fallout-Recall



Recall-Precision Curve



Fallout-Recall Curve



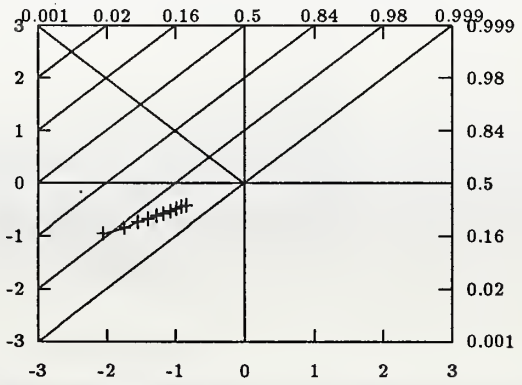
Summary Statistics	
Run Number	uicr1–full, automatic
Num of Queries	50
Total number of documents over all queries	
Retrieved:	50000
Relevant:	10489
Rel_ret:	3067

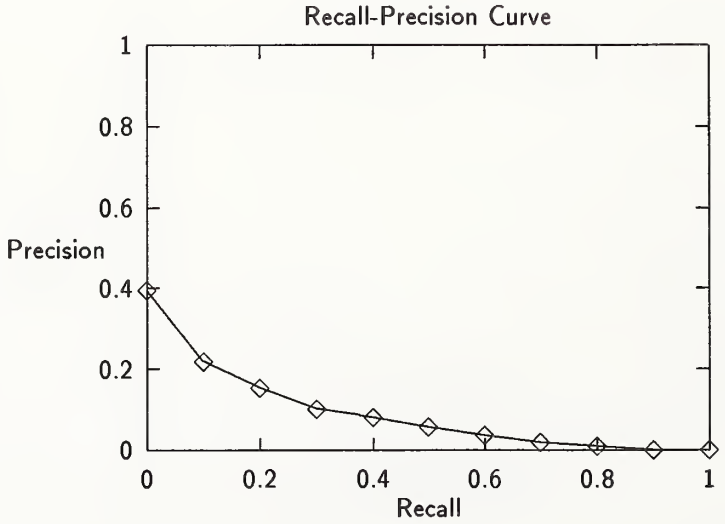
Recall Level Averages	
Recall	Precision
0.00	0.3965
0.10	0.2189
0.20	0.1543
0.30	0.1025
0.40	0.0807
0.50	0.0578
0.60	0.0361
0.70	0.0195
0.80	0.0088
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0780

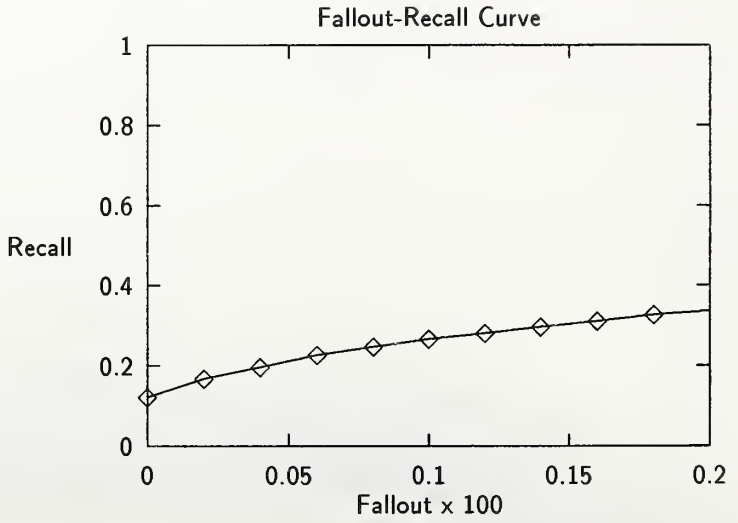
Document Level Averages	
	Precision
At 5 docs	0.1960
At 10 docs	0.2020
At 15 docs	0.2133
At 20 docs	0.2090
At 30 docs	0.2053
At 100 docs	0.1652
At 200 docs	0.1311
At 500 docs	0.0876
At 1000 docs	0.0613
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1440

Fallout Level Averages	
Fallout	Recall
0.00000	0.1231
0.00020	0.1693
0.00040	0.1977
0.00060	0.2278
0.00080	0.2474
0.00100	0.2664
0.00120	0.2814
0.00140	0.2978
0.00160	0.3115
0.00180	0.3272
0.00200	0.3373

NORMAL DEVIATE - Fallout-Recall







APPENDIX B

This appendix contains charts created from the supplemental forms filled out by each group about their system. These charts are meant to supplement the papers and contain a standarded and formatted description of system features and timing aspects.

1A. CONSTRUCTION OF INDICES, KNOWLEDGE BASES, AND OTHER DATA STRUCTURES -- METHODS USED

SYSTEM NAME	Dortmund	Cornell	Berkeley	Rutgers [6]	Siemens	UMASS	VPI
Stopword List	Yes, SMART	Yes, SMART	Yes, Augmented SMART		Yes, SMART	Yes	Yes
- Number of Words	571	571	~600		571	424	418
Controlled Vocabulary	No	No	No			Yes, 4 terms, assigned automatically	No
Stemming	Yes	Yes	Yes		Yes	Yes	No
- Standard Algorithms	SMART	SMART [3]	SMART ver. 10		SMART	Porter	
- Morphological analysis			None			None	
Term weighting	Yes [1]	Yes [4]	Yes [5]		Yes [7]	None during indexing	Yes [9]
Phrase Discovery	Yes	Yes	None			None during indexing	No
- Kind of phrase							
- Statistical Methods	[2]	[2]					
- Syntactic Methods							
Syntactic parsing			None			None during indexing	No
Word Sense Disambiguation			None		Yes [8]	None during indexing	No
Heuristic Associations			None		Yes	None during indexing	No
- Short Definition					Wordnet Links		
Spell Checking (With Manual Correction)			None			None during indexing	No
Spelling Correction			None			None during indexing	No
Proper Noun Identification	Not used	Not used	None			None during indexing	No
Tokenizer	Not used	Not used	None			Yes	Yes
- Which Patterns?						Company, city, & country names	As provided by SMART
Use of Manually Indexed Terms			No			No	No
Other Techniques to Build Data Structures						None	Yes [10]

1A. CONSTRUCTION OF INDICES, KNOWLEDGE BASES, AND OTHER DATA STRUCTURES -- METHODS USED

NOTES:

[1] "ltc" document weights for dortVI, "lsp" document weights for dortP1,
"lsp" document weights for dortQ2, dortL2
"lsp" query regression query weights for dortQ2
"nnn" query weights for dortL2

[2] Any pair of adjacent non stopwords that occur 25 times in D1.

[3] Modified Lovin's Algorithm.

[4] "ltc" document weights for adhoc and routing.
"ltc" documents weights for adhoc.
Rocchio based feedback query weights for routing

[5] Weights determined from various frequency statistics by logistic regression.

[6] Because we used the UMASS INQUERY system and its indexing, all of the answers to the questions in this section for our systems are identical to those for the UMASS system.

[7] Document vectors use "lnc" weights suggested by Buckley et al. in TREC-1 Proceedings: weight of a term is proportional to its frequency in that document and normalized using cosine factor. Query vectors use "ltn" weights: term frequency factor multiplied by inverse document frequency factor. Original query terms normalized by standard cosine normalization; additional terms normalized by the length of the original terms.

[8] Done by hand when selecting synsets to add to topic text.

[9] Wgt = 0.5 + 0.5 * tf/max_tf(doc). SMART "ann" weighting.

[10] Source text preparsed into SMART format before being processed by SMART according to above parameters.

IA. CONSTRUCTION OF INDICES, KNOWLEDGE BASES, AND OTHER DATA STRUCTURES -- METHODS USED

SYSTEM NAME	GE [1]	CLARIT	LSI	HNC	NYU	SYRACUSE	QUEENS
Stopword List	Yes	No	Yes, SMART	Yes	Yes	Yes (Stage 2)	Yes
- Number of Words	5 [2]		570	375	380	48 (Stage 2)	630
Controlled Vocabulary	No	No		No	No	No	No
Stemming	None		Yes	Yes	Yes	Yes	Yes
- Standard Algorithms			SMART	Lovins	No	Kelly & Stone [13]	Porter
- Morphological analysis		Yes [3]		No	Yes		No
Term weighting	Document Ranking Only	Yes [4]	"ltc" [10]	tf, idf	Yes	Yes [14]	Yes
Phrase Discovery	No	Yes		Yes	Yes		Yes
- Kind of phrase		simplex noun phrases [5]			NP's, VP's, Others	Proper nouns & complex nominals	2-word
- Statistical Methods		No		Yes	Partially	No	Yes
- Syntactic Methods		Yes [6]			Yes	Yes	
Syntactic parsing	No	Yes, see above		No	Yes	Yes (partially) [15]	No
Word Sense Disambiguation	No	Yes [7]		No	No	Yes [16]	No
Heuristic Associations	No	No		No	Yes	Yes	No
- Short Definition					Synonymy, specializations	Semantic relations between concepts	
Spell Checking (With Manual Correction)	No	No		No	No	No	No
Spelling Correction	No	No		No	No	No	No
Proper Noun Identification	No	Yes [8]		No	Partial	Yes	No
Tokenizer	No	No		No	Yes	Yes	No
- Which Patterns					year numbers	Dates	
Use of Manually Indexed Terms	No	No		N/A	No	No	No
Other Techniques to Build Data Structures		Yes [9]	Yes [11]			[17]	[12]

IA. CONSTRUCTION OF INDICES, KNOWLEDGE BASES, AND OTHER DATA STRUCURES - METHODS USED

NOTES:

[1] No Pre-indexing

[2] Automatic query generation only

[3] A comprehensive inflectional morphology is used to produce word roots. Participles are retained in surface forms (although normalization is possible). No derivational morphology is used.

[4] 1) IDF/TF over phrases for retrieval.
2) A combination of statistics, including frequency and distribution, for thesaurus discovery.

[5] Simplex noun phrases -- not including prepositional phrases or relative clauses.

[6] A deterministic, rule-based parser that nominates noun phrases based on testing for phrase-boundary conditions.

[7] The parser grammer includes heuristics for syntactic category disambiguation.

[8] Words not identified in the lexicon (about 100,000 root forms of English) are assumed to be "candidate" proper nouns. This technique does not appeal to capitalization information, etc.

[9] 1) Thesaurus Discovery -- which we use for query vector augmentation -- involves the identification of core characteristic terminology over a document set. It ranks all terms according to several parameters, including frequency and distribution, and then selects the subset of terminology that optimizes for these scores.
2) Documents are broken into smaller, paragraph size units called "subdocuments." The subdocuments are the units from which statistics are drawn and over which similarity is measured.

[10] Yes, "ltc"; log (term freq in doc) * term-idf weight * document length normalization.

[11] The LSI/SVD analysis of the term by document matrix. LSI takes a term-document matrix, transforms it by a user-specified weighting scheme (SMARTS's "ltc" for TREC-2 experiments), and then calculates the best k-dimensional approximation to this matrix using singular value decomposition (SVD). The number of dimensions, k, was about 200 for TREC-2. All retrieval is done in this 200-dimensional LSI space rather than using raw term overlap.

[12] A table of 528 manual and 13787 automatic 2-word phrases. When these are identified in adjacent posistions in documents or queries, they are used as additional index terms.

[13] Kelly & Stone's Algorithm with minor modifications, using LDOCE as the accompanying lexicon for table look-ups.

[14] Stage 1: based on the text structure (discourse structure) of texts.
Stage 2: implicitly with the conceptual graph matching/scoring scheme.

[15] Part-of-speech tagging, phrase and clause bracketing, and special handlers in the RCD module.

[16] Stage 1: for all words in texts.
Stage 2: only when RIT codes are assigned.

[17] 1) assignment of subject field codes to individual words
2) SFC vector construction for documents
3) proper noun knowledge base construction
4) inverted index construction
5) assignment of RIT codes
6) conversion of text into conceptual graph representation

IA. CONSTRUCTION OF INDICES, KNOWLEDGE BASES, AND OTHER DATA STRUCTURES -- METHODS USED

SYSTEM NAME	CITY	ERIM	TMC	CITRI	UCLA	SEC	ETH	TRW [9]
Stopword List	Yes	No	Yes	No	Yes	No	Yes	
- Number of Words	658 stopwords 284 semi-stopwords [1]		370		438 stop words 58 semi-stopwords [5]		250	
Controlled Vocabulary	No	No	No	No	No	No	No	
Stemming	Yes	No		Yes		Not currently	Yes	
- Standard Algorithms	Based on Porter [2]			Lovin's	Based on Porter [2]		Suffix stripping (Porter, 1980)	
- Morphological analysis						Not yet	No	
Term weighting	No	No	Yes	IDF	No	Yes	Yes [7]	
Phrase Discovery	No, a few phrases are recognized	No	Yes	No	Yes [6]	Yes	No	
- Kind of phrase			Adjacent words			Noun		
- Statistical Methods						Statistical tagging		
- Syntactic Methods			Yes					
Syntactic parsing	No	No		No	No	No	No	
Word Sense Disambiguation	No	No		No	No	None	No	
Heuristic Associations	No	No		No	No		No	
- Short Definition								
Spell Checking (With Manual Correction)	No	No		No	No	No	No	
Spelling Correction	No	No		No	No	No	No	
Proper Noun Identification	No	No		No	No	Yes	No	
Tokenizer	Not used [3]	No		No	Not used [3]	No	No	
- Which Patterns?	Date ranges				Date ranges			
Use of Manually Indexed Terms	No	No		No	No	No	No	
Other Techniques to Build Data Structures		No		[4]			Yes [8]	

IA. CONSTRUCTION OF INDICES, KNOWLEDGE BASES, AND OTHER DATA STRUCTURES -- METHODS USED

NOTES:

[1] Two stoplists were used.

System 1: For routing and manual/feedback 411 stopwords + 58 semi-stopwords.

System 2: For the automatic adhoc run 247 stopwords + 226 semi-stopwords

Semi-stopwords are not used in query expansion unless they also appear in the query.

[2] Based on Porter with enhancements to deal with peculiar plurals and partial conflation of British/American spellings.

[3] Date ranges are recognized, but no use was made of this.

[4] Fast inversion method operating in limited main memory (peak requirement 40 MB) and limited temporary disk (peak requirement 50 MB more than the final inverted file). Both text and index stored compressed. All alphanumeric strings indexed.

[5] Semi-stopwords are not used in query expansion unless they also appear in the query.

[6] Some phrases are recognized by default okapi go-list. New phrases "discovered" from concepts section of topics 1-100 by treating comma-separated text as a phrase.

[7] nff * nidf; totally n documents, feature f, document d; feature frequency: ff(f,d)
normalized feature frequency: nff (f,d) = ff(f,d)/max {ff (fi,d) |fi element d}
document frequency df(f)=|{dj|f element dj}|
inverse document frequency idf(f)=log ((n+1)/(df(f)+1))
normalized inverse document frequency
nidf(f)=idf(f)/log(n+1)=1-log (df(f)+1)/log (n+1)
nidf's are pre-computed from disk 1 & 2 only
new features from disk 3 have an nidf of 1.0

[8] Mapping each term t to a 32 bit integer by applying two hash functions to the term and by hashing the two resulting numbers into one number.
--> max 3 terms mapped to same integer (only in two cases).
--> only 426 terms (0.1% of all terms) are mapped to ambiguous numbers.

[9] We used no special index data structures for TRW1 (proximity queries).

IA. CONSTRUCTION OF INDICES, KNOWLEDGE BASES, AND OTHER DATA STRUCTURES -- METHODS USED

SYSTEM NAME	ADS	UIC [2]	Dalhousie	MEAD	UIF	IDSR	
Stopword List	Yes	Yes	No		Yes	Yes	
- Number of Words	421	632	None		166 [6]	399	
Controlled Vocabulary	No	No	No		No	No	
Stemming	Yes (ads1)	Yes	No		Yes, but not used	Yes	
- Standard Algorithms	Paice, 1990	Extracted from SMART, coded in SPITBOL	None		Lovins, modified	Paice	
- Morphological Analysis	No	No	No		None	No	
Term weighting	No	IDF [3]	No	tf * idf	Yes	Yes	
Phrase Discovery	No	Yes	No		No	No	
- Kind of phrase		[4]	None				
- Statistical Methods		Statistics on word pairs computed & used	None				
- Syntactic Methods		No	None				
Syntactic parsing	No	No	None		No	No	
Word Sense Disambiguation	No	No	No		Yes [7]	No	
Heuristic Associations	No		No		No	No	
- Short Definition		word co-occurrences	None				
Spell Checking (With Manual Correction)	No	No	No		No	No	
Spelling Correction	No	No	No		No	No	
Proper Noun Identification	No	No	No		No	No	
Tokenizer	No	No	No		No	No	
- Which Patterns							
Use of Manually Indexed Terms	No	No	No		No	No	
Other Techniques to Build Data Structures	Yes [1]		List of offsets [5]		None		

IA. CONSTRUCTION OF INDICES, KNOWLEDGE BASES, AND OTHER DATA STRUCTURES -- METHODS USED

NOTES:

[1] Binary classification trees built automatically from the original documents and the topic statements.

[2] Searching is done on the fly, as raw text is processed. The intermediate data structure is discarded as the search is completed. What is saved, however, is a record of all words appearing within three word positions of each query word.

[3] Inverse Document Frequency with a base of only those documents containing at least one of the query words.

[4] All word co-occurrences within 3 word positions of a query word are listed as word pairs.

[5] A list of offsets to the beginning of records (articles) is generated at the beginning of the session for each data file.

[6] 166 stop words, 122 abbreviations, 47 hyphenated words, 24 entries for abbreviations and alternate notation for months, 35 entries for legitimate words not to be prefixed, and 6 entries for legitimate prefixes.

[7] The semantic lexicon we used is based on word senses found in Roget's Thesaurus.

IB. CONSTRUCTION OF INDICES, KNOWLEDGE BASES, AND OTHER DATA STRUCTURES -- STATISTICS ON DATA STRUCTURES

SYSTEM NAME	DORTMUND	CORNELL	BERKELEY	RUTGERS [8]	SIEMENS	UMASS	VPI
Inverted Index	Yes	Yes	Yes		Yes	Yes	N/A
- Total Storage (MB)	~ 460 (D3) 863 (D1/D2) [2]	390 (D3) 863 (D1/D2) [4]	1GB [5]		667 (D1/D2) 337 (D3)	882 (D1/D2) 415)D3)	
- Total Computer Build Time (Hours)	4.8 (D3) 9 (D1/D2) [2]	4.8 (D3) 9 (D1/D2) [2]	6-20 [6]		8 (D1/D2) 2 (D3) [9]	40	
- Automatic Process	Yes	Yes	Yes		Yes	Yes	
- Number Manual Hours							
- Term Positions Stored	No	No	Yes [7]		No	Yes	
- Single Terms Only	No [3]	No [3]	Yes		Yes	Yes	
Clusters			None			Not used	N/A
- Total Storage (MB)							
- Total Computer Build Time (Hours)							
- Description of Method							
- Automatic Process							
- Number of Manual Hours							
N grams, Suffix arrays, Signature Files			None			Not used	N/A
- Total Storage (MB)							
- Total Computer Build Time (Hours)							
- Description of Method							
- Automatic Process							
- Number of Manual Hours							

IB. CONSTRUCTION OF INDICES, KNOWLEDGE BASES AND OTHER DATA STRUCTURES -- STATISTICS ON DATA STRUCTURES

NOTES:

[1] ~ 460 Mbytes for D3 (test set for routing
863 Mbytes for D12 (learning set for routing and test set for adhoc).

[2] 4.8 cpu hours for D3, 9 cpu hours for D12.

[3] Above figures are roughly 25% phrases.

[4] 390 Mbytes for D3 (test set for routing).
863 Mbytes for D12 (learning set for routing and test set for adhoc).

[5] One GB total for all 7 collections, 3 disks.

[6] Ranged from 6 to 20 hours for each collection.

[7] Not used in final runs.

[8] Because we used the UMASS INQUERY system and its indexing, all of the answers to the questions in this section for our systems are identical to those for the UMASS system

[9] Slightly over 2 hours to build inverted index for Disk 3 given structures for Disk 2.

IB. CONSTRUCTION OF INDICES, KNOWLEDGE BASES AND OTHER DATA STRUCTURES -- STATISTICS ON DATA STRUCTURES

SYSTEM NAME	GE [1]	CLARIT	LSI	HNC	NYU	SYRACUSE	QUEENS
Inverted Index		None	[2]		Yes	Yes	Yes [4]
- Total Storage (MB)				1500 MB (D1, D2) 760 MB (D3)	305 MB [3]	1,284 [5]	
- Total Computer Build Time (Hours)				~100	250 (WSI) 100 (SJM)	24 + Hrs. [6]	
- Automatic Process				Yes	Yes	Yes	
- Number Manual Hours							
- Term Positions Stored				No	No	No	No
- Single Terms Only				N/A	No	Yes (Stage 2)	No
Clusters		None		Yes (tested on smaller corpora)	No	No	No
- Total Storage (MB)				~1KB/cluster			
- Total Computer Build Time (Hours)				~48 Hrs. for 75K documents (230 MB)			
- Description of Method				K-MEANS			
- Automatic Process				Yes			
- Number of Manual Hours							
N grams, Suffix arrays, Signature Files		None		No	No	No	No
- Total Storage (MB)							
- Total Computer Build Time (Hours)							
- Description of Method							
- Automatic Process							
- Number of Manual Hours							

IB. CONSTRUCTION OF INDICES, KNOWLEDGE BASES AND OTHER DATA STRUCTURES -- STATISTICS ON DATA STRUCTURES

NOTES:

[1] No Pre-indexing -- most of the below don't apply.

[2] Used SMART's pre-processing to construct a term-document matrix for input to the SVD. This took about 9-10 hours. After this, SMART is used only to access raw text files. We do not store an inverted index, since we use the LSI-space for matching and retrieval.

[3] 204 MB (0.55 GB WSJ text); 101 MB (0.3 GB SJMN text).

[4] System creates a network. Files created are described in B.5 (special routing structures) and B.6 (other data structures).

[5] 1) Proper noun, complex nominal, and text structure index - 1,000 MB for WSJ and SJM.
2) conceptual graph - 284 MB (WSJ).

[6] 1) Proper noun, complex nominal, and text structure index - 24 hours (for both WSJ and SJM).
2) conceptual graph - ?

IB. CONSTRUCTION OF INDICES, KNOWLEDGE BASES, AND OTHER DATA STRUCTURES -- STATISTICS ON DATA STRUCTURES

SYSTEM NAME	CITY	ERIM	TMC	CITRI	UCLA	SEC	ETH	TRW
Inverted Index	Yes	No index, system was a filter	Yes	Yes	Yes		No	
- Total Storage (MB)	System 1: ~1260 System 2: ~1420		512 MB for entire (2.2GB) training collection	130 (document index) 185 (page index)	~750 MB	approximately equal to uncompressed text		
- Total Computer Build Time (Hours)	40 CPU hours		0.3 (20 min.)	4	~12 Hrs. [4]			
- Automatic Process	Yes [1]		Yes	Yes	Yes [1]	Yes		
- Number Manual Hours								
- Term Positions Stored	Yes, (field, sentence, word)		Yes [2]	No [3]	Yes (field, sentence, word)	No		
- Single Terms Only	Some prespecified phrases are treated as words		Yes	Yes	Some pre-specified phrases are treated as words	No		
Clusters	No	None			No	No	No	
- Total Storage (MB)								
- Total Computer Build Time (Hours)								
- Description of Method								
- Automatic Process								
- Number of Manual Hours								
N grams, Suffix arrays, Signature Files	No	None			No	No	No	
- Total Storage (MB)								
- Total Computer Build Time (Hours)								
- Description of Method								
- Automatic Process								
- Number of Manual Hours								

IB. CONSTRUCTION OF INDICES, KNOWLEDGE BASES, AND OTHER DATA STRUCTURES -- STATISTICS ON DATA STRUCTURES

NOTES:

[1] Yes, when enough disk available --> needs > 100% scratch.

[2] (Paragraph, sentence, word position).

[3] Term frequencies within document are stored.

[4] Don't know exact time, estimate 12 hours. Database was indexed by Stephen Walker at City University.

IB. CONSTRUCTION OF INDICES, KNOWLEDGE BASES AND OTHER DATA STRUCTURES -- STATISTICS ON DATA STRUCTURES

SYSTEM NAME	ADS	UIC	Dalhousie	MEAD	UIF	IDSR	
Inverted Index	No		No	Yes	No		
- Total Storage (MB)							
- Total Computer Build Time (Hours)							
- Automatic Process							
- Number Manual Hours							
- Term Positions Stored							
- Single Terms Only							
Clusters	No		No	Yes	No		
- Total Storage (MB)							
- Total Computer Build Time (Hours)							
Description of Method							
- Automatic Process							
- Number of Manual Hours							
N grams, Suffix arrays, Signature Files	No		No		No		
- Total Storage (MB)		~ Equal to size of original text [1]					
- Total Computer Build Time (Hours)		100 hours					
- Description of Method		[2]					
- Automatic Process		Yes					
- Number of Manual Hours							

IB. CONSTRUCTION OF INDICES, KNOWLEDGE BASES AND OTHER DATA STRUCTURES -- STATISTICS ON DATA STRUCTURES

NOTES:

[1] Files of all word pairs for which at least one member is in the query word list are nearly equal in size to the original text.

[2] Find shortest paths in the network of all word pairs including at least one query word. (Full realization of shortest path approach not done for TREC-2; only directly matching pairs used for official results.

IB. CONSTRUCTION OF INDICES, KNOWLEDGE BASES, AND OTHER DATA STRUCTURES -- STATISTICS ON DATA STRUCTURES (CON'T)

SYSTEM NAME	DORTMUND	CORNELL	BERKELEY	RUTGERS [3]	SIEMENS	UMASS	VPI
Knowledge Bases						None	N/A
- Total Storage							
- Number of Concepts represented							
- Type of Representation							
- Total Computer Build Time							
- Total Manual Time							
- Use of Manual Labor (*)							
- Auxiliary Files Needed (+)							
Routing Structures	Yes [1]	Yes [1]	None			None	N/A
Total Storage (MB)	1.5	1.5					
- Total Computer Build Time	1.8	1.8					
- Automatic Process	Yes	Yes					
- Number of Manual Hours							
- Description of Method							
Other Data Structures	Yes		None		Yes [4]	None	Yes [5]
- Total Storage (MB)					45 (D1/D2) 36 (D3)		
- Total Computer Build Time (Hours)					2 (D1/D2) 1.75)D3)		
- Automatic Process					Yes		Yes
- Description of Method	[2]						[6]

(*) (a) Mostly manually built using special interface
(b) Mostly machine built with manual correction
(c) Initial core manually built to "bootstrap" for completely machine-built completion
(d) Other

(+) (a) Machine-readable dictionary
(b) Other

IB. CONSTRUCTION OF INDICES, KNOWLEDGE BASES, AND OTHER DATA STRUCTURES -- STATISTICS ON DATA STRUCTURES (CON'T)

NOTES:

[1] Occurrence statistics for the most frequently occurring (in learning set rel docs) 1000 terms for each routing query.

[2] For the adhoc runs, the 'query regression' method was used. The query regression coefficients were computed from the query.nnn and doc.lsp-file (which was created by polynomial regression). Afterwards reweighting of the q3-query-file. 4 query.nnn -> query.lsp.

[3] Because we used the UMASS INQUERY system and its indexing, all of the answers to the questions in this section for our systems are identical to those for the UMASS system.

[4] Document vector files and term dictionary produced by SMART: Each individual collection was indexed separately, so sizes/times are average per collection, with the range of values specified. The collection statistics are based on the summation of individual collection values so are perhaps less accurate; the collection size of the term dictionary cannot be effectively estimated with this approach. Term positions are not stored within the document vector files.

	Average	Range	Collection
Document Vector Files (MB)	120	31-124	1100
Term Dictionary (MB)	16	15-17	Unknown
Time to create both above files (Hours)	10	6-14	120

[6] Standard process as implemented by SMART, following parameters as in Part I, Section A.

IB. CONSTRUCTION OF INDICES, KNOWLEDGE BASES, AND OTHER STRUCTURES -- STATISTICS ON DATA STRUCTURES (CON'T)

SYSTEM NAME	GE	CLARIT	LSI	HNC	NYU	SYRACUSE	QUEENS
Knowledge Bases		None			Yes	No	No
- Total Storage					1.7		
- Number of Concepts represented					31,320		
- Type of Representation					Associations		
- Total Computer Build Time					20		
- Total Manual Time					0		
- Use of Manual Labor (*)					None		
- Auxiliary Files Needed (+)					(a) OALD		
Routing Structures		None		No		No	See "Other Data Structures" [7]
- Total Storage (MB)							1GB [8]
- Total Computer Build Time							8 x 0.75 Hrs. starting from direct file
- Automatic Process							Yes
- Number of Manual Hours							
- Description of Method	[1]						[9]
Other Data Structures	Expansions from corpus (20,000 terms)	Yes [2]	Yes [3]	Word & document context vectors		Yes [15]	Yes [10]
- Total Storage (MB)	2	1	676 MB [4]	620 MB for words 740 MB for Docs		2,201 MB [16]	[11]
- Total Computer Build Time (Hours)	12	3	42 Hrs. [5]	~72 Hrs. for Disks 1-3		Less than 1 hour [17]	[12]
- Automatic Process	Yes	Yes	Yes	Yes		Yes	No [13]
- Description of Method		See [2]	[6]			[18]	[14]

(+) (a) machine-readable dictionary
(b) other

(*) (a) Mostly manually built using special interface
(b) Mostly machine built with manual correction
(c) Initial core manually built to "bootstrap"
(d) Other

NOTES:

[1] Used TF/IDF to pull out significant terms from documents, found significant term pairs using entropy-based statistic.

[2] A first-order thesaurus (which is a collection of core terminology) is constructed for each topic. In the case of routing topics, the source for the thesaurus for each query is the set of query-specific known relevant documents. In the case of ad-hoc queries, relevant documents are automatically identified via an initial retrieval (querying) pass over the entire collection.

[3] LSI uses "reduced-dimensional" term and document vectors (see below for details of how they are constructed). For TREC-2, we used approximately 200 dimensions. Each term and each document has a real-valued vector in this 200 dimensional space. For the routing queries, we used an 88112 term x 68385 doc sample to construct an LSI-space. Routing vectors are embedded in this space. (At this point, all the documents used in the scaling could be removed. We did not do so.) The resulting data structure was: 127 MB. (Could be reduced to 70 MB if documents omitted.) For adhoc queries, we constructed an 82968 term x 69997 doc LSI-space. We folded in the 672358 CD-12 documents that were not in this sample. The resulting data structure was 549 MB.

[4] 127 MB for routing (could be reduced to 70 MB), 549 MB for Adhoc.

[5] Approximately 20 hours for the routing SVD and 20 hours for the adhoc SVD. In addition, 672K documents were added for the adhoc run, taking about 2 hours. (All times are on a Sparc10 with 128 MB RAM or 384 MB RAM.)

[6] LSI/SVD analysis of document collections.

1. Create a raw term-by document matrix, and transform the cell entries by the appropriate weighting scheme. Used SMART pre-processing for this.
2. Calculate the best reduced k-dimensional approximation to this matrix using singular value decomposition (SVD). For the TREC-2 experiments, about 200 dimensions were used in the approximation - 199 dimensions for adhoc and 204 dimensions for the routing queries. Retrieval uses this 200-dimensional LSI-space.
3. If necessary, fold in any terms or documents that are not in the original SVD analysis. Necessary for adhoc queries, not for routing queries.

[7] Network node, edge files; routing using network node and edge files is straightforward.

[8] Node file: 8x20; Edge File: 8x5 for 1 GB.

[9] Routing:

1. Process 1 GB from Disk 1 (WSJ1, AP1, DOE, FR1, ZIF1).
2. Process queries against Disk 1 (training).
3. Process new Disk 3 as if they were queries -- to make use of Disk 1 statistics.
4. Combine queries, (old) dictionary and Disk 3 into network for retrieval.

[10]

1. Subdocument file
2. Coded file (direct file)
3. DOC ID checking file
4. TERM ID checking file
5. Docnum file
6. Termnum (dictionary) file
7. Node file
8. Edge file

[11] (For 1 GB):

1.	1 GB	4.	6	7.	8 x 20
2.	1.1 GB	5.	20	8.	8 x 5
3.	16	6.	7.5		

[12] (For 1 GB):

1. 3
2 - 6. 40
7, 8. 8 x 0.75 = 6

[13] Yes, if sufficient RAM and disk space. For this experiment, No. Two hours of manual labor.

[14] Raw Text --> Subdocument file
Subdocument --> coded file, DOC ID file, TERM ID file, docnum file, termnum (dictionary) file.
Coded, termnum, docnum --> node, edge files.

[15] Synonymous complex nominal listings;
concept - relation - concept triples

[16] 3 MB (synonymous complex nominal listings;
2,198 MB - WSJ (concept - relation - concept triples)

[17] Less than one hour for building complex nominal lists based on 50 topic statements.

[18]

1) Special purpose grammar which is written to extract complex nominals
2) A set of special handlers process tagged and bracketed text based on the knowledge base to extract concept - relation - concept triples.

IB. CONSTRUCTION OF INDICES, KNOWLEDGE BASES AND OTHER DATA STRUCTURES -- STATISTICS ON DATA STRUCTURES (CON'T)

SYSTEM NAME	CITY	ERIM	TMC	CITRI	UCLA	SEC	ETH	TRW
Knowledge Bases	No	None			No	Lexicons	No	
- Total Storage								
- Number of Concepts represented								
- Type of Representation								
- Total Computer Build Time						?		
- Total Manual Time						None		
- Use of Manual Labor (*)						None		
- Auxiliary Files Needed (+)						None		
Routing Structures	No	None			No	None	No	
Total Storage (MB)								
- Total Computer Build Time								
- Automatic Process								
- Number of Manual Hours								
- Description of Method								
Other Data Structures	No	None		TREC text was compressed	No		Yes [3]	Yes [7]
- Total Storage (MB)				632.3 [1]			[4]	5
- Total Computer Build Time (Hours)				8 [2]			[5]	<24
- Automatic Process				Yes			Yes	Yes
- Description of Method				Huffman coding, word based model			[6]	

(+) (a) machine-readable dictionary
(b) other

(*) (a) Mostly manually built using special interface
(b) Mostly machine built with manual correction
(c) Initial core manually built to "bootstrap
(d) Other

IB. CONSTRUCTION OF INDICES, KNOWLEDGE BASES AND OTHER DATA STRUCTURES -- STATISTICS ON DATA STRUCTURES (CON'T)

NOTES:

[1] 605.1 MB for compressed text
27.2 MB for auxiliary structures
Complete retrieval system, including index, occupies 40% of space required by original unindexed text.

[2] 4 hours for compression, plus the 4 hours for indexing; 8 hours total build time.

[3] Combination of signatures and non-inverted document descriptions.

[4] 1. Experiment: topics 51-100 versus disk 3.

signatures:	169 MB
non-inverted document descriptions:	278 MB
normalized inverse document frequencies:	2.1 MB
document lengths:	0.4 MB
mapping of features to numbers:	4.1 MB

2. Experiment: topics 101-150 versus disks 1 and 2.

signatures:	374 MB
non-inverted document descriptions:	618 MB
normalized inverse document frequencies:	2.1 MB
document lengths:	0.4 MB
mapping of features to numbers:	4.1 MB

[5] Uncompressing and indexing:
ca. 21.5h CPU (all collections of all 3 disks)
loading descriptions into access structure:
~ 10 msec./document

[6] For each feature occurring in a document description, a bit is set in the signature of the document by applying a hash function to the feature number. The signatures are used to determine an approximate RSVO. The documents are ranked according to these RSVO's. Beginning at the top of the ranked list, the exact RSV's are computed using the non-inverted document descriptions. It is not necessary to compute all exact RSV's because documents can already be provided to the user as soon as their exact RSV is bigger than the RSVO of the actually regarded document.

[7] For TRW2 (statistical queries), we built a combined word frequency table, phrase frequency table (2 and 3 word phrases), and a special features frequency table. These were based on a selected subset of the training database and were used to calculate term weights. They had no direct role in the execution of the queries.

IB. CONSTRUCTION OF INDICES, KNOWLEDGE BASES, AND OTHER DATA STRUCTURES -- STATISTICS ON DATA STRUCTURES (CON'T)

SYSTEM NAME	ADS	UIC	Dalhousie	MEAD	UIF	IDSR	
Knowledge Bases	No		No		No		
- Total Storage							
- Number of Concepts represented							
- Type of Representation							
- Total Computer Build Time							
- Total Manual Time							
- Use of Manual Labor (*)							
- Auxiliary Files Needed (+)							
Routing Structures	No		query/filter files 1 per query		Yes, Neural Network		
Total Storage (MB)			10-500 characters each		15 KB		
- Total Computer Build Time			negligible		5 minutes		
- Automatic Process			No		Yes		
- Number of Manual Hours							
- Description of Method			[3]		[5]		
Other Data Structures	Yes, classification vectors; actually integer arrays		record offset file (1 per article in database)		None	Yes	
- Total Storage (MB)	A few KB [1]		770 KB			2 MB	
- Total Computer Build Time (Hours)	4 [2]		3 seconds CPU			1.5 hours	
- Automatic Process	Yes		Yes			Yes	
- Description of Method			[4]			[6]	

(+) (a) machine-readable dictionary
(b) other

(*) (a) Mostly manually built using special interface (b) Mostly machine built with manual correction
(c) Initial core manually built to "bootstrap (d) Other

IB. CONSTRUCTION OF INDICES, KNOWLEDGE BASES AND OTHER DATA STRUCTURES -- STATISTICS ON DATA STRUCTURES (CON'T)

NOTES:

[1] Only a few KB for the training sets used for the official scores - we used TOPIC for the actual test.

[2] Feature extraction takes of the order of 10 seconds per document - total time for the training data (disk 2 only) was of the order of 4 hours.

[3] Ran queries as adhoc queries against the test data (WSJ) then typed in query sequence to be used as filter.

[4] Scan data file and save offset for each occurrence of <DOC> string.

[5] The neural network was used to represent the topics. East output node is associated with a topic. Each input node is associated with a Roget category.

[6] Document frequency for each topic for a list of 1400 candidate features.

IC. CONSTRUCTION OF INDICES, KNOWLEDGE BASES, AND OTHER DATA STRUCTURES--DATA FUILT FROM OTHER SOURCES

SYSTEM NAME	DORTMUND	CORNELL	BERKELEY	RUTGERS [1]	SIEMENS	UMASS	VPI
Internally Built Auxiliary Files			None				N/A
- Domain Independent or Domain Specific						Domain Independent	
- Type of File						Countries & cities [3]	
- Total Storage (MB)						<1	
- Number of Concepts						2 (country, city)	
- Type of Representation						List of strings	
- Total Computer Build Time (Hours)						<1 [4]	
- Computer Time to Modify (Hours)							
- Manual Time to Build						~2 [5]	
- Manual Time to Modify							
- Use of Manual Labor (*)						b	
Externally-built Auxiliary Files			None		Yes		N/A
- Type of File					Wordnet 1.3		
- Total Storage (MB)					6.5 (Nouns Only, our format)		
- Number of Concepts					77,656 noun senses in 41,263 synonym		
- Type of Representation					Semantic net [2]		

(*) (a) Mostly manually built using special interface
(b) Mostly machine built with manual correction
(c) Initial core manually built to "bootstrap" for completely machine-built completion
(d) Other

IC. CONSTRUCTION OF INDICES, KNOWLEDGE BASES, AND OTHER DATA STRUCTURES--DATA BUILT FROM OTHER SOURCES

NOTES:

[1] Because we used the UMASS INQUERY system and its indexing, all of the answers to the questions in this section for our systems are identical to those for the UMASS system.

[2] Closest to a semantic net: sets of synonyms interrelated through lexical relations.

[3] Lists of countries and cities, extracted from gazeteer.

[4] Built for TREC-1 in <1 hour. Not modified for TREC-2.

[5] Built for TREC-1 in ~2 hours. Not modified for TREC-2.

IC. CONSTRUCTION OF INDICES, KNOWLEDGE BASES, AND OTHER DATA STRUCTURES -- DATA BUILT FROM OTHER SOURCES

SYSTEM NAME	CLARIT	CLARIT	HNC	HNC	QUEENS	SYRACUSE	QUEENS
Internally Built Auxiliary Files	Lexicon for English	Word frequency statistics for common English	Stemming exception list	Word pair list		Yes	
- Domain Independent or Domain Specific	Domain independent	Domain independent	Domain independent	Domain independent	Stopword file	Domain independent	Phrase file
- Type of File	Lexicon	Database of words with frequency statistics	Exception list	Word pair list	Lexicon	1) Lexicon 2) knowledge bases	Word pair
- Total Storage (MB)	2 MB	2 MB	28 KB	51 KB	0.004	15.3 [3]	0.2
- Number of Concepts	>100,000 [1]	139,481 Words	1,300	3700	630	82,669 [4]	14,000
- Type of Representation	Database records	Database records	List	List		[5]	
- Total Computer Build Time (Hours)	N/A	~20 min.	N/A	N/A	0	20 Hours [6]	4
- Computer Time to Modify (Hours)	None	None					
- Manual Time to Build	N/A	None	~ 96 Hrs.	96-120 Hrs.		48 Hours [7]	
- Manual Time to Modify	None	None					
- Use of Manual Labor (*)	[2]	None	b	b		b (lexicon & proper noun KB)	
Externally-built Auxiliary Files	None	None	None	None			None
- Type of File							
- Total Storage (MB)							
- Number of Concepts							
- Type of Representation							

(*) (a) Mostly manually built using special interface
(b) Mostly machine built with manual correction
(c) Initial core manually built to "bootstrap" for completely machine-built completion
(d) Other

IC. CONSTRUCTION OF INDICES, KNOWLEDGE BASES, AND OTHER DATA STRUCTURES -- DATA BUILT FROM OTHER SOURCES

NOTES:

[1] Over 100,000 root forms for English words (lexical items).

[2] The CLARIT Lexicon was manually constructed using word lists extracted from on-line sources during early phases of the CLARIT research project (1988-1989).

[3]
1) 13.9 MB (for lexicon built from LDOCE)
2) 0.3 MB (for proper noun knowledge base)
3) 1.1 MB (for verb and normalization verb case frame)

[4]
1) 43,941 (lexicon)
2) 9,889 (proper noun knowledge base)
3) 28,839 (case frame)

[5]
1) inverted index (lexicon)
2) frames (proper noun knowledge base)
3) case frames used as rules for concept-relation-concept triples (case frame)

[6]
1) 10 hours (lexicon)
2) 10 hours (proper noun knowledge base)

[7]
1) 24 hours (lexicon)
2) 124 hours (proper noun knowledge base)

IC. CONSTRUCTION OF INDICES, KNOWLEDGE BASES, AND OTHER DATA STRUCTURES -- DATA BUILT FROM OTHER SOURCES

SYSTEM NAME	CITY	ERIM	TMC	CITRI	UCLA	SEC	ETH	TRW
Internally Built Auxiliary Files	One manaully built file (for each system)	None			Yes [4]		No	
- Domain Independent or Domain Specific	Mostly very general [1]				Mostly very general [1]			
- Type of File	Synonym classes, go phrases, stopwords & semi-stopwords [2]				Synonym classes, go phrases, stopwords & semi-stopwords [2]			
- Total Storage (MB)	<<1 MB				<< 1MB			
- Number of Concepts	~1500				~1200			
- Type of Representation	look up table				Look up table			
- Total Computer Build Time (Hours)	Manually built - "compiled" at runtime, ~1 sec.				Manually built "compiled" at runtime, ~1 sec.			
- Computer time to Modify (Hours)								
- Manual Time to Build	Not recorded [3]				Not recorded			
- Manual Time to Modify								
- Use of Manual Labor (*)	Manual using text editor				Manual using text editor			
Externally-built Auxiliary Files	No	None			No		No	
- Type of File								
- Total Storage (MB)								
- Number of Concepts								
- Type of Representation								

(*) (a) Mostly manually built using special interface
(b) Mostly machine built with manual correction
(c) Initial core manually built to "bootstrap" for completely machine-built completion
(d) Other

IC. CONSTRUCTION OF INDICES, KNOWLEDGE BASES, AND OTHER DATA STRUCTURES -- DATA BUILT FROM OTHER SOURCES

NOTES:

[1] Somewhat angled towards American data, but mostly very general.

[2] Contains synonym classes (e.g. [child, children]), go phrases (e.g. Des Moines), stopwords and semi-stopwords.

[3] Not recorded. Because of disk shortage, System 1 included a number of additional stopwords suggested by high frequencies in the TREC data.

[4] One manually-built file that contains common business acronyms and abbreviations and abbreviations of organizations. The business acronyms were compiled by UCLA. The list of organizations is based on entries from the files "un.txt" and "organizations.txt" that are available from Project Gutenberg (the *.txt files were made available to UCLA by Bruce Croft, UMASS).

IC. CONSTRUCTION OF INDICES, KNOWLEDGE BASES, AND OTHER DATA STRUCTURES -- DATA BUILT FROM OTHER SOURCES

SYSTEM NAME	ADS	UIC	DALHOUSIE	MEAD	UIF	IDSR	
Internally Built Auxiliary Files	None		No		Semantic Lexicon	None	
- Domain Independent or Domain Specific					Domain independent		
- Type of File					Semantic Lexicon built from 1911 Roget's Thesaurus		
- Total Storage (MB)					0.5 MB		
- Number of Concepts					~ 1000 [1]		
- Type of Representation					Could be reviewed as rules		
- Total Computer Build Time (Hours)					22 Hrs. [2]		
- Computer time to Modify Time (Hours)							
- Manual Time to Build					None		
- Manual Time to Modify							
- Use of Manual Labor (*)					(a); See [2]		
Externally-built Auxiliary Files	None		No		No	Stopword list	
- Type of File							
- Total Storage (MB)							
- Number of Concepts							
- Type of Representation							

(*) (a) Mostly manually built using special interface
(b) Mostly machine built with manual correction
(c) Initial core manually built to "bootstrap" for completely machine-build completion
(d) Other

IC. CONSTRUCTION OF INDICES, KNOWLEDGE BASES, AND OTHER DATA STRUCTURES -- DATA BUILT FROM OTHER SOURCES

NOTES:

[1] There are around 1000 semantic categories used. The original 1911 Roget major categories are used by removing the suffix on our semantic codes. For example, the semantic category 121nv.3 is shortened by ignoring nv.3.

[2] Since the 1911 edition of Roget's Thesaurus became public domain recently, we spent approximately 16 hours creating the software to process the 1911 Thesaurus. Approximately 6 hours of processing time was required to automatically extract 20,000 lexicon entries.

IIA. QUERY CONSTRUCTION -- AUTOMATICALLY BUILT QUERIES (ADHOC)

SYSTEM NAME	Dortmund	Cornell	Berkeley	Rutgers	Siemens	UMASS	VPI
Topic Fields			All			[4]	
Total Computer Time to Build (cpu sec)	~12h 20 min [1]	6/50 per query	~0.7 seconds per query			12.1	
Methods Used to Construct							
- Term Weighting (based on topic terms)	Yes [2]	Yes	Yes			Yes [5]	
- Phrase extraction	Yes	Yes	No			Simple noun phrases	
- Syntactic Parsing			No			Tagging & noun phrase bracketing	
- Word Sense Disambiguation			No			No	
- Proper Noun Identification			No			(adjacent) capitalized words	
- Tokenizer			No			None	
-- Which Patterns			None				
- Heuristic Associations			No			None	
- Expansion of Queries Using Previous Constructed Data Structure		Yes	None			None	
-- Which Structure		Phrases from controlled list					
- Automatic addition of Boolean Connectors/Proximity Operators			No			Yes [6]	
- Other		Reweighting of terms in a local context	[3]				

IIA. QUERY CONSTRUCTION -- AUTOMATICALLY BUILT QUERIES (ADHOC)

NOTES:

[1] Computing of query regression co-efficients: 12h 20 min reweighting of query file: 3 sec.

[2] Query regression, query weighting, polynomial regression document weighting.

[3] Optimally relativised frequencies used to weight stems.

[4] Title, description, concepts, factors, narrative.

[5] Weights initially based on number of occurrences in topic. 3x multiplier for title field, 2x multiplier for concepts field.

[6] Capitalized noun groups, hyphenated words, concept words delimited by commas, words occurring in title.

IIA. QUERY CONSTRUCTION -- AUTOMATICALLY BUILT QUERIES (ADHOC)

SYSTEM NAME	GE	CLARIT	LSI	HNC	NYU	SYRACUSE	QUEENS
Topic Fields	Title, desc, narr, con	Title, desc, narr, con	All	Turned in various runs using everything but factors, definitions, nationality	Num, title, desc, narr	Topic, desc, narr, con	Natural language: <title>, <desc>, <narr>, <con>; Boolean:<desc>,<con>
Total Computer Time to Build (cpu sec)	<0.1 sec	0.80 sec CPU time per query	About 0.1 sec per query [4]	< 1 sec.	3.0	Less than a sec. (Stage 1)	3 (average for each query)
Methods Used to Construct							
- Term Weighting (based on topic terms)	Yes, for ranking only	Yes [1]	Yes	Not used		Yes	Yes
- Phrase extraction	Yes, for concepts only			Not used	Yes	Yes	No
- Syntactic Parsing		Yes [2]		Not used	Yes	Partial	No
- Word Sense Disambiguation		Rule based syntactic category disambiguation		Not used		Yes	No
- Proper Noun Identification		Unknown words treated as proper nouns			Partial	Yes	No
- Tokenizer		No		Not used	Yes	Yes, part of the proper noun identification alg.	No
-- Which Patterns?					Year number	Dates	
- Heuristic Associations		No		Not used	Yes		No
- Expansion of Queries Using Previous Constructed Data Structure	Yes	No		Not used		Yes (Stage 1); Yes (partially for one run - Stage 2	Yes
-- Which Structure?	Automatic term expansion based on corpus associations				Term clusters	[5]	Word-pair phrase file
- Automatic addition of Boolean Connectors/Proximity Operators		No		Not used		Yes (Stage 1)	Yes
- Other		[3]		Boolean operators on concepts portion of topics	Syntactic phrases	Similar terms in a given topic statement were identified manually for inexact matching of concepts (Stage 2)	None

IIA. QUERY CONSTRUCTION -- AUTOMATICALLY BUILT QUERIES (ADHOC)

NOTES:

[1] Topic terms are weighted using IDF/TF * Importance, where the Importance coefficient has a value of 1, 2, or 3, and is assigned based on heuristics relating to position of the term in the topic. Specifically, terms in the first paragraph receive an Important coefficient of 3; terms in the second (middle) paragraph receive a score of 2; all other terms receive a score of 1.

[2] Terms are extracted from the topics using the natural language processing described above. However, all terms that occur in a sentence that also contains a negative construction (e.g. "NOT") are eliminated from the query vector.

[3] Queries are expanded with terms that derive from a thesaurus that is extracted from documents that are retrieved from the corpus. The process is strictly automatic. A first-pass querying of the corpus is made using terms from the topic; top ranking documents are automatically selected and processed using thesaurus extraction to extract additional terminology. The procedure is essentially identical to the method used in the processing of routing topics, except that the set of 'relevant' documents is automatically identified by the system.

[4] Time required to take vector sum of terms in query.

[5]
1) Synonyms complex nominal listings (Stage 1)
2) Variant/expandable proper noun listings (Stage 1)
3) RIT (Stage 2).

IIA. QUERY CONSTRUCTION -- AUTOMATICALLY BUILT QUERIES (ADHOC)

SYSTEM NAME	CITY	ERIM	TMC	CITRI	UCLA	SEC	ETH	TRW
Topic Fields	Title and Concepts	Concept strings; nationality strings	All	All	Title and concepts		All	
Total Computer Time to Build (cpu sec)	< 1 second each	~ 2 seconds	15 min. for initial queries; 4 hours for optimization	Negligable	< 1 second each		10-20 msec./ query	
Methods Used to Construct				None				
- Term Weighting (based on topic terms)			No				Yes, nff*nidf	
- Phrase extraction			Yes (adjacent words)				No	
- Syntactic Parsing		Yes, simple tag identification					No	
- Word Sense Disambiguation							No	
- Proper Noun Identification							No	
- Tokenizer							No	
-- Which Patterns								
- Heuristic Associations							No	
- Expansion of Queries Using Previous Constructed Data Structure							No	
-- Which Structure								
- Automatic addition of Boolean Connectors/Proximity Operators							No	
- Other	Robertson/Sparck-Jones F4 predictive weights (approx. IDF in the absence of relevance information).		IDF weights based on collection frequencies in training docs for routing db.		Robertson/Sparck-Jones F4 predictive weights (approx. IDF in the absence of relevance information).		No	

IIA. QUERY CONSTRUCTION -- AUTOMATICALLY BUILT QUERIES (ADHOC)

SYSTEM NAME	ADS	UIC	DALHOUSIE	MEAD	UIF	IDSR
Topic Fields		Concept fields: each unique word stem				
Total Computer Time to Build (cpu sec)		1 second				
Methods Used to Construct		None				
- Term Weighting (based on topic terms)						
- Phrase extraction						
- Syntactic Parsing						
- Word Sense Disambiguation						
---- Proper Noun Identification						
- Tokenizer						
-- Which Patterns						
- Heuristic Associations						
- Expansion of Queries Using Previous Constructed Data Structure						
-- Which Structure						
- Automatic addition of Boolean Connectors/Proximity Operators						
- Other						

IIB. QUERY CONSTRUCTION--MANUALLY CONSTRUCTED QUERIES (ADHOC)

SYSTEM NAME	DORTMUND	CORNELL	BERKELEY	RUTGERS	SIEMENS	UMASS	VPI
Manually Constructed Queries (Adhoc)			None		[2]		3 sets [7]
Topic Fields				All	Entire topic statement [3]	Yes [5]	Yes [8]
Average Build Time (minutes)				112.5 for combined query [1]	5-10 min. per uery [4]	1 min.	15 per query
Type of Query Builder (*)				Human search expert	b	b	b
Tools Used (+)				No special tools	Wordnet interface	None	N/A
Methods Used in Construction							
- Term Weighting				Yes	Yes	Yes	pnorm: system assigned weights
- Boolean Connectors				Yes		No	pnorm: AND/OR
- Proximity Operators				Yes		Yes	No
- Addition of Terms				Yes	Yes	No	all: possibly
-- Source of Terms				Personal knowledge of searchers	Wordnet synonym sets	[6]	[9]
- Other				None			[10]

(*) (a) domain expert (b) computer systems expert
(+) (a) word frequency list (b) knowledge base browser (c) other lexical tools

NOTES:

[1] Average time to build query (minutes): 112.5 for combined query (52.5 for the five individual query formulations, and 60 for translating them into INQUERY format). These correspond to 10.5 and 12 minutes, respectively, for each individual query formulation.

[2] Queries had manually chosen synsets added to the input text. All processing of the text was automatic.

[3] Entire topic statement read when selecting synsets, Concepts (<con>), Description (<desc>), Factors (<fac>), Narrative (<narr>), Nationality (<nat>), Title (<title>) used in automatic part.

[4] 5 - 10 minutes a query to select the synonym sets to add an average of 1.2 seconds to automatically process the topic text (1 minute for 50 queries [there were other jobs on the machine!]).

[5] Title, Description, Concepts, Factors, Narrative.

[6] Addition and deletion of terms selected from the narrative field.

[7] Three sets of queries were constructed: one pnorm boolean query set and two vector query sets, one longer than the other. They are called pnorm, long vector and short vector queries below.

[8] All query sets: title, description, concepts pnorm and long vector: Narrative; long and short vector: Definitions.

[9] Domain knowledge of computer system expert, limited use to compensate for omissions in topic descriptions.

[10] Boolean operators were assigned equal weights (P-values) for the pnorm queries, P-values of 1.0, 1.5 and 2.0 were used for different evaluations of the queries.

IIB. QUERY CONSTRUCTION--MANUALLY CONSTRUCTED QUERIES (ADHOC)

SYSTEM NAME	GE	CLARIT	LSI	HNC	NYU	SYRACUSE	QUEENS
Manually Constructed Queries (Adhoc)				N/A	No	No	No
Topic Fields		Title, desc, narr, con					
Average Build Time (minutes)		2 - 5 min. per query					
Type of Query Builder (*)		b [1]					
Tools Used (+)		None					
Methods Used in Construction							
- Term Weighting							
- Boolean Connectors							
- Proximity Operators							
- Addition of Terms							
-- Source of Terms							
- Other		Yes [2]					

(*) (a) domain expert (b) computer systems expert
(+) (a) word frequency list (b) knowledge base browser (c) other lexical tools

IIB. QUERY CONSTRUCTION--MANUALLY CONSTRUCTED QUERIES (ADHOC)

NOTES:

[1] Yes -- but this is merely the fact of the matter: the people who rendered judgments about the CLARIT-nominated query terms happened to be computer experts. The method does not require special expertise of any sort.

[2] Manual queries are constructed by correcting the query terms nominated via automatic query construction. Team members analyzed the output of the automatic process and (1) deleted/corrected any parser errors, (2) adjusted Importance coefficients in situations where the weighting heuristics fail, (3) corrected errors in the application of the term negation heuristic, and (4) added additional relevant terms where appropriate. (New or additional terms were rarely added.)

IIB. QUERY CONSTRUCTION--MANUALLY CONSTRUCTED QUERIES (ADHOC)

SYSTEM NAME	CITY	ERIM	TMC	CITRI	UCLA	SEC	ETH	TRW
Manually Constructed Queries (Adhoc)		None			N/A		No manually built queries	
Topic Fields	Any							
Average Build Time (minutes)	Varies							
Type of Query Builder (*)	Three faculty, two research students (Inf. Sci.)							
Tools Used (+)	None							
Methods Used in Construction								
- Term Weighting	Yes [1]							
- Boolean Connectors	Yes							
- Proximity Operators	Yes (adj, same sentence)							
- Addition of Terms	Yes							
-- Source of Terms	Searchers knowledge, records seen during search,...							
- Other	[2]							

(*) (a) domain expert (b) computer systems expert
(+) (a) word frequency list (b) knowledge base browser (c) other lexical tools

IIB. QUERY CONSTRUCTION--MANUALLY CONSTRUCTED QUERIES (ADHOC)

NOTES:

[1] The system does this. It was possible for searchers to influence the weights, but I believe none of them did so (they were not told how to).

[2] The default operator (when searcher didn't specify an op) was a "best-match" opeator utilizing Robertson/Sparck-Jones F4 with a within-document term frequency factor and an overall document length factor.

IIB. QUERY CONSTRUCTION--MANUALLY CONSTRUCTED QUERIES (ADHOC)

SYSTEM NAME	ADS	UIC	DALHOUSIE	MEAD	UIF	IDSR
Manually Constructed Queries (Adhoc)	N/A				No	
Topic Fields			Yes			
Average Build Time (minutes)			15 minutes			
Type of Query Builder (*)			b			
Tools Used (+)			No			
Methods Used in Construction						
- Term Weighting			No			
- Boolean Connectors			Yes			
- Proximity Operators			No			
- Addition of Terms			No			
-- Source of Terms						
- Other			None			

(*) (a) domain expert (b) computer systems expert
(+) (a) word frequency list (b) knowledge base browser (c) other lexical tools

IIC. QUERY CONSTRUCTION -- FEEDBACK (ADHOC)

SYSTEM NAME	Dormund	Cornell	Berkeley	Rutgers	Siemens	UMASS	VPI
Feedback (adhoc)			Not used	None		None	N/A
Initial Query Manual or Automatic							
Type of person doing feedback (*)							
Average Time for complete feedback							
- cpu time (cpu seconds)							
- clock time (minutes)							
Average number of iterations							
- Average number of Documents Per Iteration							
Minimum number of Iterations							
Maximum number of Iterations							
What determines end of Iteration							
Feedback methods used							
- Automatic Term Reweighting							
- Automatic Query Expansion (+)							
- Other automatic methods							
- Manual Methods							

(+) (a) all terms in relevant documents added (b) only top X terms added (c) user selected terms added
(*) (a) domain expert (b) system expert

IIC. QUERY CONSTRUCTION -- FEEDBACK (ADHOC)

SYSTEM NAME	GE	CLARIT	LSI	HNC	NYU	SYRACUSE	QUEENS
Feedback (adhoc)					No	No	
Initial Query Manual or Automatic				Automatic			
Type of person doing feedback (*)				b			
Average Time for complete feedback							
- epu time (cpu seconds)				< 1 second (only did one iteration			
- clock time (minutes)				~10 min.			
Average number of iterations				1			
- Average number of Documents Per Iteration				20			
Minimum number of Iterations				N/A			
Maximum number of Iterations				N/A			
What determines end of Iteration				N/A			
Feedback methods used							
- Automatic Term Reweighting							
- Automatic Query Expansion (+)							
- Other automatic methods				Adding relevant document context vectors in the query context vector			
- Manual Methods				N/A			

(+) (a) all terms in relevant documents added (b) only top X terms added (c) user selected terms added

IIC. QUERY CONSTRUCTION -- FEEDBACK (ADHOC)

SYSTEM NAME	CITY	ERIM	TMC	CITRI	UCLA	SEC	ETH	TRW
Feedback (adhoc)		None					No feedback methods	
Initial Query Manual or Automatic	Manual (See IIB)				Automatic (See IIA)			
Type of person doing feedback (*)	Searchers (as IIB)				Searchers: 1 faculty 1 Ph.D. student			
Average Time for complete feedback								
- cpu time (cpu seconds)	Varies, about 20-300 [1]				Information not available			
- clock time (minutes)	10-120				Information not available			
Average number of iterations	2.0				2.0 (initial query + 1 feedback iteration)			
- Average number of Documents Per Iteration	13.3				17.8			
Minimum number of Iterations	0				2			
Maximum number of Iterations	4				2			
What determines end of Iteration	Searcher decides				When 10 relevant documents were found, or when 20 documents had been examined.			
Feedback methods used								
- Automatic Term Reweighting								
- Automatic Query Expansion (+)	2, 3 [2]				2, 3 [3]			
- Other automatic methods								
- Manual Methods								

(+) (a) all terms in relevant documents added (b) only top X terms added (c) user selected terms added

IIC. QUERY CONSTRUCTION -- FEEDBACK (ADHOC)

NOTES:

[1] Varies widely, about 20-300. Mean not recorded but probably about 80.

[2] Term pool consisted of all terms from relevant documents and all terms entered by searcher in search statements. Terms were weighted using Robertson/Sparck-Jones F4. Searchers' terms were given a bonus as if they had occurred in four out of five hypothetical relevant records. The top 20 terms were selected using the criterion:

selection_value=term_weight * rels_containing_term/rels

[3] Term pool consisted of all terms from relevant documents. The top 10 terms as determined by w(p-q) were chosen for expansion and were searched together with the initial query terms. Terms were weighted using Robertson/Sparck-Jones F4.

IIC. QUERY CONSTRUCTION -- FEEDBACK (ADHOC)

SYSTEM NAME	ADS	UIC	DALHOUSIE	MEAD	UIF	IDSR
Feedback (adhoc)	N/A		Not used		No	
Initial Query Manual or Automatic						
Type of person doing feedback (*)						
Average Time for complete feedback						
- cpu time (cpu seconds)						
- clock time (minutes)						
Average number of iterations						
- Average number of Documents Per Iteration						
Minimum number of Iterations						
Maximum number of Iterations						
What determines end of Iteration						
Feedback methods used						
- Automatic Term Reweighting						
- Automatic Query Expansion (+)						
- Other automatic methods						
- Manual Methods						

(+) (a) all terms in relevant documents added (b) only top X terms added (c) user selected terms added
(*) (a) domain expert (b) system expert

IID. QUERY CONSTRUCTION--AUTOMATICALLY BUILT QUERIES (ROUTING)

SYSTEM NAME	Dortmund	Cornell	Berkeley	Rutgers	Siemens	UMASS	VPI
Automatically Built Queries (routing)			Yes	None		[1]	N/A
Topic Fields Used	Everything except Definitions	Everything except Definitions	All			See [1]	
Total Computer Time to Build (cpu seconds)	28/50 for each query	212/50 (crnlR1) 50/50 (crnlC1) per query	0.7 seconds per query			4623 for 50 queries	
Methods Used in Building Query							
- Terms Selected from (*)	1, 3	1, 3	1			3	
- Term Weighting with weights based on (*)	1, 2	1, 2	1			3	
- Phrase Extraction from (*)	1, 3	1, 3	No			No	
- Syntactic Parsing of (*)			No			No	
- Word Sense Disambiguation using (*)			No			No	
- Proper Noun Identification Algorithm from (*)			No			No	
- Tokenizer from (*)			No			No	
-- Which Patterns?							
- Heuristic Associations to add terms from (*)			No			No	
- Expansion of Queries Using Previously Constructed Data Structure	History of term occurrence in relevant docs	History of term occurence in relevant docs	No			No	
-- Which Structure?							
- Automatic Addition of Boolean Connectors Proximity Operators Using Information from (*)			No			No	
- Other			Additional term specific weights added for 2nd routing run			None	

(*) (1) Topic (2) All training Documents (3) Documents with relevance judgements

IID. QUERY CONSTRUCTION--AUTOMATICALLY BUILT QUERIES (ROUTING)

NOTES:

[1] The general approach was to use the automatic method of query formation, and then apply relevance feedback to the results obtained from running the automatic query on a different TREC collection.

IID. QUERY CONSTRUCTION--AUTOMATICALLY BUILT QUERIES (ROUTING)

SYSTEM NAME	GE	CLARIT	LSI	HNC	NYU	SYRACUSE	QUEENS
Automatically Built Queries (routing)			Yes [5]				
Topic Fields Used		Title, desc, narr, con	"lsir1" - all "lsir2" - no topic info; used only text of relevant documents (from training)	Same as adhoc	Same as adhoc	topic, desc, narr, con	<title>, <desc>, <narr>, <con>
Total Computer Time to Build (cpu seconds)		180 seconds per query[1]	~0.1 - 0.2 sec/query [6]	10-300 seconds (depending on amount of training data)	3.2	Less than a second (Stage 1)	3 (average for each query)
Methods Used in Building Query							
- Terms Selected from (*)		1, 3 [2]	1 ("lsir1") 3 ("lsir2")	1	2	1	1
- Term Weighting with weights based on (*)		1, 2, 3 [3]	3 ("lsir1" & "lsir2")	No	2	1, 2 (Stage 2)	1, 2, 3 [7]
- Phrase Extraction from (*)				No	1, 2	1, 2	No
- Syntactic Parsing of (*)		1, 2, 3 [4]		No	1, 2	1	No
- Word Sense Disambiguation using (*)		Rule based syntactic category disambiguation		No		1	No
- Proper Noun Identification Algorithm from (*)		Unknown words treated as proper nouns		No	1 (partial) 2 (partial)	1	No
- Tokenizer from (*)		No		No		1	No
-- Which Patterns?						Dates	
- Heuristic Associations to add terms from (*)		No		No	2	No	No
- Expansion of Queries Using Previously Constructed Data Structure		Yes		No			Yes
-- Which Structure?		The first-order thesaurus described in I.			Clusters from training data		Word-pair phrase file
- Automatic Addition of Boolean Connectors Proximity Operators Using Information from (*)		No		No		Yes (Stage 1)	1
- Other		No		Boolean operators on concepts portion of topics		No	No

(*) (1) Topic (2) All training Documents (3) Documents with relevance judgments

IID. QUERY CONSTRUCTION--AUTOMATICALLY BUILT QUERIES (ROUTING)

NOTES:

[1] 180 seconds per query -- encompassing the parsing of all known relevant documents, thesaurus extraction, and merging of the thesaurus and query terms.

[2] (3) Thesaurus extraction over the set of relevant documents results in a set of terms to add to a query.

[3] (1) The Importance Coefficient (described above) is derived from the 'discourse location' of terms in the topic.
(2) Statistics for IDF/TF calculation are derived from the training documents.
(3) All terms extracted from known relevant documents via thesaurus extraction are assigned an Importance Coefficient of 0.5.

[4] (2) Yes. (This is necessary in order to extract statistics.)

[5] We submitted two routing runs. lsir1 - simply used the text of the query topics to construct a routing filter. lsir2 - used the text of all relevant documents for a topic to construct the routing filter for that topic.

[6] Time required to take vector sum of terms in query (lsir1) or relevant documents (lsir2).

[7] (1) Yes
(2) Subset of 1 GB from WSJ1, AP1, DOE, FRI, ZIF1.
(3) Subset of them.

IID. QUERY CONSTRUCTION--AUTOMATICALLY BUILT QUERIES (ROUTING)

SYSTEM NAME	CITY	ERIM	TMC	CITRI	UCLA	SEC	ETH	TRW
Automatically Built Queries (routing)					N/A			
Topic Fields Used	None	Concept strings; Nationality strings	All			All	All	
Total Computer Time to Build (cpu seconds)	Time for individual queries not known	1 - 1.5 seconds	8 hours for initial queries; 2 hours for optimization			600	10-20 msec./query	
Methods Used in Building Query								
- Terms Selected from (*)	3 [1]	1	1			1	1 (stoplist and suffix stripping)	
- Term Weighting with weights based on (*)	3, weights were Robertson/Sparck-Jones F4		2			2	1 (nff); 2 (nidf)	
- Phrase Extraction from (*)			1			1, 2	No	
- Syntactic Parsing of (*)		1, using simple identification of concepts				1, 2	No	
- Word Sense Disambiguation using (*)							No	
- Proper Noun Identification Algorithm from (*)						1	No	
- Tokenizer from (*)							No	
-- Which Patterns?								
- Heuristic Associations to add terms from (*)							No	
- Expansion of Queries Using Previously Constructed Data Structure							No	
-- Which Structure?								
- Automatic Addition of Boolean Connectors Proximity Operators Using Information from (*)							No	
- Other	Yes [2]						No	

IID. QUERY CONSTRUCTION--AUTOMATICALLY BUILT QUERIES (ROUTING)

NOTES:

[1] All non-stop terms were extracted from all known relevant docs for each query (disks 1 & 2). For cityr1, the top 20 terms by the above selection criterion were used.

For cityr2, the same term pool and weights were used, but the number of terms varied between 3 and 31, based on the results of a long series of runs against disks 1 and 2.

[2] For cityr1, a simple best match operator was used (doc weight = sum of term weights).

For cityr2, the operator used was based on the results of experiments with the training set. For some queries, it was as for cityr1, for others, as in Section IIB. Other.

IID. QUERY CONSTRUCTION--AUTOMATICALLY BUILT QUERIES (ROUTING)

SYSTEM NAME	ADS	UIC	DALHOUSIE	MEAD	UIF	ISDR
Automatically Built Queries (routing)	Yes		Not used		Yes	
Topic Fields Used	narr, con, def	None [2]			Just the concept section	
Total Computer Time to Build (cpu seconds)	Less than 30 seconds [1]	70 seconds			1 second	
Methods Used in Building Query						
- Terms Selected from (*)	1, 3	3			1 (4)	
- Term Weighting with weights based on (*)	No	None			1 (4)	
- Phrase Extraction from (*)	No				No	
- Syntactic Parsing of (*)	No				No	
- Word Sense Disambiguation using (*)	No				No	
- Proper Noun Identification Algorithm from (*)	No				No	
- Tokenizer from (*)	No				No	
-- Which Patterns?						
- Heuristic Associations to add terms from (*)	No				No	
- Expansion of Queries Using Previously Constructed Data Structure	No				Yes	
-- Which Structure?					Semantic lexicon in Section IC.	
- Automatic Addition of Boolean Connectors Proximity Operators Using Information from (*)	No				No	
- Other	No				Term weighting with weights based on terms in topics	

(*) (1) Topic (2) All training Documents (3) Documents with relevance judgments

IID. QUERY CONSTRUCTION--AUTOMATICALLY BUILT QUERIES (ROUTING)

NOTES:

[1] Takes less than 30 seconds to build the CART tree - this does depend on the size of the training set.

[2] None. Queries were constructed by finding all word pairs within three words that were found only in the relevant documents.

IIE. QUERY CONSTRUCTION -- MANUALLY CONSTRUCTED QUERIES (ROUTING)

SYSTEM NAME	Dortmund	Cornell	Berkeley	Rutgers	Siemens	UMASS	VPI
Manually Constructed Queries (Routing)			None			[6]	3 sets [7]
Topic Fields Used				All	See adhoc		[8]
Average Time to Build (minutes)				125 min. for combined queries [1]	up to 15 min. [5]		15 min per query
Type of Query Builder				system expert	system expert		system expert
Data Used for Building Query (*)				a, c [2]	c, indirectly		a
Tools Used to Build Query (+)				none, other than searchers knowledge	b, WordNet interface		N/A
Methods Used to Build Query							
- Term Weighting				Yes [3]	Yes		pnorm: system assigned weights
- Boolean Connectors				Yes			pnorm: AND, OR
- Proximity Operators				Yes			No
- Addition of Terms				Yes	Yes		Yes
-- Source of Terms				[4]	WordNet synonym sets		[9]
- Other				None			

(a) word frequency list (b) knowledge base browser (c) other lexical tools (d) machine analysis of training documents
(a) from training topic (b) from all training documents (c) from documents with relevance judgements (d) from other sources

IIE. QUERY CONSTRUCTION -- MANUALLY CONSTRUCTED QUERIES (ROUTING)

NOTES:

[1] For combined queries, 125 minutes (65 minutes for the five individual queries, 60 minutes for translating them into INQUERY format, including incorporation of the weights.

[2] Specifically, the performance of query formulations on the training set was used to determine their weights in the scheme used for routing.

[3] In the sense that each individual query formulation within the combined query was weighted, and as implemented by INQUERY.

[4] From the personal knowledge of those forming the searches.

[5] The process of selecting synsets was iterated based on the effectiveness of the previous selection. Some topics probably took more than 15 minutes total over the course of the iterations. Creation of query from text averaged 0.7 seconds (35 seconds for 50 queries, less competition on the machine than in the adhoc case!).

[6] Combination of automatic query formation from Part IID and a manually modified query using the method from Part IIB.

[7] Again, three query sets were constructed as per adhoc manual.

[8] All: title, description, concepts
long vector, pnorm: Narrative
long and short vector: Definitions

[9] Domain knowledge of computer system expert, limited use to compensate for omissions in topic descriptions.

IIE. QUERY CONSTRUCTION -- MANUALLY CONSTRUCTED QUERIES (ROUTING)

SYSTEM NAME	GE	CLARIT	LSI	HNC	NYU	SYRACUSE	QUEENS
Manually Constructed Queries (Routing)				No	No	No	No
Topic Fields Used	All	Title, desc, narr, con					
Average Time to Build (minutes)	About 60 [1]	2 - 5 min. per query					
Type of Query Builder	System expert	Yes [3]					
Data Used for Building Query (*)	b, c [2]	a, b, c, d [4]					
Tools Used to Build Query (+)	a, b	None					
Methods Used to Build Query							
- Term Weighting	Yes, ranking only	IDF/TF and Importance Coefficients					
- Boolean Connectors	Yes	No					
- Proximity Operators		No					
- Addition of Terms	Yes	Yes					
-- Source of Terms	Corpus, relevant documents	Additional terminology nominated by team members as appropriate to topic					
- Other		Yes [5]					

(+) (a) word frequency list (b) knowledge base browser (c) other lexical tools (d) machine analysis of training documents

(*) (a) from training topic (b) from all training documents (c) from documents with relevance judgments (d) from other sources

IIE. QUERY CONSTRUCTION -- MANUALLY CONSTRUCTED QUERIES (ROUTING)

NOTES:

[1] About 60 (but this is a tough question because these were adhoc queries before).

[2] (b) Word frequencies.
(c) Topic weights (for ranking, manual correction).

[3] Yes -- but this is merely the fact of the matter: the people who rendered judgments about the CLARIT-nominated query terms happened to be computer experts. The method does not require special expertise of any sort.

[4] (a) Terms extracted from topic by natural-language processing.
(b) Statistics from the training documents used for IDF/TF scoring.
(c) Terms extracted from relevant documents by natural-language processing and thesaurus extraction.
(d) Additional terminology nominated by team members as appropriate to the topic.

[5] As with manual adhoc topics, manual routing queries reflect hand-correction of the output of the automatic query generating process. Correction took place prior to the expansion of the query vector with thesaurus terminology.

IIE. QUERY CONSTRUCTION -- MANUALLY CONSTRUCTED QUERIES (ROUTING)

SYSTEM NAME	CITY	ERIM	TMC	CITRI	UCLA	SEC	TRW1	TRW2
Manually Constructed Queries (Routing)		None					(Proximity queries)	(Statistical queries)
Topic Fields Used							Title, description, narrative, concepts	
Average Time to Build (minutes)							~250	~5 days [2]
Type of Query Builder							System expert	System expert
Data Used for Building Query (*)							a, b, c	b, c (only used documents judged relevant)
Tools Used to Build Query (+)							d [1]	a, along with 2 & 3 word phrases and special features
Methods Used to Build Query								
- Term Weighting							Implicitly derived from query rankings	Yes
- Boolean Connectors							Yes	Yes
- Proximity Operators							Yes	Yes
- Addition of Terms								
-- Source of Terms								
- Other							Counting operators (term frequency in documents) antecedent - reference recognition	

(+) (a) word frequency list (b) knowledge base browser (c) other lexical tools (d) machine analysis of training documents)
(*) (a) from training topic (b) from all training documents (c) from documents with relevance judgments (d) from other sources

IIE. QUERY CONSTRUCTION -- MANUALLY CONSTRUCTED QUERIES (ROUTING)

NOTES:

[1] Each topic was represented by a set of queries. Individual queries were parameterized invocations of a single template query. Selectivity vs. parameter value from training set was used to rank different queries.

[2] < 5 seconds for automatically generated first cut.
~ 5 minutes average tweek time per topic.
~ 5 days to develop special features queries used across all topics.

IIE. QUERY CONSTRUCTION -- MANUALLY CONSTRUCTED QUERIES (ROUTING)

SYSTEM NAME	ADS	UIC	DALHOUSIE	MEAD	UIF	IDSR
Manually Constructed Queries (Routing)	N/A				None	dom, title, desc, narr, con, def (and head, by mistake)
Topic Fields Used			Yes			
Average Time to Build (minutes)			15 minutes			10 minutes
Type of Query Builder			System expert			System expert
Data Used for Building Query (*)			a, b, c			a, c (used word frequencies)
Tools Used to Build Query (+)			No			a
Methods Used to Build Query						
- Term Weighting			No			Yes
- Boolean Connectors			Yes			No
- Proximity Operators			No			Directly adjacent pairs of words
- Addition of Terms			No			
-- Source of Terms						From other topics
- Other			None			Manual assessments of some relationships between topics

(+) (a) word frequency list
(*) (a) from training topic
(b) knowledge base browser
(b) from all training documents
(c) other lexical tools
(c) from documents with relevance judgments
(d) machine analysis of training documents)
(d) from other sources

III. SEARCHING

SYSTEM NAME	Dortmund	Cornell	Berkeley	Rutgers	Siemens	UMASS	VPI
Total Computer Time to Search (cpu seconds)	632/50 to 1196/50 per query [1]	825/50 to 8568/50 per query [2]		[5]			~ 4-7 min per topic [9]
- Retrieval Time (cpu seconds)			~ 16 sec/query for any collection separately		13.8 to 19.5 [6]	1 cpu sec per query term [8]	
- Ranking Time (cpu seconds)			included in search time		N/A [7]	2 cpu sec for top 1000	
Methods Used in Machine Searching							
- Vector Space Model	Yes	Yes	No		Yes		Yes [10]
- Probabalistic Model	Yes		Yes [3]	Probabilistic inference nets (INQUERY)		Yes	
- Cluster Searching			No				
- N-gram Matching			No				
- Boolean Matching			No				
- Fuzzy Logic			No				Yes [10]
- Free Text Scanning			No				
- Neural Networks			No				
- Conceptual Graph Matching			No				
- Other			Yes [4]				

III. SEARCHING

NOTES:

[1] 632/50 cpu seconds per query if no query expansion (dortVI). 1196/50 cpu seconds per query if expand by 20 terms (dortP1).

[2] 829/50 (i.e., 16.5) seconds for each query for crnlV2. 8568/50 seconds for each query for crnlL2 (involves re-indexing from scratch the top 1750 docs for each query and comparing each indexed local component against the query). 3273/50 seconds for each query for crnlR1 on a Sparc 1 with 12MB of memory (all other work was done on a Sparc 2 with 64 Mbytes). 2928/50 seconds for each query for crnlC1 on a Sparc 1 with 12MB of memory.

[3] Probabalistic searching based on linked dependence assumption and logistic regression.

[4] An equation derived by logistic regression was used to estimate a probability of relevance for each query and document.

[5] Since our experiments were done on a time-shared system, we present here both the mean cpu time of seven runs of each experiment, as an estimate of search and sorting time in a realistic multi-user environment; and, the minimum cpu time of the seven runs, as an upper bound on search and sorting time for a single-user system. In the systems we were using, we were unable to separate search and sorting time and therefore, present the total cpu time for producing the sorted lists of 1000 retrieved items.

Our timing results are reported as follows: first for the routing topics, using all 50 topics, giving total cpu time for the experiment (mean and minimum) and mean and minimum cpu time per topic (our rutcombx run); then for the adhoc topics, in which we used only 25 topics, the same figures (our rutcomb1 run). For the rutfmed run (24 adhoc topics, each having five individual searches, the resulting lists being subsequently combined), we estimate the upper bound of total time as being five times the minimum per topic time for runcomb1, plus the time required for combining the lists. These are presented as total cpu time for the experiment, and mean cpu time per topic.

rutcombx	Total:	981.55	(973.55)
	Per Topic:	19.631	(19.471)
rutcomb1	Total:	1226.225	(1208.325)
	Per Topic:	49.049	(48.333)
rutmedf	Total:	5799.96 + 373.2 = 6173.16	
	Per Topic	241.665 + 15.55 = 257.215	

[6] 13.8 cpu seconds on average for routing queries against test documents (691 cpu seconds for 50 queries). 17.4 cpu seconds on average for unexpanded adhoc queries versus documents on disks 1 & 2 (869 cpu seconds for 50 queries). 19.5 cpu seconds on average for expanded adhoc queries versus documents on disk 1 & 2 (974 cpu seconds for 50 queries).

[7] Not applicable; list of top 1000 maintained during search.

[8] 1 cpu second per query term on per gigabyte. Queries averaged about 45 terms each.

[9] Approximately 4 - 7 minutes per topic for combination runs (multiple queries) per topic, for a given collection.

[10] Combination of results from both pnorm (fuzzy logic) and vector (vector space model) queries.

III. SEARCHING

SYSTEM NAME	GE	CLARIT	LSI	HNC	NYU	SYRACUSE	QUEENS
Total Computer Time to Search (cpu seconds)	[1]		Can compute about 60,000 query-document similarities per minute when vectors are in memory				
- Retrieval Time (cpu seconds)		~4 Hrs. [2]	adhoc - ~10 min. routing - ~0.1 sec [3]	N/A (there is no unranked document list	Total time (cpu & I/O) search and ranking is about 1 minute per query		30-200 per query [5]
- Ranking Time (cpu seconds)		12 min.	Done in comparison routine, included in above time	300 seconds (for disks 1 & 2)			To remove duplicate subdocuments and rank: 8x450 (clock time)
Methods Used in Machine Searching							
- Vector Space Model		Yes - cosine distance measure	A modified vector space model [4]	Yes	Yes	Yes (Stage 1)	
- Probabalistic Model				No			Yes
- Cluster Searching				No			
- N-gram Matching				No			
- Boolean Matching	Yes			No		Yes (Stage 1)	
- Fuzzy Logic				No			Soft Boolean
- Free Text Scanning				No			
- Neural Networks				Yes			Yes
- Conceptual Graph Matching				No		Yes (Stage 2)	
- Other							

III. SEARCHING

NOTES:

[1] This is a batch routing system, with no pre-indexing. The system processes about 1,000 texts per minute.

[2] The querying program processed all 50 topics (either routing or adhoc) in parallel on four machines. Processing took approximately four hours.

[3] For adhoc, need to compare a query to ALL documents, so it takes about 10 minutes. For routing, we compare each new document to each of the 50 routing filters. This takes about 0.1 seconds.

[4] A modified vector space model. Fewer dimensions than typical vector space - e.g., 200 dimensional real valued term and doc vectors.

[5] 30 - 60 per query without soft-boolean (combine 2 methods).
60 - 200 per query with soft-boolean (combine 3 methods).

III. SEARCHING

SYSTEM NAME	CITY	ERIM	TMC	CITRI	UCLA	SEC	ETH	TRW1	TRW2
Total Computer Time to Search (cpu seconds)							[9]		
- Retrieval Time (cpu seconds)	Mean was 84 sec. [1]	Ranged from 12-24 hours for running all queries [2]	From less than 1 sec to 5 sec [4]	3-10 sec. depending upon selected parameters	Mean was about 84 sec [7]			Individual topics; 270-11,000 sec.; Average topic: 3300 sec.	~ 5 min FDF search time per topic; ~4.25 Hrs. for all 50 routing topics
- Ranking Time (cpu seconds)	Included in above	Typical times were from 1500 to 2000 cpu sec.	0.05 seconds total [5]	Negligible [6]	Included above			0 [8]	< 1 sec
Methods Used in Machine Searching									
- Vector Space Model			Yes	Cosine rule using tf idf weightings			Yes, See IB		
- Probabalistic Model	Yes				Yes	Yes			
- Cluster Searching									
- N-gram Matching		Yes [3]						Yes	Yes
- Boolean Matching								Yes	Yes
- Fuzzy Logic									
- Free Text Scanning								Yes	Yes
- Neural Networks									
- Conceptual Graph Matching									
- Other				Retrieval of pages of text rather than whole documents					

NOTES:

[1] Depends on number of terms, etc., etc. For the automatic adhoc run (cityau), the mean was 84 cpu seconds including ranking.

[2] Hard to give a meaningful number. The system was divided across 7 or 8 different SUN workstations of differing speeds. The total elapsed time for running all queries (say, for the adhoc task) ranged from 12 to 24 hours, depending on distribution of machine load.

[3] Our system uses n-gram matching of multiple query strings while scanning the original free text. It uses a weighting scheme on the different query strings based on their position in the original topic text.

[4] Less than a second for a small query (3 terms) and 5 seconds for a query with 80 terms.

[5] 0.05 seconds total, 0.03 for local ranking, 0.02 for global ranking (among the cm5 processors).

[6] Negligible -- heap used to extract top r accumulators, time is included above.

[7] Depends on the number of terms etc., etc. For the automatic adhoc run with automatic query expansion (uclaa2), the mean was 84 wall-clock sec including ranking.

[8] Hits come out tagged by query. Individual queries are already ranked, so hits emerge.

[9] In our approach, retrieval time and ranking time cannot be separated.

1. Experiment: topics 51-100 versus disk 3

	# of docs	#of feats	First rank avg. (median) [sec. CPU]	1000th rank avg. (median) [sec. CPU]
AP3	78'325	142	1.84 (1.12)	4.67 (4.58)
PATN3	6'708	181	0.21 (0.15)	0.80 (0.80)
SJM3	90'253	126	1.30 (1.27)	4.78 (4.60)
ZIFF3.1	100'000	26	1.04 (0.97)	1.78 (1.67)
ZIFF3.2	61'021	120	1.01 (0.88)	3.74 (3.75)

2. Experiment: topics 101-150 versus disks 1 and 2

	# of docs	#of feats	First rank avg. (median) [sec. CPU]	1000th rank avg. (median) [sec. CPU]
AP1	84'677	141	1.47 (1.50)	5.18 (5.02)
AP2	79'923	138	1.38 (1.37)	4.96 (4.92)
DOE1.1	100'000	42	1.42 (1.43)	2.93 (2.18)
DOE1.2	100'000	43	1.41 (1.47)	2.98 (2.15)
DOE1.3	26'087	43	0.41 (0.42)	0.92 (0.90)
FR1	26'207	166	0.68 (0.55)	2.22 (1.98)
FR2	20'108	170	0.57 (0.42)	1.77 (1.63)
WSJ1	98'627	121	1.64 (1.68)	5.96 (5.55)
WSJ2	74'520	115	1.17 (1.20)	3.57 (3.62)
ZIFF1	75'180	107	1.27 (1.23)	4.22 (3.80)
ZIFF2	56'920	103	0.90 (0.88)	2.90 (2.80)

III. SEARCHING

SYSTEM NAME	ADS	UIC	DALHOUSIE	MEAD	UIF	IDSR
Total Computer Time to Search (cpu seconds)	2 - 5 minutes [1]				3 - 5 seconds per document for routing.	
- Retrieval Time (cpu seconds)		After indexing, approximately 60 seconds	110 cpu seconds			18,000 total time (not cpu time) including decompression of all test data files.
- Ranking Time (cpu seconds)		1 second	No ranking, items are ordered by date (all items satisfy query).			600 total time (not cpu time)
Methods Used in Machine Searching						
- Vector Space Model						
- Probabalistic Model						Yes
- Cluster Searching						
- N-gram Matching						
- Boolean Matching						
- Fuzzy Logic						
- Free Text Scanning			Yes			
- Neural Networks					4	
- Conceptual Graph Matching						
- Other	Binary classification trees mapped into TOPIC query trees	Shortest paths among query words as they occur in documents (although only direct matches used in official results).				

III. SEARCHING

NOTES:

[1] We are not able to give separate answers for these - using the disk 3 data indexed under TOPIC, it takes between 2 - 5 minutes to perform a complete search for a single topic - it depends on the complexity of the topic.

III. SEARCHING (CON'T)

SYSTEM NAME	Dortmund	Cornell	Berkeley	Rutgers	Siemens	UMASS	VPI
Factors in Ranking				See INQUERY			
- Term Frequency	Yes	Yes	Yes		both docs and queries	Yes	Yes [9]
- Inverse Document Frequency	Yes	Yes	tried but discarded		query terms only	Yes	No
- Other Term Weights	Yes	Yes	Yes, see "Other"		Yes [3]	Yes (Query)	No
- Semantic Closeness			No		Yes [4]	No	No
- Position in Document			[1]			No	No
- Syntactic Clues			No			No	No
- Proximity of Terms	Yes	Yes	tried but discarded			Yes	No
- Information Theoretic Weights	Yes		No			Yes [7]	No
- Document Length	Yes	Yes	Yes, to calculate relative frequencies		Yes [5]	No [8]	indirectly [10]
- Completeness			No			No	No
- N-gram Frequency			No			No	No
- Word specificity			No		Yes [6]	No	No
- Word Sense Frequency			No			No	No
- Cluster Distance			No			No	No
- Other			Yes [2]				

NOTES:

[1] Stem occurrence frequencies in titles were doubled in some collections.

[2] Variables used were: optimally relativized query and document stem frequencies, global relative frequency of stem in all document texts, 2nd routing run: stem-specific statistics.

[3] Link type weight that reflects relative importance of different lexical relations; set to 0.5 for these experiments.

[4] Insofar as terms, closely related to selected synonym sets are added to query.

[5] Cosine normalization of weights in both documents and queries.

[6] Used subjectively when choosing synsets to add.

[7] Mutual Information Measure determines how phrases are evaluated, which indirectly affects the rank.

[8] We use maximum TF though, which seems to be correlated with document length.

[9] Modified per SMART ann ranking given above.

[10] Indirectly via use of maximum term frequency.

III. SEARCHING (CON'T)

SYSTEM NAME	GE	CLARIT	LSI	HNC	NYU	SYRACUSE	QUEENS
Factors in Ranking			Rank is determined by the cos (query, doc) in LSI-space				
- Term Frequency	Yes	TF_AUG (0.5 +0.5 * TF/MAX_TF)	Used in the term-document input to the SVD, and also in the query weight	Yes	Yes	Yes (Stage 2)	Yes
- Inverse Document Frequency	Yes	Yes	Used in the term-document input to the SVD, and also in the query weight	Yes	Yes	Yes [3]	
- Other Term Weights	Yes, relevance weights for routing	Yes [1]			Term similarities	Yes (Stage 1)	Within-item term frequency; inverse collection term frequency; total word occurrences
- Semantic Closeness			The cosine between a query and a document (in LSI-space) represents their derived similarity			Yes (Stage 2)	
- Position in Document							
- Syntactic Clues						Yes [4]	
- Proximity of Terms							
- Information Theoretic Weights							
- Document Length	Yes		Yes, as a part of the cosine measure			Yes [3]	Yes
- Completeness						Yes [5]	
- N-gram Frequency							
- Word specificity							
- Word Sense Frequency						Yes	
- Cluster Distance							
- Other		Yes [2]		Document vector distance to query vector		Semantic relation-ship between concepts (Stage 2)	

NOTES:

[1] An importance coefficient (with a value of 0.5, 1, 2, or 3) was assigned based on position of the term in the topic statement. Assignments were subject to correction in the case of manual processing.

[2] Similarity scores are calculated over sub-documents; the maximum score for any sub-document is assigned as the score for the whole document.

[3] Yes (Stage 1's subject field code module)
Yes (Stage 2)

[4] Yes (the existence and nature of relationship between concepts are important - Stage 2)

[5] Yes (Stage 1's proper noun, complex nominal and text structure boolean criteria matching module)
Yes (Stage 2).

III. SEARCHING (CON'T)

SYSTEM NAME	CITY	ERIM	TMC	CITRI	UCLA	SEC	ETH	TRW1	TRW2
Factors in Ranking									
- Term Frequency	Sometimes		Yes	Yes	Sometimes		Normalized feature frequency	Yes	
- Inverse Document Frequency	Yes		Yes	Yes	Yes	Yes	Normalized inverse documents frequency		
- Other Term Weights	From relevance information when available		Based on relevant (training) documents retrieved by a given term		From relevance information when available				Term weights are derived from statistical analysis of training database and sample relevant documents
- Semantic Closeness									
- Position in Document			Based inversely on the distance in paragraphs from the beginning of the document						
- Syntactic Clues								Lexical recognition of some antecedent-reference combinations	
- Proximity of Terms			Adjacent words are used for phrases					Yes	
- Information Theoretic Weights									
- Document Length	Sometimes			Yes	Sometimes		Euclidean length of document vector		
- Completeness									
- N-gram Frequency									
- Word specificity									
- Word Sense Frequency									
- Cluster Distance									
- Other		[1]							

III. SEARCHING (CON'T)

SYSTEM NAME	ADS	UIC	DALHOUSIE	MEAD	UIF	IDSR
Factors in Ranking						
- Term Frequency	No				4	Yes
- Inverse Document Frequency	No	Yes, within subset of documents containing at least one query word.			4	
- Other Term Weights	No					
- Semantic Closeness	No	Yes, semantic net distance is used.				
- Position in Document	No					
- Syntactic Clues	No					
- Proximity of Terms	No	Not a weight but a threshold of 3 word positions is used in initial processing.				Two adjacent terms
- Information Theoretic Weights	No					
- Document Length	No					
- Completeness	No					
- N-gram Frequency	No					
- Word specificity	No					
- Word Sense Frequency	No					
- Cluster Distance	No					
- Other	Statistical estimates of misclassification rate of the classifier.		Order in data file			Topic frequency in collection (i.e., topic prior probability).

IV. MACHINE SPECIFICATIONS

SYSTEM NAME	Dortmund	Cornell	Berkeley	Rutgers	Siemens	UMASS	VPI
Type of Machine Used	Sparc 2 Adhoc: Sparc 10	Sparc 2 [1]	Decstation 5000/125	SunSparc 10/51	Sparc 10/41	Sun Sparcserver 690	Decstation 5000/25 with ~7GB diskspace
Amount of RAM	64 MB; Adhoc: 160 MB	64 MB	48 MB	96 MB	128 MB	128 MB	40 MB
Clock Rate of CPU			25MHz	50MHz	40MHz	Unknown	25MHz

IV. MACHINE SPECIFICATIONS

NOTES:

[1] Except actual retrieval runs for routing were done on Sparc 1 with 12 MB RAM.

IV. MACHINE SPECIFICATIONS

SYSTEM NAME	GE	CLARIT	LSI	HNC	NYU	SYRCUSE	QUEENS
Type of Machine Used	Sparc 10	4 x DEC 3000/400 (ALPHA AXP) Running DEC OSF/1	Sparc 10	Sparc 10	Sparc 2	SUN workstations (several different kinds)	Sparc 10-30
Amount of RAM	64 MB	1@ 128 MB 1@ 64 MB 2@ 32 MB	One machine had 128 MB; another had 384 MB	512 MB	96 MB		128 MB
Clock Rate of CPU		133.33 MHz (however, effective performance with currently available compilers is approximately 2 times slower than the clock rate would suggest).		40 MHz	28.5 MIPS		

IV. MACHINE SPECIFICATIONS

SYSTEM NAME	CITY	ERIM	TMC	CITRI	UCLA	SEC	ETH	TRW
Type of Machine Used	SPARC mainly	2 Sparcstations & 6 Sparcstation 2's	A 64 node CM5	Sun Model 512 (one processor only)	SunSPARC 2	PC compatible 486/25	SPARC MP 690 Model 41	FDF-3 parallel search engine
Amount of RAM	40 MB some of the time	Each machine had at least 32 MB	32 MB/node or 2GB for the configuration	160 MB	32 MB	12 MB	128 MB	N/A
Clock Rate of CPU	Don't know	33 MHz for the Sparc 2 (I think)	40 MHz (Sparc chip)	50 MIP	Don't know	25 MHz	40 MHz	FDF-3 used for TREC searched at 3-3.5 MB/sec.

IV. MACHINE SPECIFICATIONS

SYSTEM NAME	ADS	UIC	DALHOUSIE	MEAD	UIF	IDSR
Type of Machine Used	Tree building primarily done on a Sparc 1, routing results generated using a Sparc 10.	IBM 3090-300J for initial indexing, then RS-6000 for weighting.	Sparcstation 10		SparcServer 690 MP	MacIntosh Centris 650
Amount of RAM	16 MB Sparc 1 32 MB Sparc 10	IBM 3090 running CMS allows a maximum of 16 MB of CPU as a virtual machine. RS-6000 had 128 MB of RAM	128 MB		128 MB	8 MB
Clock Rate of CPU	Unknown - but in any case, most of the computing was done with networked data servers so clock rates don't tell us anything.	IBM 3090: 69 MHz RS-6000: 66 MHz	40 MHz		4 Cypress CY 6050 processors	25 MHz

V. SYSTEM COMPARISON

SYSTEM NAME	Dortmund	Cornell	Berkeley	Rutgers	Siemens	UMASS	VPI
How Much "Software Engineering" went into the development?	Several years	Several years	None except for the probabilistic logic. The Berkeley system is an experimental prototype only, programmed as a minimal modification of the SMART system.	For the data fusion part, approximately 60 hours, for the query combination parts, approximately 150 hours.	"Our" system is essentially SMART; SMART has been well-engineered with a primary goal of flexibility, not raw speed. Modifications made by the SMART group at Cornell for last years TREC were used in these runs.	INQUERY is a research system. About 10 person years went into its development prior to these experiments.	Basic system was 1985 version of SMART with many enhancements (i.e., pnorm query processing) added from previous projects before TREC. SE during TREC consisted of creations of outside programs for merging combining the results from individual SMART retrieval runs and adding support for multiple query and index processing during a single retrieval run.
Given appropriate resources, could your System be made To Run Faster? By How Much?	If the feature vectors for the query terms were stored in a cache, query regression would take 20-30% less time.	Of course	Yes, see discussion in SMART's documentation: SMART is "not strongly optimized for any one particular use." The Berkeley system has roughly the same efficiency characteristics as SMART.	For the data fusion part, by a factor of 8; for the other parts, unknown.		Yes, at least a factor of 2.	Use of inverted files (given enough disk space) would have significantly increased the retrieval time. The 'multiple retrieval' code added to SMART was restricted by the existing SMART code and could have been made faster if implemented as a separate retrieval system instead of being fitted into existing code.
What Features are missing that would benefit your system?		No feature recognition (eg., company names, geographical locations, dates, amounts of money.	Might benefit from a conflator, thesaurus, disambiguator, and the use of many other clue types.	For the data fusion part, a lookup procedure to convert raw scores to ranks, based on the training set. This is necessary for true routing as opposed to batch scoring of "routing" queries.		Word finder. (An on-line concept association database).	Phrase identification and matching; proper noun identification.

V. SYSTEM COMPARISON

SYSTEM NAME	GE	CLARIT	LSI	HNC	NYU	SYRACUSE	QUEENS
How Much "Software Engineering" went into the development?	Zero	The system used for TREC-2 processing was developed as a university-research prototype. It is engineered for robustness and flexibility, rather than speed. Most of the components of the system are less than two years old. The research-prototype code (essentially all C) is not production-quality.	The LSI system was built as a research prototype to look at human interface issues, and designed to work on much smaller databases. I'd guess that about 1-2 person years were spent on various aspects of the system.	Several years	Quite a lot of code rewriting was done to adjust NIST system to handle the large index (8 times larger than without compound terms).	Not much. It was the first prototype testing with the current functionalities	Quite a lot of reprogramming to truncate record sizes, remove intermediate files.
Given appropriate resources, could your System be made To Run Faster? By How Much?	For routing, it could be a bit faster. For retrieval, it is compatible with any inverted indexing strategy.	We anticipate at least an order of magnitude speed improvement in the system within the next six months. This will be possible due to (1) re-engineering of the system and (2) the use of optimization utilities sold for the DEC ALPHA platform. (The current OSF compiler does not optimize code appropriately for the ALPHA (64 bit) architecture, with disappointing results).	[1]	Yes, 20%-40% searching entire database. Many orders of magnitude faster with document clustering (had an order of 15 speed-up on a foreign corpus.)	Base IR system is much better than it was during TREC-1. However, second phase of index building is still slow and fragile.	Yes, with careful design of the data structures and elimination of the features added for experimental purposes, the speed can be improved significantly, at least by an order of magnitude.	Yes. Some of code was translated from Pascal to C and used as is. More RAM can support larger networks in core and speedup.
What Features are missing that would benefit your system?	This approach is very simple and has no fancy features. Better tokenization, special purpose query handling, proximity, and negation, for example, could help a lot, as would better ranking.	The CLARIT TREC-2 system did not take advantage of several processing options that may have given improved results, including tokenization, sub-lexicon discovery over training sets, and EQ-class discovery for thesaurus terms.	Better tokenization, including proper noun identification, phrases, and perhaps some better treatment of "nots." Precision enhancing methods would also help some.	Word sense disambiguation (already in early development). Document cluster (speed up retrievals).	- There is still a lot of room for improvement of NLP programs. - A feedback mechanism would be helpful. - Faster indexing	[2]	Sense disambiguation, better phrases, phrase-phrase matching

V. SYSTEM COMPARISON

[1]

(a) I'd guess that simple database and matching operations could be speeded up by a factor of 2-3 with a rewrite of the software to do what we now do (rather than what we thought we might want when we started).

(b) Most of the initial analysis time is spent in computing the SVD decomposition of the term-document matrix. The sparse-iterative algorithm we now use is two orders of magnitude faster than the dense algorithm we used 2 years ago. We might find additional improvements of 2-3 times by using more memory, single-precision arithmetic. Parallel algorithms might help, but again, only by a factor of 2. This analysis is a one-time cost for relatively stable domains.

(c) Query processing is slow. Although the LSI vectors have many fewer dimensions than standard vector representations, the vectors are dense - every term is related to every document; it's just a matter of how much. Thus, we cannot take advantage of efficient inverted indices or other structures. It is, however, trivial to match queries to documents in parallel. Improvements here are limited only by the number of processors we have! We are also exploring some heuristic methods for finding near neighbors in high-dimensional spaces.

[2]

There are many features that have not been included in the system because it was the very first prototype. As in the paper, many improvements are under way, especially in reducing errors in text processing, reducing complexity of representation, improving quality of knowledge bases, and improving the time and space complexity with redesign of the data structures and implementations.

V. SYSTEM COMPARISON

SYSTEM NAME	CITY	ERIM	TMC	CITRI	UCLA	SEC	ETH	TRW
How Much "Software Engineering" went into the development?	Very little TREC-specific in the search system, although TREC has spurred the development of some additional features. It is a generalized bibliographic retrieval system which has undergone continual modification to meet the requirements of a number of research projects since 1983, involving many thousands of person-hours.	Less than a week	Our system is a research prototype, it took two person months to build.	A reasonable amount. We have been interested in algorithmic aspects-- speed, and memory and disk requirements .	Same as CITY		250 - 300 hours.	- About 1 week to develop tools for TRW1 test runs - About 1 month to develop tools for TRW2 test runs - Underlying FDF system has been developed over 7+ years at TRW and PARACEL
Given appropriate resources, could your System be made To Run Faster? By How Much?	Disk I/O is the most serious bottleneck in searching and in outputting documents. Keeping entire database in core would speed real time by > order of magnitude, CPU by much less. Perhaps 3 GB of core is too much to expect yet. More practically, faster disks and faster bus. Indexing on the other hand is CPU bound most of the time. This could be distributed over N processors. Time would behave approx. like A + B/N + CN for a given database, where typically B > A, and B >> C.	By converting it to a true retrieval system, I would guess that we could improve the system speed by at least 20-fold	Yes, it scales linearly with the size of the CM5, but even on the current size, the retrieval time could be speeded up by a factor of 2 or 3. Memory optimization is the main limiting factor for now.	30%??	Same as CITY	A lot		Sure. We expect substantial performance increases for a single FD-3 unit (3-10X) from a variety of hardware and firmware improvements. Future releases of FDF software will include features to automatically harness multiple FD-3 units in parallel running the same queries. Performance should improve linearally with the number of FDF's used. Incorporating ideas from these TREC experiments into the automatic query generation tool will reduce query size, thus improve system performance. Such tools are under development.
What Features are missing that would benefit your system?	[1]	There is an itemized list in my paper.	Normalization (document length, cosine normalization, etc.), more elaborate use of the proximity information and clustering of terms, query expansion, stemming. Use of relevance feedback.	System has a relatively basic user interface. No mechanism for feedback or other query modification.	Same as CITY	The major feature that is designed but not yet implemented is the use of feedback.		[2]

NOTES:

[1] It must need something, because other systems do better! We've not done much with phrases; it seems likely that extensive use of phrases would help. The system can handle them already. What we need is a good method of phrase discovery. We were attracted by the idea of treating paragraphs as documents this time, but didn't have time to do this. Needs more elaborate database model.

[2] We are more directly incorporating term weighting into our system. Better query construction, evaluation and refinement tools are under development. We are also working to incorporate several ideas developed in this exercise into automatic query generation and refinement tools. Incorporation of these ideas should improve the recall-precision performance of our system. Some specific improvements include:

- Better integration of term weights.
- Better tools for initial query construction
- Better stemming and stop-word elimination
- Evaluation of search term independence
- Better document similarity metrics.

V. SYSTEM COMPARISON

SYSTEM NAME	ADS	UIC	DALHOUSIE	MEAD	UIF	IDSR
How Much "Software Engineering" went into the development?	Approximately 4 person weeks to "clean-up" last years TREC-1 experimental code - both the CART algorithm and the tool we used to convert CART trees into TOPIC trees were "off-the- shelf."	TREC-2 upgrades involved approximately one person-month.	Strictly research prototype. Retrofit of adhoc interactive system to process filters. Upsize to work on the large item sets generated from the large data files.		Approximately 280 hours of programming have been used to develop the neural network. The system is implemented in C and uses a scanner for text processing. The scanner is borrowed from the QA system built for NASA.	4 person months, to develop entire system.
Given appropriate resources, could your System be made To Run Faster? By How Much?	Undoubtedly - if we had started out intending to use TOPIC as the actual test environment, we would have designed a system that made use of TOPIC's data preparation utilities - giving us an order of magnitude speed in tree building.	With parallel processing, an order of magnitude increase in speed would be expected. Without parallel processing, improvements on the order of 100% would be expected from optimizations on current software. Restructuring data representation probably results in an order of magnitude increase in a serial processing mode.	Yes, 100-200% faster. Being a prototype optimization of searching for multiple terms was not implemented and lots of messages, including one per record, are still displayed on the screen as the system churns away.		Yes. Our system can easily run in parallel. The processing time can be approximately reduced by the following ratio [EQN "center [#R [time required using one CPU] over [#R [total number of CPUs]]]].	With sufficient disk storage, we could decompress the data files before the test runs, saving several hours per test run.
What Features are missing that would benefit your system?	We still have not experimented with external resources such as part-of-speech taggers and lexicons that might be used to both expand the feature set and the complexity of the CART trees; nor have we experimented with using low-level topics as features - all of these could be expected to give improved results.	Shortest path algorithm needs to be implemented. For TREC-2, only direct pair matches were involved. Identifying indirect paths is under development. Tests with several topics after official results were submitted showing that use of indirect paths results in improvements to capture of relevant documents at 93%. Weighting, however, needs further enhancements because improvements are at 45% for the top 1000 document cutoff.	Functions to screen for primatives other than simple strings, such as dates and names. Automatic analysis of contents of the data files to assist the user in recognizing patterns that 'may' be useful when searching that particular file.		The following software improvements would benefit the retrieval performance: 1. Adding inverted index and term frequency information for term weighting. 2. Using larger and more accurate Semantic Lexicon. 3. Using training text to improve the neural network performance.	- Better modelling of relationships between features. - Better feature extraction routines (synonyms, phrases) - A decision-theoretic perspective.

•U.S. GOVERNMENT PRINTING OFFICE: 1994-300-568/03020

ANNOUNCEMENT OF NEW PUBLICATIONS ON COMPUTER SYSTEMS TECHNOLOGY

Superintendent of Documents
Government Printing Office
Washington, DC 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Institute of Standards and Technology Special Publication 500-.

Name _____

Company _____

Address _____

City _____ State _____ Zip Code _____

(Notification key N-503)

NIST Technical Publications

Periodical

Journal of Research of the National Institute of Standards and Technology—Reports NIST research and development in those disciplines of the physical and engineering sciences in which the Institute is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Institute's technical and scientific programs. Issued six times a year.

Nonperiodicals

Monographs—Major contributions to the technical literature on various subjects related to the Institute's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NIST, NIST annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NIST under the authority of the National Standard Data Act (Public Law 90-396). NOTE: The Journal of Physical and Chemical Reference Data (JPCRD) is published bimonthly for NIST by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements are available from ACS, 1155 Sixteenth St., NW, Washington, DC 20056.

Building Science Series—Disseminates technical information developed at the Institute on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NIST under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NIST administers this program in support of the efforts of private-sector standardizing organizations.

Consumer Information Series—Practical information, based on NIST research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

Order the **above** *NIST publications from: Superintendent of Documents, Government Printing Office, Washington, DC 20402.*

Order the **following** *NIST publications—FIPS and NISTIRs—from the National Technical Information Service, Springfield, VA 22161.*

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NIST pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NIST Interagency Reports (NISTIR)—A special series of interim or final reports on work performed by NIST for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Service, Springfield, VA 22161, in paper copy or microfiche form.

U.S. Department of Commerce
National Institute of Standards and Technology
Gaithersburg, MD 20899