NIST Special Publication 500–225

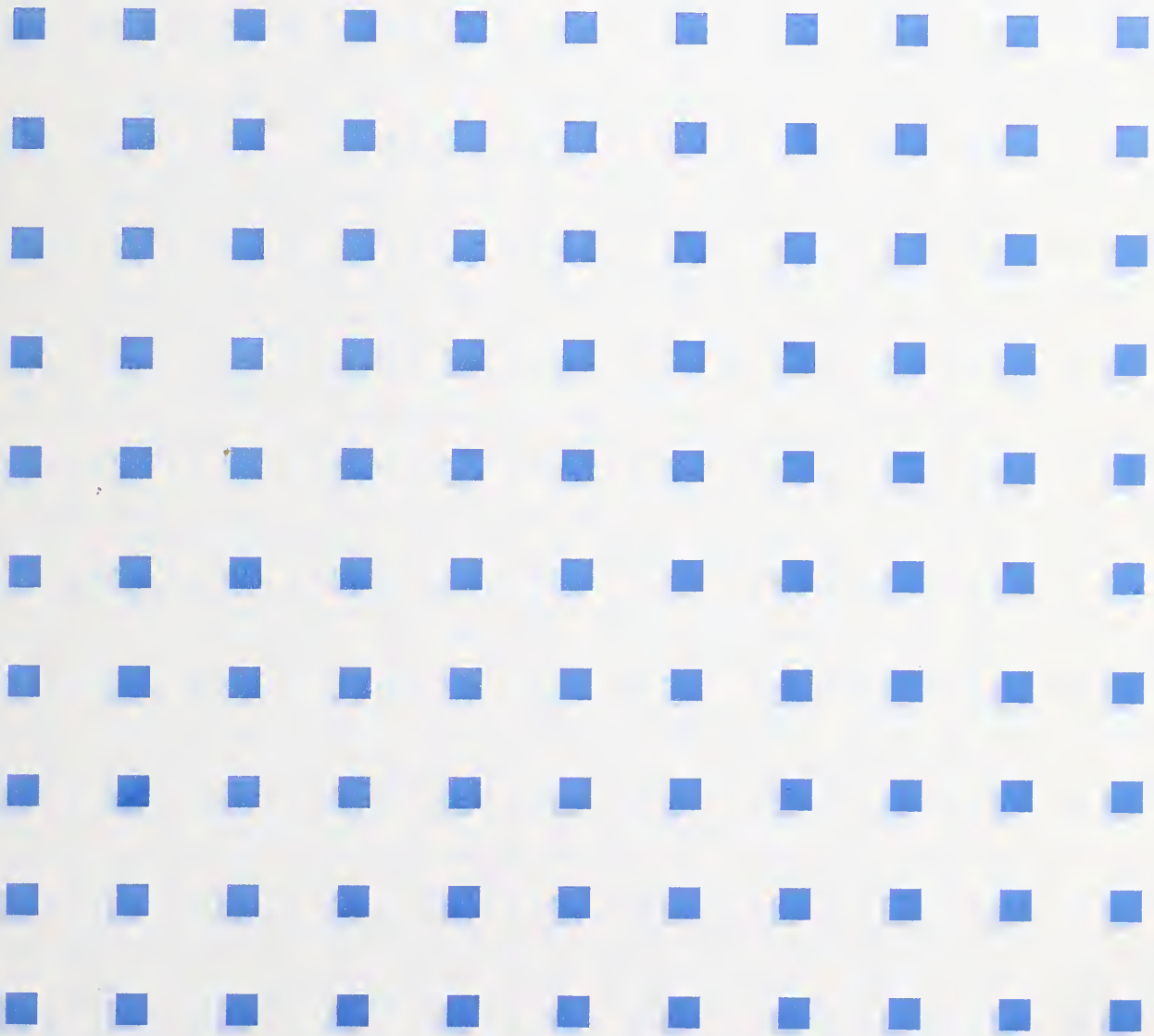
Computer Systems Technology

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

NIST

Overview of the Third Text REtrieval Conference (TREC–3)

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 develops measurement techniques, test methods, standards, and 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.

Office of the Director
- Advanced Technology Program
- Quality Programs
- International and Academic Affairs

Technology Services
- Manufacturing Extension Partnership
- Standards Services
- Technology Commercialization
- Measurement Services
- Technology Evaluation and Assessment
- Information Services

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

Chemical Science and Technology Laboratory
- Biotechnology
- Chemical Kinetics and Thermodynamics
- Analytical Chemical Research
- Process Measurements[2]
- 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
- Intelligent Systems
- Manufacturing Systems Integration
- Fabrication Technology

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

Building and Fire Research Laboratory
- Structures
- Building Materials
- Building Environment
- Fire Safety
- Fire Science

Computer Systems Laboratory
- Office of Enterprise Integration
- 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
- Computer Systems and Communications[2]
- Information Systems

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

NIST Special Publication 500–225

Overview of the Third Text REtrieval Conference (TREC–3)

D. K. Harman, Editor

Computer Systems Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-0001

April 1995

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 500-225
Natl. Inst. Stand. Technol. Spec. Publ. 500-225, 575 pages (April 1995)
CODEN: NSPUE2

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1995

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

Foreword

This report constitutes the proceedings of the third Text REtrieval Conference (TREC-3) held in Gaithersburg, Maryland, November 2-4, 1994. 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 32 participating groups. The conference was the third in an on-going series of workshops to evaluate new technologies in text retrieval.

The workshop included plenary sessions, discussion groups and demonstrations. 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
March 24, 1995

TREC-3 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
Matt Mettler, TRW
John Prange, U.S. Department of Defense
Steve Robertson, City University, UK
Alan Smeaton, Dublin City University, Ireland
Karen Sparck Jones, Cambridge University, UK
Richard Tong, Verity, Inc.

TABLE OF CONTENTS

Abstract

This report constitutes the proceedings of the third Text REtrieval Conference (TREC-3) held in Gaithersburg, Maryland, November 2-4, 1994. 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 33 participating groups.

The goal of the conference was to bring research groups together to discuss their work on a large test collection. There was a wide variation of retrieval techniques reported on, including methods using automatic thesauri, 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), tables of the system results, and brief system descriptions including timing and storage information.

Overview of the Third Text REtrieval Conference (TREC-3)

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 conference became the first in a series of ongoing conferences dedicated to encouraging research in retrieval from large-scale test collections, and to encouraging increased interaction among research groups in industry and academia. From the beginning there has been an almost equal number of universities and companies participating, with an emphasis on exploring many different types of approaches to the text retrieval problem.

The research done by the participating groups in the three TREC conferences has varied, but has followed a general pattern. TREC-1 required significant system rebuilding by most groups due to the huge increase in the size of the document collection (from a traditional test collection of several megabytes in size to the 2 gigabyte TIPSTER collection). The TREC-1 results should therefore be viewed as very preliminary due to severe time constraints. The second TREC conference (TREC-2) occurred in August of 1993, less than 10 months after the first conference [Harman 1994a]. 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. In some senses, however, the TREC-2 results should be viewed as a baseline for more complex experimentation.

The TREC-3 results reflect some of that more complex experimentation. For some groups that meant more extensive experiments based on their basic system techniques. For other groups it involved trying techniques from other groups and exploring more hybrid approaches. Some groups tried approaches that were radically different from their original approaches. As should be expected, those groups new to TREC had the same scaling problems as seen in TREC-1.

This paper provides an overview of the TREC-3 conference, including a review of the TREC task, a very 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 Task and the Participants

The three TREC conferences have all centered around two tasks based on traditional information retrieval modes: a "routing" task and an "adhoc" task. In the routing task it is assumed that the same questions are always being asked, but that new data is being searched. This task is similar to that done by news clipping services or by library profiling systems. In the adhoc task it is assumed that new questions are being asked against a static set of data. This task is similar to how a researcher might use a library, where the collection is known, but it is unknown what questions are likely to be asked.

A schematic of those tasks is shown in Figure 1.

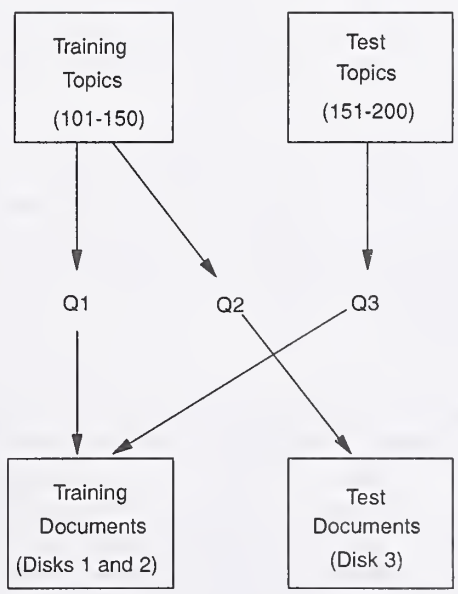


Figure 1. The TREC Task.

Table 1: TREC-3 Participants (14 companies, 19 universities)

Australian National University	Bellcore
Carnegie Mellon University/CLARITECH	CITRI, Australia
City University, London	Cornell University
Dublin City University	Environment Research Institute of Michigan
Fulcrum	George Mason University
Logicon Operating Systems	Mayo Clinic/Foundation
Mead Data Central	National Security Agency
New York University	NEC Corporation
Queens College	Rutgers University (two groups)
Siemens Corporate Research Inc.	Swiss Federal Institute of Technology (ETH)
TRW/Paracel	Universitaet Dortmund, Germany
University of California - Berkeley	University of Central Florida
University of Massachusetts at Amherst	VPI&SU (Virginia Tech)
University of Minnesota	University of Toronto
Universite de Neuchatel, Switzerland	Verity Inc.
West Publishing Co.	Xerox Palo Alto Research Center

In TREC the routing task is represented by using known topics and known relevant documents for those topics, but new data for testing. This is shown on the left side of Figure 1. The participants are given a set of known (or training) topics, shown in the top left-hand box, along with a set of known relevant documents (relevance judgments) for those topics. The topics consist of natural language text describing a user's information need (see section 3.3 for more description of the topics). These topics are used to create a set of queries (the actual input to the retrieval system) which are then used against the training documents. This is represented by Q1 in the diagram. Many sets of Q1 queries might be built to help adjust systems to this task, to create better weighting algorithms, and in general to train the system for testing. The results of this research are used to create Q2, the final routing queries to be used against the test documents.

The adhoc task is represented by using known documents, but new topics with no known relevant documents. This is shown on the right-hand side of Figure 1, where the 50 new test topics are used to create Q3 as the adhoc queries for searching against the training documents. The results from searches using Q2 and Q3 are the official test results sent to NIST.

In addition to clearly defining the tasks, other guidelines are used in TREC. These 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 are constructed to reflect an actual operational environment, and to allow as fair as possible separation among the diverse query construction approaches. Three generic categories of query construction were defined in TREC-3, based on the amount and kind of manual intervention used.

1. AUTOMATIC (completely automatic query construction)
2. MANUAL (manual query construction)
3. INTERACTIVE (use of interactive techniques to construct the queries)

There were 33 groups participating in TREC-3 (see Table 1), using a wide variety of retrieval techniques. One of the participants (Fulcrum) withdrew their results before the conference and therefore no results from this company appear in the proceedings. 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). Each participating 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 INTERACTIVE), 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.

TREC-3 introduced a second language (Spanish) to the task, with four groups working with a small Spanish collection in addition to their work in English. This collection, and the results, are discussed in section 5.4.

3. The Test Collection (English)

3.1 Introduction

Like most traditional retrieval collections, there are three

Table 2: 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)	270	259	245	262	186
(disk 2)	247	241	178	211	
(disk 3)	290	242	349	245	
Number of records					
(disk 1)	98,732	84,678	75,180	25,960	226,087
(disk 2)	74,520	79,919	56,920	19,860	
(disk 3)	90,257	78,321	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	

distinct parts to this collection -- the documents, the questions or topics, and the relevance judgments or "right answers." These test collection components are discussed very briefly in the rest of this section. For a more complete description of the collection, see [Harman 1994b].

3.2 The Documents

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
- FR -- *Federal Register* (1988)

Disk 3

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

Table 2 shows some basic document collection statistics. Note that although the collection sizes are roughly equivalent in megabytes, there is a range of document lengths across collections, 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, ZIFF and especially FR documents have much wider range of lengths within their collections.

The documents are uniformly formatted into SGML, with a DTD included for each collection to allow easy parsing.

<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

.
.
</TEXT>
</DOC>

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 new topics used in TREC-3 reflect a slight change in this decision. The topics in TREC-1 and 2 (topics 1-150) were not only very long, but contained complex structures. These topics were designed to mimic a real user's need, and were written by people who are actual users of a retrieval system. However they were intended to represent long-standing information needs for which a user might be willing to create elaborate topics, and therefore are more suited to the routing task than to the adhoc task, where users are likely to ask much shorter questions.

The new topics used in TREC-3 (topics 151-200) are not only much shorter, but missing the complex structure of the earlier topics. In particular the concepts field has been removed. This field contained a mini-knowledge base about a topic such as a real searcher might possess. The field was removed because it was felt that real adhoc questions would not contain this field, and because inclusion of the field discouraged research into techniques for expansion of "too short" user need expressions. Note that the shorter topics do not create a problem for the routing task, as experience in TREC-1 and 2 has shown that the use of the training documents allows a shorter topic (or no topic at all).

In addition to being shorter, the new topics were written by the same group of users that did the assessments. Specifically, each of the new topics (numbers 151-200) were developed from a genuine need for information brought in by the assessors. Each assessor constructed his/her own topics from some initial statements of interest, and performed all the relevance assessments on these topics (with a few exceptions).

The following is one of the new topics used in TREC-3. Each topic is formatted in the same standard method to allow easier automatic construction of queries.

<num> Number: 168
<title> Topic: Financing AMTRAK

<desc> Description:
A document will address the role of the Federal Government in financing the operation of the National Railroad Transportation Corporation (AMTRAK).

<narr> Narrative: A relevant document must provide information on the government's responsibility to make AMTRAK an economically viable entity. It could also discuss the privatization of AMTRAK as an alternative to continuing government subsidies. Documents comparing government subsidies given to air and bus transportation with those provided to AMTRAK would also be relevant.

</top>

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. All three TRECs have used the pooling method [Sparck Jones & van Rijsbergen 1975] to assemble the relevance assessments. In this method a pool of possible relevant documents is created by taking a sample of documents selected by the various participating systems. This sample is then shown to the human assessors. The particular sampling method used in TREC is to take the top X documents retrieved by each system for a given topic and merge them into the pool for assessment. This is a valid sampling technique since all the systems used ranked retrieval methods, with those documents most likely to be relevant returned first.

Evaluation of retrieval results using the assessments from this sampling method is based on the assumption that the vast majority of relevant documents have been found and that documents that have not been judged can be assumed to be not relevant. A test of this assumption was made between TREC-2 and TREC-3, using TREC-2 results. Thirty-six (18 adhoc and 18 routing) topics were selected for additional relevance assessments, using a pseudo-random selection based only on the number of original relevant documents and on selecting equal numbers of topics from each assessor. For each selected topic, a new pool of documents was created by taking the top 200 documents from seven different runs known to achieve good results and to have little overlap in their document selection. New judgments were made on this pool, using the same judges that made the original decisions for each topic.

Table 3 gives the results of this test. On average, 30 new relevant documents (16%) were found for each of the top-

Table 3: Analysis of Completeness of Relevance Judgments (TREC-2)

Percent New Rel.	No. of Topics	Average New Rel.	Average Total Rel.	Average No. Jud.	Average "Hardness"
0%	5	0	46	381	0.3477
1-9%	11	10	173	257	0.4190
10-19%	9	36	277	343	0.2610
20-29%	6	47	185	190	0.3660
30-33%	5	73	242	233	0.5212
Average (over all 36 topics)		30	193	282	
Median		21	190	220	
Average (over the 18 routing topics)		18	188	373	
Median		8	160	376	
Average (over the 18 adhoc topics)		42	197	190	
Median		28	209	150	

Table 4: Overlap of Submitted Results

	Adhoc			Routing		
	Possible	Actual	Relevant	Possible	Actual	Relevant
TREC-1	3300	1279 (39%)	277 (22%)	2200	1067 (49%)	371 (35%)
TREC-2	4000	1106 (28%)	210 (19%)	4000	1466 (37%)	210 (14%)
TREC-3						
at 100	4800	1005 (21%)	146 (15%)	4900	703 (14%)	146 (21%)
at 200	9600	1946 (20%)	196 (10%)	9800	1333 (14%)	187 (14%)

ics, with a median of only 21 (11%) new relevant documents per topic. The median is much lower than the average because of the relatively large number of new documents found for those five topics with over 30% additional relevant documents found.

Table 3 also shows that there is some correlation between the number of new relevant documents found and the original number of relevant documents, particularly in that topics with few relevant documents initially tended to have few new ones found. In contrast, there is no correlation between the number of new relevant documents found and the number of new judgments made, or between the number of new relevant found for a topic and the "hardness" of that topic (a measure of average system performance for that topic). More new relevant documents were found for the adhoc task than for the routing task. This may reflect more "available" relevant documents for the adhoc task (twice the amount of searchable text) or may be caused by the more complete and accurate queries used in routing task due to the training data.

A different measure of the effect of pooling can be seen by examining the overlap of retrieved documents. Table 4 shows the statistics from the merging operations in the three TREC conferences. For TREC-1 and TREC-2 the top 100 documents from each run (33 runs in TREC-1 and 40 runs in TREC-2) could have produced a total of 3300 and 4000 documents to be judged (for the adhoc task). The average number of documents actually judged per topic (those that were unique) was 1279 (39%) for TREC-1 and 1106 (28%) for TREC-2. Note that even though the number of runs has increased by more than 20% (adhoc), the number of unique documents found has actually dropped. The percentage of relevant documents found, however, has not changed much. The more accurate results going from TREC-1 to TREC-2 mean that fewer "noisy" nonrelevant documents are being found by the systems. This trend continued in TREC-3, even though the pooling method was changed.

Because of expected constraints in assessor time, only one run from each TREC-3 group was judged, with the groups specifying which run. However, due to the increase in overlap (as shown in Table 4), and more efficient judging, extra time became available and the decision was made to judge the top 200 documents for those runs. Table 4 gives the results of the TREC-3 mergings at both 100 documents and 200 documents. The percentage of unique documents found continues to drop compared with TREC-2, with a major drop for the routing. The total number of relevant documents found in TREC-1, TREC-2, and TREC-3 has dropped only somewhat, however, and that drop has been caused by a deliberate tightening of the topics between TREC-1 and TREC-2. Table 4 also shows

Table 5: Analysis of Pooling Methodologies (Adhoc)

TREC-2 -- Relevant Documents Found in "Second" Run			
Percent New Rel.	No. of Topics	Average New Rel.	Average No. Rel.
0%	0	-	-
1-9%	6	9	123
10-19%	19	26	163
20-29%	19	68	274
30-36%	5	109	296
Average		48	210
Median		30	201
TREC-3 -- Relevant Documents Found above 100			
Percent New Rel.	No. of Topics	Average New Rel.	Average No. Rel.
0%	1	0	85
1-9%	12	3	65
10-19%	7	13	96
20-29%	22	59	237
30-36%	8	137	381
Average		50	196
Median		30	122

Table 6: Analysis of Pooling Methodologies (Routing)

TREC-2 -- Relevant Documents Found in "Second" Run			
Percent New Rel.	No. of Topics	Average New Rel.	Average No. Rel.
0%	4	0	6
1-9%	8	4	61
10-19%	21	33	220
20-29%	11	88	345
30-36%	6	84	259
Average		44	210
Median		33	163
TREC-3 -- Relevant Documents Found above 100			
Percent New Rel.	No. of Topics	Average New Rel.	Average No. Rel.
0%	7	0	24
1-9%	9	6	106
10-19%	16	19	129
20-29%	16	94	354
30-36%	2	91	249
Average		41	187
Median		13	123

the drop in relevant documents found beyond the 100 document cutoff. This not only reflects the ranking done by the systems, but shows the diminishing numbers of relevant documents to be found even as the judged pool continues to grow.

The use of a different pooling method in TREC-3 provided a chance to compare the two methods. Tables 5 and 6 show this comparison. The first method (that used in TREC-2) took the top 100 documents from two runs, whereas the second method (that used in TREC-3) took the top 200 documents from a single run. The "base" for both methods is the top 100 documents in the single or "first" run. The additional documents to be compared are the number of relevant documents in the top 100 for the "second" run (TREC-2) versus the number of relevant documents in the second 100 in the single run for TREC-3.

Table 5 shows that both pooling methods worked equally well for the adhoc task. About the same numbers of relevant documents were found by each method, with similar averages, medians, and distributions of "new" relevant documents across the topics. This verifies the TREC-2 completeness experiments shown in Table 3, in that the average and median number of "new" documents found beyond the 100 document cutoff is similar to those found in TREC-3.

For the routing task, however, Table 6 shows that the first pooling method (TREC-2) seems to have found more relevant documents (higher median). Whereas this could reflect something about the different topics used in TREC-2 and TREC-3, it is more likely a reflection of the difference between system performance in the adhoc and routing tasks. Routing runs are generally more accurate in finding documents and more effective in ranking them, due to the availability of training data. Therefore the second 100 documents are less likely to contain additional relevant documents for the routing task than for the adhoc task. Again this verifies the completeness experiments shown in Table 3, which show far fewer new relevant documents being found for the routing task after the 100 document cutoff.

This analysis suggests a return to the TREC-2 pooling methodology, and that is what is planned for TREC-4. Participating groups would also prefer judgments on both official runs as this allows more exactness in evaluating run variations.

After pooling, each topic was judged by a single assessor to insure the best consistency of judgment. Some testing of this consistency was done after TREC-2, and showed an average agreement between two judges of about 80%. More consistency testing will be done in the future.

4. Evaluation

An important element of TREC is to provide a common evaluation forum. Standard recall/precision and recall/fallout figures have been calculated for each TREC system and are shown in Appendix A, along with some single evaluation measures for each system. A detailed explanation of the measures is also included in the appendix. New for TREC-3 is a histogram for each system showing performance on each topic. In general more emphasis is being placed on a per topic analysis this year in an effort to get beyond the averages. (Although work has been done to find statistical differences between the averages, see paper "A Statistical Analysis of the TREC-3 Data" by Jean Tague-Sutcliffe and James Blustein.)

Additional data about each system was collected that describes system features and system timing, and allows some primitive comparison of the amount of effort needed to produce the results. The individual system descriptions are given in Appendix B.

5. Results

5.1 Introduction

One of the important goals of the TREC conferences is that the participating groups freely devise their own experiments within the TREC task. For some groups this means doing the routing and/or adhoc task with the goal of achieving high retrieval effectiveness performance. For other groups, however, the goals are more diverse and may mean experiments in efficiency, unusual ways of using the data, or experiments in how "users" would view the TREC paradigm.

The overview of the results discusses the effectiveness of the systems and analyzes some of the similarities and differences in the approaches that were taken. Additionally it points to some of the other experiments run in TREC-3 where results cannot be measured completely using recall/precision measures.

In all cases, readers are referred to the system papers in this proceedings for more details.

5.2 Adhoc Results

The adhoc evaluation used the new topics (topics 151-200) against the two disks of training documents (disks 1 and 2). A dominant feature of the adhoc task in TREC-3 was the removal of the concepts field in the topics (see more on this in the discussion of the topics, section 3.3) Many of the participating groups designed their experiments around techniques to expand the shorter and less "rich" topics.

There were 48 sets of results for adhoc evaluation in TREC-3, with 42 of them based on runs for the full data set. Of these, 28 used automatic construction of queries, 12 used manual construction, and 2 used interactive construction.

Figure 2 shows the recall/precision curves for the 6 TREC-3 groups with the highest non-interpolated average precision using automatic construction of queries. The runs are ranked by the average precision and only one run is shown per group (both official Cornell runs would have qualified for this set).

A short summary of the techniques used in these runs shows the breadth of the approaches. For more details on the various runs and procedures, please see the appropriate papers in this proceedings.

cityal -- City University, London (see paper "Okapi at TREC-3" by S.E. Robertson, S. Walker, S. Jones, M.M. Hancock-Beaulieu and M. Gatford) used a probabilistic term weighting scheme similar to that used in TREC-2, but expanded the topics by up to 40 terms (average around 20) automatically selected from the top 30 documents retrieved. They also used dynamic passage retrieval in addition to the whole document retrieval in their final ranking.

INQ101 -- University of Massachusetts at Amherst (see paper "Document Retrieval and Routing Using the INQUERY System" by John Broglio, James P. Callan, W. Bruce Croft and Daniel W. Nachbar) used a version of probabilistic weighting that allows easy combining of evidence (an inference net). Their basic term weighting formula (and query processing) was simplified from that used in TREC-2, and they also used passage retrieval and whole document information in their ranking. The topics were expanded by 30 phrases that were automatically selected from a phrase "thesaurus" that had been previously built automatically from the entire corpus of documents.

CrnlEA -- Cornell University (see paper "Automatic Query Expansion Using SMART: TREC-3 by Chris Buckley, Gerard Salton, James Allan and Amit Singhal) used the vector-space SMART system, with term weighting similar to that done in TREC-2. The top 30 documents were used in a Rocchio relevance feedback technique to massively expand (500 terms + 10 phrases) the topics. No passage retrieval was done in this run; the second Cornell run (*CrnlLA*) used their local/global weighting schemes (with no topic expansion).

westp1 -- West Publishing Company (see paper "TREC-3 Ad Hoc Retrieval and Routing Experiments using the WIN System" by Paul Thompson, Howard Turtle,

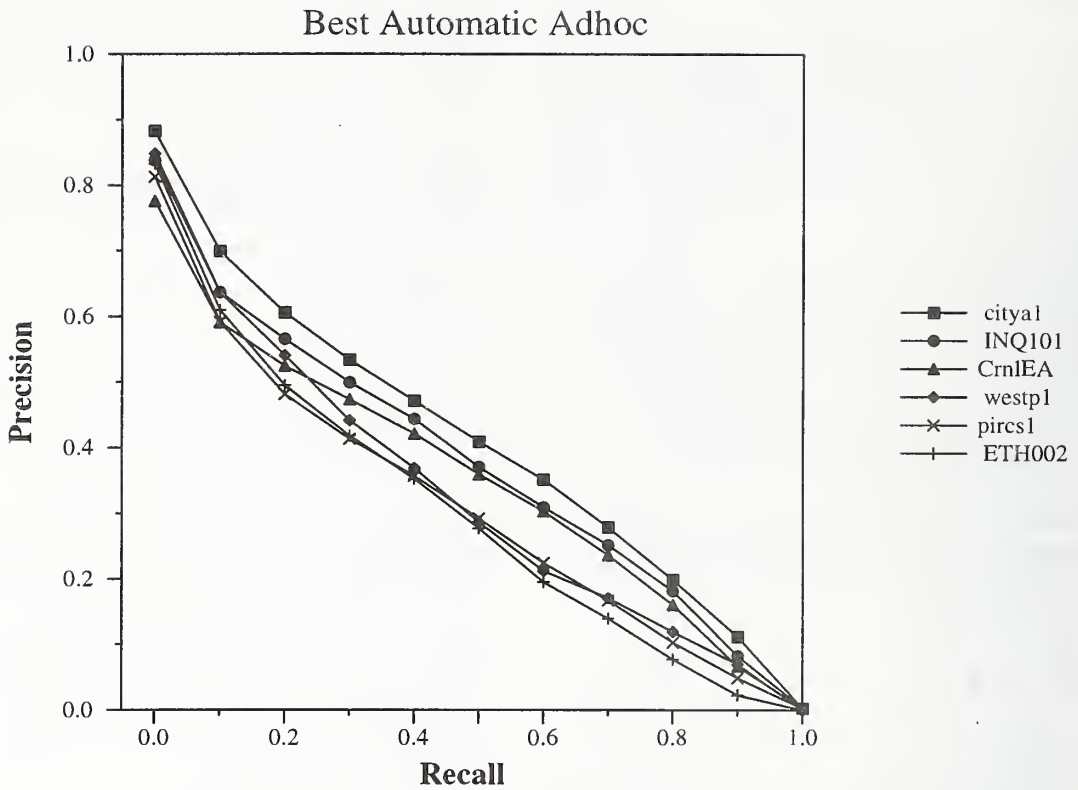


Figure 2. Best Automatic Adhoc Results.

Bokyung Yang and James Flood) used their commercial product (WIN) which is based on the same inference method used in *INQ101*. Both passages and whole documents were used in document ranking, but only minimal topic expansion was used, with that expansion based on preconstructed general-purpose synonym classes for abbreviations and other exact synonyms.

pircs1 -- Queens College, CUNY (see paper "TREC-2 Ad-Hoc, Routing Retrieval and Thresholding Experiments using PIRCS" by K.L. Kwok, L. Grunfeld and D.D. Lewis) used a spreading activation model on subdocuments (550-word chunks). Topic expansion was done by allowing activation from the top 6 documents in addition to the terms in the original topic. The highest 30 terms were chosen, with an average of 11 of those not in the original topic.

ETH002 -- Swiss Federal Institute of Technology (ETH) (see paper "Improving a Basic Retrieval Method by Links and Passage Level Evidence" by Daniel Knaus, Elke Mittendorf and Peter Schäuble) used a completely new method in TREC-3 based on combining information from three very different retrieval techniques. The three techniques are a vector-space system, a passage retrieval method using a Hidden Markov model, and a "topic expansion" method based on document links generated automatically from analysis of common phrases.

The dominant new themes in the automatic adhoc runs are the use of some type of term expansion beyond the terms contained in the shorter (TREC-3) topics, and some form of passage or subdocument retrieval element. Note that term expansion is mostly a recall device; adding new terms to a topic increases the chances of matching the wide variation of terms usually found in relevant documents. But adding terms also increases the "noise" factor, so accuracy may need to be improved via a precision device, and hence the use of passages, subdocuments, or more local weighting.

Two main types of term expansion were used by these top groups: term expansion based on a pre-constructed thesaurus (for example the INQUERY PhraseFinder) and term expansion based on selected terms from the top X documents (as done by City, Cornell, and PIRCS). Both techniques worked well. The top 3 runs (*cityal*, *INQ101*, and *CrnlEA*) have excellent performance (see Figure 2) in the "middle" recall range (30 to 80%), with this performance likely coming from the query expansion.

The use of the top 30 documents as a source of terms, as opposed to using the entire corpus, should be sensitive to the quality of the documents in this initial set. Notably, for 6 of the 8 topics in which the *INQ101* run was superior (a 20% or more improvement in average precision) to

the *city1* run, the *INQ101* run was also superior to the *CrnlEA* run. These topics tended to have fewer relevant documents, but also tended to be topics for which the systems bringing terms in manually (such as by manually selecting from a thesaurus or outside sources) did well.

Clearly there are topics in which this technique does not work well, but it does seem to provide an excellent focussing effect for many topics. This may not be the case outside of TREC, where there are fewer relevant documents. However, this type of expansion should be considered a worthwhile tool for query modification, especially for environments where no thesaurus exists.

Another factor in topic expansion is the number of terms being added to the topics. The average number of terms in the queries is widely varied, with the City group averaging around 50 terms (20 terms from expansion), the INQUERY system using around 100 terms on average, and the Cornell system using 550 terms on average. This huge variation seemed to have little effect on results, largely because each group found the level of topic expansion appropriate for their retrieval techniques. The *city1* run tended to "miss" more relevant documents than the *CrnlEA* run (7 topics were seriously hurt by this problem), but was better able to rank relevant documents within the 1000 document cutoff so that more relevant documents appeared in the top 100 documents. This better ranking could have happened because of the many fewer terms that were used, or could be caused by the use of passage retrieval in the City run.

The use of passages or subdocuments to reduce the noise effect of large documents has been used for several years in the PIRCS system. City, INQUERY and Cornell all did many experiments for TREC-3 to first determine the correct length of a passage, and then to find the appropriate use of passages in their ranking schemes. INQUERY and Cornell use overlapped passages of fixed length (200 words) as compared to City's non-overlapped passages of 4 to 30 paragraphs in length. All three systems use information from passages and whole documents retrieved rather than passage retrieval alone. (Cornell's version of this is called local/global weighting.) Both INQUERY and City combined the passage retrieval with query expansion; Cornell did two separate runs.

Note that the first two groups used passage retrieval to improve ranking and to regain the precision lost during the topic expansion. Cornell did not combine these operations even though they used term expansions on the order of 500 terms. The vector-space model seems less susceptable to "noise", as has been demonstrated in routing tasks. However in comparing the 2 Cornell runs, there were 16 topics in which the local/global run (*CrnlLA*) was superior, with 12 of these from better ranking, as opposed to only 8 topics that were superior in the expanded run (*CrnlEA*), 6 of which came from finding more relevant documents. A way of combining these runs should help performance, even for Cornell.

The *westp1* run did not use topic expansion, although a mixture of passages and whole documents was used in the final ranking of documents. The performance has suffered for this in the middle recall range. West Publishing used their production system to see how far it differed from the research systems and therefore did not want to use more radical topic expansion methods. Additionally they used a shortened topic (title + description + first sentence of narrative) because it was more similar in length to the topics submitted by their users. The *INQ101* run had 18 topics with superior performance to the *westp1* run, mostly because of new relevant documents being retrieved to the top 1000 document set. The 11 topics in which the *westp1* was superior to the *INQ101* run were mostly caused by better ranking for those topics.

The *pircs1* system used both passage retrieval (subdocuments) and topic expansion. This system used far fewer top documents for expansion (the top 6 as opposed to the top 30), and this may have hurt performance. There were 22 topics in which the *INQ101* run was superior to the *pircs2* run, and these were mostly because of missed relevant documents. Even though both systems added about the same number of expansion terms, using only the top 6 documents as a source of terms for spreading activation might have provided too much focussing of the concepts.

The *ETH001* run used both topic expansion and passages, in addition to a baseline vector-space system. Both the topic expansion and the passage determination were completely new (untried) techniques; additionally there are known difficulties in combining multiple methods. In comparison to the Cornell expansion results (*CrnlEA*), the main problems appear to be missed relevant documents for all 17 of the topics where the Cornell results were superior. The 8 topics with superior ETH results were mostly because of better ranking. Clearly this is a very promising approach and more experimentation is needed.

Table 7 shows a breakdown of improvements from expansion and passage retrieval that combines information from the non-official runs given in the individual papers. In general groups seem to be getting about 20% improvement over their own baselines (less for ETH and PIRCS), with that improvement coming in different percentages from passage retrieval or expansion, depending on the specific retrieval techniques being used.

Figure 3 shows the recall/precision curves for the 6 TREC-3 groups with the highest non-interpolated average precision using manual construction of queries. A short

Table 7: Comparison of Performance (Average Precision)
for Passage Retrieval and Topic Expansion

	base run	passages	expansion	both
City	0.337	-	0.388 (15%)	0.401 (19%)
INQUERY				
(11 pt. average)	0.318	0.368 (16%)	0.348 (9%)	0.381 (20%)
Cornell	0.2842	0.3302 (16%)	0.3419 (20%)	-
ETH	0.2578	0.2853 (11%)	0.2737 (6%)	0.2916 (13%)
PIRCS	-	0.2764	-	0.3001 (9%)

summary of the techniques used in these runs follows. Again, for more details on the various runs and procedures, see the appropriate papers in this proceedings.

INQ102 -- University of Massachusetts at Amherst. This run is a manual modification of the *INQ101* run, with strict rules for the modifications to only allow removal of words and phrases, modification of weights, and addition of proximity restrictions.

Brkly7 -- University of California, Berkeley (see paper "Experiments in the Probabilistic Retrieval of Full Text Documents" by William S. Cooper, Aitao Chen and Fredric C. Gey) is a modification of the *Brkly6* run, with that modification being the manual expansion of the queries by adding synonyms found from other sources. The *Brkly6* run uses a logistic regression model to combine information from 6 measures of document relevancy based on term matches and term distribution. The coefficients were learned from the training data in a manner similar to that done in TREC-2, but the specific set of measures used has been expanded and modified for TREC-3. No passage retrieval was done.

ASSCTV1 -- Mead Data Central, Inc (see paper "Query Expansion/Reduction and its Impact on Retrieval Effectiveness" by X. Allan Lu and Robert B Keefer) is also a manual expansion of queries using an associative thesaurus built from the TREC data. The retrieval system used in *ASSCTV1* is the SMART system.

VTc2s2 -- Virginia Tech (see paper "Combination of Multiple Searches" by Joseph A. Shaw and Edward A. Fox) used a combination of multiple types of queries, with 2 types of natural language vector-space queries and 3 types of manually constructed P-Norm (soft Boolean) queries.

pircs2 -- Queens College, CUNY. This run is a modification of the base PIRCS system to use manually constructed soft Boolean queries.

rutfua1 -- Rutgers University (see paper "Decision Level Data Fusion for Routing of Documents in the TREC3 Context: A Best Cases Analysis of Worst Case Results" by Paul B. Kantor) used data fusion methods to combine the retrieval ranks from three different retrieval schemes all using the INQUERY system. Two of the schemes used Boolean queries (one with ranking and one without) and the third used the same queries without operators.

The three dominant themes in the runs using manually constructed queries are manual modification of automatically generated queries (*INQ102*), manual expansion of queries (*Brkly7* and *ASSCTV1*) and combining of multiple retrieval techniques or queries. Three runs can be compared to a "baseline" run to check the effects of manual versus automatic query construction.

INQ102, the manually modified version of *INQ101*, had a 15% improvement in average precision over *INQ101*, and 17 topics that were superior in performance for the manual system (as opposed to only 3 for the automatic system). An analysis of those topics shows that many more relevant documents were in the top 1000 documents and the top 100 documents, probably caused by manually eliminating much of the noise that was producing higher ranks for nonrelevant documents. This noise elimination could have happened because many spurious terms had been manually removed from the queries (*INQ102* had an average of about 30 terms as opposed to nearly 100 terms in *INQ101*), or could have come from the use of the proximity operators.

The *Brkly7* run, a manually expanded version of *Brkly6*, used about the same number of terms as the *INQ102* run (around 36 terms on average), but the terms had been manually pulled from multiple sources (as opposed to editing an automatic expansion as done by INQUERY). The improvement from *Brkly6* to *Brkly7* is a 34% gain in average precision, with 25 topics having superior performance in the manually expanded run. Note however that there was no topic expansion done in the automatic *Brkly6* run, so this improvement represents the results of a good manual topic expansion over no expansion at all.

The INQUERY system outperforms the Berkeley system by 14% in average precision, with much of that difference coming in the high recall end of the graph (see Figure 3).

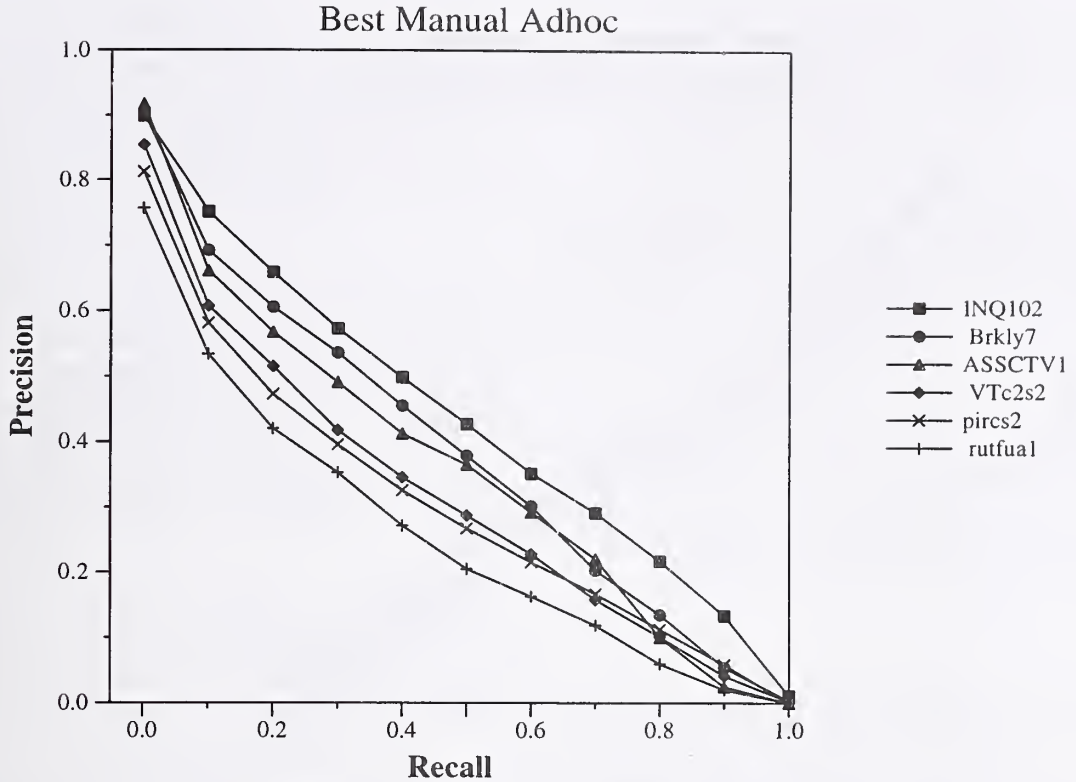


Figure 3. Best Manual Adhoc Results.

This is consistent with the difference in their topic expansion techniques in that the automatic expansion (even manually edited) is likely to bring in terms that users might not select from "non-focussed" sources.

The *ASSCTV1* run also represents a manual expansion effort, but using a pre-built thesaurus as opposed to using textual sources for the expansion. The topics were expanded to create a query averaging around 135 terms and then were run using the default Cornell SMART system. A comparison of the automatically expanded *CrnlEA* run and the manually expanded *ASSCTV1* run shows minimal difference in average precision, but superior performance in 18 of the topics for the manual expansion (as opposed to only 10 of the topics having superior performance for the automatic Cornell run). In both cases, the improvements come from finding more relevant documents because of the expansions, but different expansion methods help different topics.

The *pircs2* run is a manual query version of the baseline PIRCS system. A soft Boolean query is created from the topic, but no topic expansion is done. There is minimal difference in average precision between the two PIRCS runs, but more topics show superior performance for the soft Boolean query *pircs2* run (8 superior topics versus 4 superior topics for the topic expansion *pircs1* run). It is not clear whether this difference comes from the increased precision of the soft Boolean approach or from the relatively poor performance of the PIRCS term expansion results.

In TREC-3, as opposed to TRECs 1 and 2, the manual query construction methods perform better than their automatic counterparts. The removal of some of the topic structure (the concepts) has allowed differences to appear that could not be seen in earlier TRECs. Since topic expansion was necessary to produce top scores, the superiority of the manual expansion over no expansion in the Berkeley runs should not be surprising. Less clear is why the manual modifications in the *INQ102* run showed superior performance to the automatic run with no modifications. The likely explanation is that the automatic term expansion methods are relatively uncontrolled in TREC-3 and manual intervention plays an important role.

The last two groups in the top six systems using manual query construction used some form of combination of retrieval techniques. The Virginia Tech group (*VTc2s2*) combined the results of up to 5 different types of query construction (3 P-Norms with different P values and 2 vector-space, one short and one manually expanded) to create their results. They used a simple combination method (adding all the similarity values) and tested various combinations of query types. Their best result combined only two of the query types, one a P-Norm and one

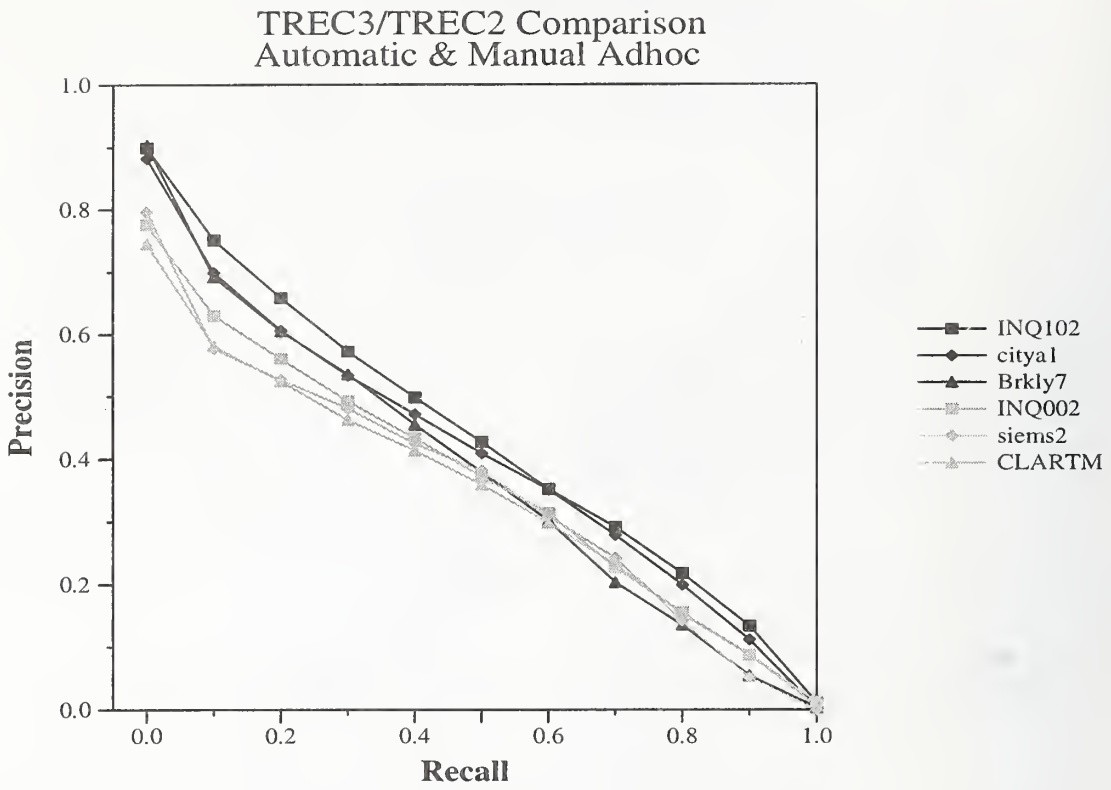


Figure 4. Comparison of Adhoc Results for TREC-2 and TREC-3

a vector-space. A series of additional runs (see paper for details) confirmed that the best method was to combine the results of the best two query techniques (the "long" vector-space and the P=2 P-Norm). They concluded that improvements from combining results only occurred when the input techniques were sufficiently different.

Although the Rutgers group (*rutfua1*) used more elaborate combining techniques, they came to the same conclusion. Combining different retrieval techniques offers improvements over a single technique (over 30% for the Virginia Tech group), but the input techniques need to be more varied to get further improvements. But the more varied the individual techniques, the more need for elaborate combining methods such as used in the *rutfua1* run. The automatic *ETH001* run best exemplifies the direction needed here; first getting "good" performance for three very different but complementary techniques and then discovering the best ways of combining results.

Several comments should be made with respect to the overall adhoc recall/precision averages. First, the better results are very similar and it is unlikely that there is any statistical difference between them. The Scheffe´ tests run by Jean Tague-Sutcliffe (see paper "A Statistical Analysis of the TREC-3 Data" by Jean Tague-Sutcliffe and James Blustein) show that the top 20 category A runs (manual and automatic mixed) are all statistically equivalent at the α=0.05 level. This lack of system differentiation comes from the very wide performance variation across topics (the cross-topic variance is much greater than the cross-system variance) and points to the need for more research into how to statistically characterize the TREC results.

As a second point, it should be noted that these adhoc results represent significant improvements over TREC-2. Figure 4 shows the top three systems in TREC-3 and the top three systems in TREC-2. This improvement was unexpected as the removal of the concepts section seemed likely to cause a considerable performance drop (up to 30% was predicted). Instead the advance of topic expansion techniques caused major improvements in performance with less "user" input (the concepts). Because of the different sets of topics involved, the exact amount of improvement cannot be computed. However the Cornell group has run older systems (those used in TREC-1 and TREC-2) against the TREC-3 topics. This shows an improvement of 20% for their expansion run (*CrnlEA*) over the TREC-2 system, and this is likely to be typical for many of the systems this year.

5.3 Routing Results

The routing evaluation used a subset of the training topics (topics 101-150 were used) against the disk of test docu-

ments (disk 3). Although this disk had been used in TREC-2, its use in TREC-3 was unexpected as new data had been promised. The last minute unavailability of this new data made the reuse of disk 3 necessary, but since groups had not been training with this disk (and no relevance judgments were available for this disk against topics 101-150), the routing results should not be biased by the reuse of old material.

The routing task in TREC has remained constant; however there has been a major evolution in the thrust of the research for this task. There was minimal training data for TREC-1, and most groups felt that their results were even more preliminary than for the adhoc results because the training data that was available was incomplete and inconsistent. This means that routing became a particularly interesting challenge in TREC-2 when adequate training data (the results from TREC-1 adhoc topics) became available.

The TREC-2 results therefore represent an excellent baseline of what could be achieved using traditional algorithms with large amounts of relevance information. Most notable was the effective use of the Rocchio feedback algorithm in SMART, where up to 500 new terms were added to the routing topics from the training data. Equally good results were achieved by a probabilistic system from the University of Dortmund, where only 30 terms were added, but very precise term weighting was learned from the training data. Manual construction of queries consistently gave poorer performance as the availability of training data allowed an automatic tuning of the queries that would be difficult to duplicate manually without extensive analysis.

For TREC-3, many groups made only minor modifications to their TREC-2 techniques (and concentrated on the adhoc task). There were a total of 49 sets of results for routing evaluation, with 46 of them based on runs for the full data set. Of the 46 systems using the full data set, 24 used automatic construction of queries, 18* used manual construction, and 4 used interactive query construction.

Figure 5 shows the recall/precision curves for the 12 TREC-3 groups with the highest non-interpolated average precision for the routing queries. The runs are ranked by the average precision and only one run per group is shown (both official runs sometimes would have qualified for this set). A short summary of the techniques used in these runs follows. For more details on the various runs and procedures, please see the appropriate papers in this proceedings.

cityr1 -- City University, London (see paper "Okapi at TREC-3" by S.E. Robertson, S. Walker, S. Jones, M.M. Hancock-Beaulieu and M. Gatford) used the same probabilistic techniques as for the adhoc task, but constructed the query using a very selective set of terms (17 on average) from the relevant documents.

pircs3 -- Queens College, CUNY (see paper "TREC-2 Ad-Hoc, Routing Retrieval and Thresholding Experiments using PIRCS" by K.L. Kwok, L. Grunfeld and D.D. Lewis) used a spreading activation model based on the topic and on terms selected from about 35% of the relevant material.

INQ103 -- University of Massachusetts at Amherst (see paper "Document Retrieval and Routing Using the INQUERY System" by John Broglio, James P. Callan, W. Bruce Croft and Daniel W. Nachbar) used the inference net engine (same as for the adhoc task), with topic expansion of about 60 terms selected from the relevant documents.

dortR1 -- University of Dortmund (see paper "Routing and Ad-hoc Retrieval with the TREC-3 Collection in a Distributed Loosely Federated Environment" by Nikolaus Walczuch, Norbert Fuhr, Michael Pollmann and Birgit Sievers) used the SMART retrieval system with a Rocchio relevance feedback expansion adding 12% new terms and 4% new phrases from the training documents.

lsir2 -- Bellcore (see paper "Latent Semantic Indexing (LSI): TREC-3 Report" by Susan Dumais) used the latent semantic indexing system to construct a reduced dimension vector centroid of the relevant documents (no use was made of the topics).

CrnlRR -- Cornell University (see paper "Automatic Query Expansion Using SMART: TREC-3 by Chris Buckley, Gerard Salton, James Allan and Amit Singhal) used the vector-space SMART system and a basic Rocchio relevance feedback algorithm adding about 300 terms and 30 phrases to the topic.

Brkly8 -- University of California, Berkeley (see paper "Experiments in the Probabilistic Retrieval of Full Text Documents" by William S. Cooper, Aitao Chen and Fredric C. Gey) used only the relevant documents to select a large number of terms (average 1,357 terms/topic) which were combined and weighted using a logodds formula. A chi-square test was used to select the terms.

---

* 11 of these runs were abbreviated runs from one group

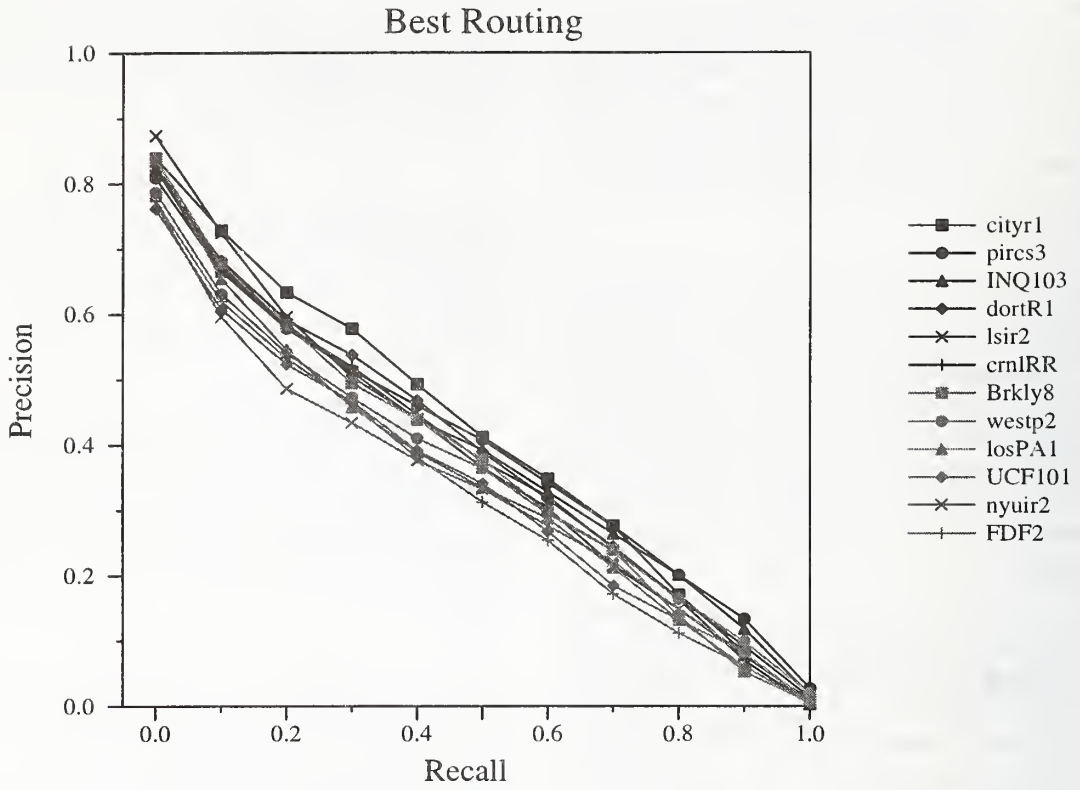


Figure 5. Best Routing Results.

westp2 -- West Publishing Company (see paper "TREC-3 Ad Hoc Retrieval and Routing Experiments using the WIN System" by Paul Thompson, Howard Turtle, Bokyung Yang and James Flood) used their commercial product (WIN), but expanded the topics using up to 50 terms from specially selected parts of relevant documents.

losPA1 -- Logicon, Inc. (see paper "Research in Automatic Profile Creation and Relevance Ranking with LMDS" by Julian A. Yochum) constructed profiles based on the top 10 selected terms from the relevant documents, with term selection based on binomial probability distributions. The profile was used to select all documents containing any of those terms and the documents were then ranked using a weighting formula.

UCF101 -- University of Central Florida (see paper "Using Database Schemas to Detect Relevant Information" by James Driscoll, Gary Theis and Gene Billings) manually constructed entity-relationship schemas for each topic and also manually created synonym lists for each labelled component in the ER schema. These schemas and lists were then used to select and rank documents.

nyuir2 -- New York University (see paper "Natural Language Information Retrieval: TREC-3 Report" by Tomek Strzalkowski, Jose Carballo and Mihnea Marinescu) used NLP techniques to discover syntactic phrases in the documents. Both single terms and phrases were indexed and specially weighted. The *nyuir2* run used topic expansion based on the relevant documents.

FDF2 -- Paracel, Inc. (see paper "The FDF Query Generation Workbench" by K.I. Yu, P. Scheibe and F. Nordby) used a series of tools to generate profiles. These tools used statistical methods to create several alternative queries, and automatically evaluated the queries against the training data to select the best query for each topic.

The recall/precision curves shown in Figure 5 are very close in performance for the routing, with the Scheffe´ tests done by Jean Tague-Sutcliffe showing that there is no significant differences between the top 22 runs. It is, however, useful to look at the results on a per topic basis to find trends in performance across techniques.

The main issue for the TREC-3 routing runs is how to best select terms for topic expansion. Note that for the adhoc task the main issue was how to expand a topic beyond its original terms, hopefully with as little loss in precision as possible. For the routing task, however, the pool of terms for expansion is easily determined (i.e., the terms in the relevant documents), and the problem is how to select terms from this very large pool. Correspondingly, the major differences in results between the routing

runs are not how many relevant documents were "missed" (as for the adhoc task), but how well the relevant documents were ranked.

An example of this is a comparison between the two City runs. The *cityr1* system used all relevant documents to select the top T terms, where T varied between 3 and 100 (average 47). Then they used the training material to optimize the queries, selecting only those terms that improved results. On average only about 17 terms were used in an optimized query. The unoptimized version of these queries was used at the *cityr2* run (not shown in Figure 5), which did not work as well. The difference in average precision between the two runs is only about 12%, but the optimized *cityr1* run had 14 superior topics (topics with a 20% or more improvement in average precision), all caused by better ranking (more relevant documents moved into the top 100 documents from the top 1000 documents). A similar comparison can be made between the *cityr1* run and the *pircs3* run. Even though there were more relevant documents found by the *pircs3* technique, the *cityr1* run had 15 superior topics (versus 7 superior for *pircs3*), all caused by better ranking.

The ability to assign better ranks to relevant documents is not strictly tied to being highly selective of terms. A comparison of the *cityr1*, *pircs3*, *INQ103* and *CrnlRR* runs shows that the INQUERY and PIRCS techniques both used an average of around 100 terms in their queries and retrieved the largest number of relevant documents in the top 1000 documents. The *cityr1* run, with only about 17 terms, missed a few relevant documents, but did a much better job of ranking the ones they found. However, even though the *CrnlRR* run used a massive expansion of greater than 300 terms, the *CrnlRR* runs were stronger in ranking than in finding relevant documents. A comparison of the *INQ103* run to that of Cornell shows that Cornell had 12 "inferior" topics, mostly due to missed relevant documents, and 9 superior topics, mostly due to better ranking. Clearly the appropriate number of terms to use in a routing query varies across retrieval techniques. This same result was seen in the adhoc task, where the appropriate number of expansion terms also varied across systems.

The top routing results tend to fall into three categories--those groups that used minimal effort in selecting terms (*CrnlRR*, *lsir2*), those groups that selected terms based on using only a portion of the relevant material (*pircs3* and *westp2*), and those groups that used all the material, but carefully selected terms (*cityr1*, *INQ103*, *brkly8* and *losPA1*).

Both the Cornell runs and the LSI runs were repeats of their TREC-2 techniques. The LSI runs tested using only the topic to create a query (no expansion) versus using all the relevant documents (no topic) to create a centroid for use as the query (the *lsir2* run). There is a 30% improvement using the relevant documents only. The Cornell runs used both the topic and a massive Rocchio relevance feedback expansion (300+ terms). Both groups used techniques based on a vector-space model (loosely based for the LSI technique), and this model appears to be able to effectively rank documents despite very massive queries. The strength of the Cornell ranking was mentioned before, but the LSI ranking is comparable or even better (18 superior topics for LSI, 9 for Cornell, all caused by better ranking).

Two groups (the PIRCS system and the WIN system from West) experimented with using only portions of the training data. This is mostly an efficiency issue, but also serves as a term selection method. The *pircs4* run (not shown in Figure 5) used only short documents, where short is defined as not more than 160 unique non-stop stems. This run did somewhat worse than the *pircs3* run, where a combination of these short documents and the top 2400 subdocuments were used. In both runs many fewer documents were used (12% and 35% of the relevant material respectively), yet the results were excellent. The West group tried multiple experiments using various segments of the relevant documents (best documents only, best 200 paragraphs, and best top paragraph). Up to 50 terms were added using a combination of the various approaches, with selection of approaches done on a per topic basis. This selective use of material caused some relevant documents to be missed. A comparison of the *westp2* run and the *INQ103* run shows that the 12 topics in which the *INQ103* run was superior were mostly caused by new relevant documents being found, whereas the 7 topics in which the *westp2* run was superior were all caused by better ranking.

Four groups (*cityr1*, *INQ103*, *brkly8*, and *LosPA1*) used all the relevant documents, but made careful selection of the terms to use. The City results have already been discussed. The *INQ103* run used an adaptation of the Rocchio algorithm with their inference engine technique. A statistical formula was used to select the top 32 terms to use for expansion for each topic, and then 30 additional terms were selected based on their proximity to those terms already selected. This technique retrieved a large number of the relevant documents into the top 1000 slots, but had more difficulties doing the ranking within that set. The *brkly8* run selected an average of over 1000 terms by using a chi-square test to indicate which stems were statistically associated with document relevance to a topic. These terms were weighted and used as the query. The *losPA1* run used a similar technique, calculating a binomial probability to select the top 1000 terms, selecting a pool of documents using an OR of the top 10 terms,

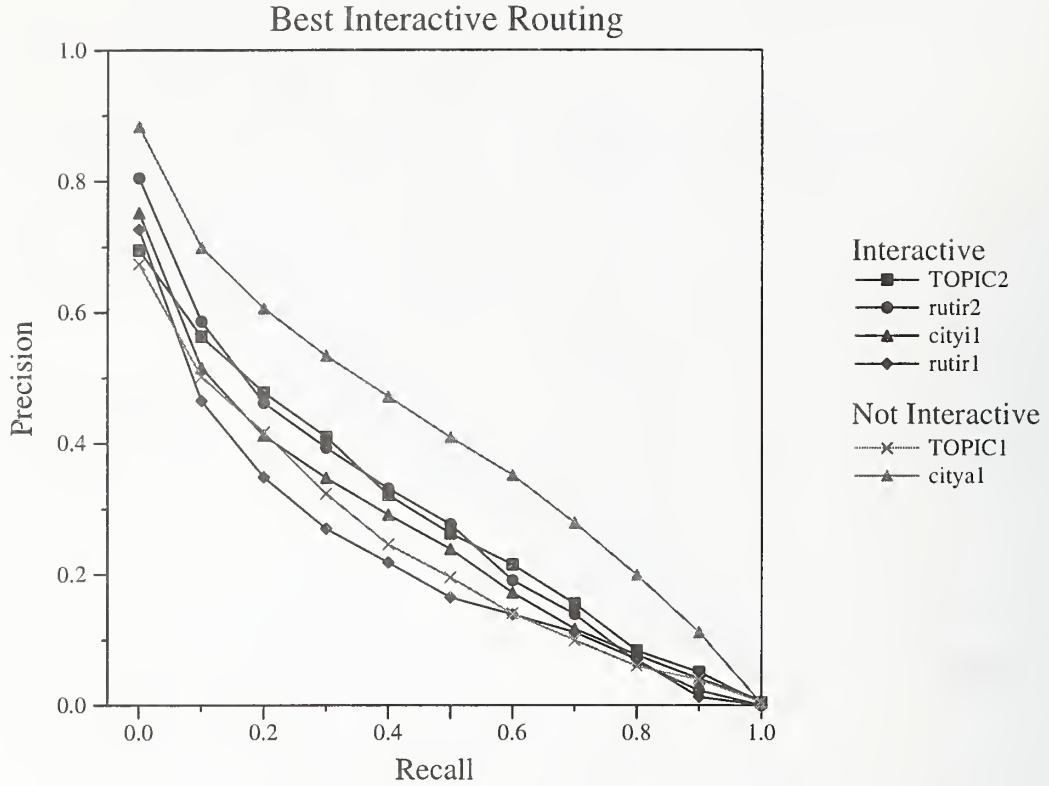


Figure 6. Interactive Results.

and then scoring the documents using a weighting algorithm based on occurrances of the 1000 terms in those documents. If results from these two systems are compared to the more traditional *INQ103* method, it seems that the strengths of these methods are in the ranking, with some problems in missing relevant documents.

As was the case in earlier TRECs, the manual construction of routing queries was not very competitive with automatic query construction. The manual *INQ104* run, consisting of a merge of the *INQ103* queries and a manually edited version of these queries was little different in results from the *INQ103* run. An exception to this was the reasonable results of the *UCF101* run. This run combined manually constructed detailed entity-relationship schema with manually constructed synonym lists. These were run against the documents, producing results that are comparable with the automatic results.

There is some improvement in overall routing results compared with those from TREC-2. This is mostly shown by the comparative position of the *CrnlRR* run, which was the "top-ranked" run in TREC-2, and now is more the "middle of the pack."

5.4 Other Experiments in TREC-3

In addition to the results aimed at producing high recall/precision performances, several groups did experiments using the TREC tasks to investigate other areas.

The largest area of experimentation was in interactive query construction, with four groups participating. One of the questions addressed by these groups was how well humans could perform the routing task, given a "rules-free" environment and access to the training material. The larger issue addressed by these experiments, however, was the entire interaction process in retrieval systems, since the "batch mode" evaluation of TREC does not reflect the way that most systems are used.

Figure 6 shows the three sets of results for the category A interactive runs, plus several baseline runs for comparison. A short summary of the systems follows, and readers are referred to the individual papers for more details.

TOPIC2 -- Verity, Inc. (see paper "Interactive Document Retrieval Using TOPIC (A report on the TREC-3 experiment)" by Richard Tong) used 12 Verity staff members ranging in search experience using TOPIC from novice to expert to build their queries. The initial queries were the manual-constructed queries used by Verity in TREC-2, and the results from these queries are shown in Figure 6 as *TOPIC1*. The searchers then improved the initial queries by periodically evaluating their "improved" queries against the training data. When sufficiently

improved scores were achieved, the queries were declared final and used for TREC-3.

rutir1, *rutir2* -- Rutgers University (see paper "New Tools and Old Habits: The Interactive Searching Behavior of Expert Online Searches using INQUERY" by Jurgen Koenemann, Richard Quatrain, Colleen Cool and Nicholas Belkin) used the INQUERY system and had 10 experienced online searchers with no prior experience using that system build their queries. The entire query building process was restricted to 20 minutes per topic, and used the training data both for automatic relevance feedback (if desired) and for the searchers to check if a given retrieved document was relevant (as opposed to periodically evaluating their results). At some point during the 20 minute limit the queries were declared finished by the searchers and the results from these queries are shown in Figure 6 as *rutir1*. As a comparison, the experimenters also did the task themselves (*rutir2*).

cityi1 -- City University, London (see paper "Okapi at TREC-3" by S.E. Robertson, S. Walker, S. Jones, M.M. Hancock-Beaulieu and M. Gatford) used the OKAPI team as searchers. The initial query was manually generated using traditional operations. The retrieved documents (or a brief summary of them) were then displayed, and searchers checked the relevance judgments (generally viewing 10 or 12 relevant documents). Automatic relevance feedback was then applied and the searchers could choose to modify the resulting query or not (35 of the 50 topics were modified). Multiple iterations could be done before a decision was made on the final query.

Not shown in Figure 6 is a category B interactive result from the University of Toronto (see paper "Interactive Exploration as a Formal Text Retrieval Method: How Well can Interactivity Compensate for Unsophisticated Retrieval Algorithms" by Nipon Charoenkitkarn, Mark Chignell and Gene Golovchinsky). This group developed their TREC experiments from what was initially a browsing system. Boolean operators and promixity operators were used to construct the initial query. The queries were then "loosened" until around 1000 documents were retrieved. Then the results of these queries were run against the training data and reviewed, with changes possibly made to the query based on retrieval results.

As a group, the interactive results were considerably worse than the automatic routing results. This was somewhat unexpected since in all four cases the queries could be classified as the best manual queries possible. Although no definite reasons have been cited for this, the likely cause is the very strong performance of the automatic systems given the large amounts of training data.

A comparison of the City interactive run (*cityi1*) and the City automatic run (*citya1*) illustrates the problems. For BOTH runs, the query lengths were short, an average of around 17 terms. Only about 20% of these terms were in common, i.e., the searchers (*cityi1*) and the "computer" (*citya1*) picked different sets of terms. The difference in the results from these queries, however, is very large, as shown in Figure 6. The automatic run has a 63% improvement in average precision, and 33 topics with superior results (a 20% or more improvement in average precision) versus one topic with inferior results.

Regardless of the poorer performance, all four groups were able to draw interesting conclusions about their own interactive experiments. The Verity group found a 24% improvement in results (*TOPIC1* to *TOPIC2*) that can be obtained by humans using the training material over the (manually created) initial query. Other groups were able to gain insight into better tools needed by their system or insight into how online searchers handle the new techniques available. Of particular interest are the reports in these papers about the detailed human/computer interactions, as this provides insight on how systems might work in an operational setting.

A second area that drew more attention in TREC-3 was that of efficiency. Efficiency has always been an issue in TREC because sufficient efficiency (in both time and storage) is necessary to finish the tasks, and greater efficiency allows more experiments to be done within the same time period. Additionally the commercial systems in TREC must make any new algorithms fit into their already very efficient methodologies (the TRW/Paracel Fast Data Finder is a good example of these problems).

Many groups addressed efficiency issues in their TREC-3 papers, but the group from RMIT (see paper "Information Retrieval Systems for Large Document Collections" by Alistair Moffat and Justin Zobel) has specialized in efficiency issues in all the TRECs. In TREC-3 they investigated the issue of creating a centralized index in blocks for more efficient retrieval. They also tested text compression methods for dynamic document databases. Efficiency is likely to continue to be a major issue in TREC, possibly playing a larger part in the future.

A third area, that of properly handling heterogeneous collections such as the five main "subcollections" in TREC, was comprehensively addressed by the Siemens group (see paper "The Collection Fusion Problem" by Ellen Voorhees, Narendra Gupta and Ben Johnson-Laird). This group examined two different collection fusion techniques and was able to obtain results within 10% of the average precision of a run using a merged collection index. This type of investigation is important for real-world collections, and also to allow researchers to take advantage of

possible variations in retrieval techniques for heterogeneous collections.

Several groups ran some experiments in thresholding as an alternative method of evaluating the routing task. For details on one of these experiments, see paper "TREC-2 Ad-Hoc, Routing Retrieval and Thresholding Experiments using PIRCS" by K.L. Kwok, L. Grunfeld and D.D. Lewis.

The final set of experiments in TREC-3 involved starting work in a second language. Four groups worked with 25 topics in Spanish, using a document collection consisting of about 200 megabytes (58,000 records) of a Mexican newspaper from Monterey (*El Norte*). Since there was no training data for testing (similar to the startup problems for TREC-1), the groups used simple techniques.

CrnlVS, *CrnlES* -- Cornell University (see paper "Automatic Query Expansion Using SMART: TREC-3 by Chris Buckley, Gerard Salton, James Allan and Amit Singhal) used a baseline SMART run (*CrnlVS*) and a SMART run with massive topic expansion (*CrnlES*) similar to their English adhoc run. A simple stemmer and a stoplist of 342 terms were used.

SIN002, *SIN001* -- University of Massachusetts at Amherst (see paper "Document Retrieval and Routing Using the INQUERY System" by John Broglio, James P. Callan, W. Bruce Croft and Daniel W. Nachbar) used the INQUERY system, with *SIN001* being a manually modified version of a basic automatic INQUERY run (*SIN002*) without topic expansion. A Spanish stemmer produced a 12% improvement in later experiments.

DCUSP1 -- Dublin City University (see paper "Indexing Structures Derived from Syntax in TREC-3: System Description" by Alan Smeaton, Ruairi O'Donnell and Fergus Kelledy) used a trigram retrieval model, with weighting of the trigrams from traditional frequency weighting. A Spanish stemmer based on the Porter algorithms was also used.

erims1 -- Environmental Research Institute of Michigan (see paper "Using an N-Gram-Based Document Representation with a Vector Processing Retrieval Model" by William Cavnar) used a quad-gram retrieval model, also with weighting using some of the traditional weighting mechanisms.

The major result from this very preliminary experiment in a second language was the ease of porting the retrieval techniques across languages. Cornell reported that only 5 to 6 hours of system changes were necessary (beyond creation of any stemmers or stopword lists).

6. Summary

The main conclusions that can be drawn from TREC-3 are as follows:

Automatic construction of routers or filters from training data is very effective, much more effective than manual construction of these types of queries. This holds even if the manual construction is based on unrestricted use of the training data.

- Expansion of the shorter TREC-3 topics was highly successful, using either automatic topic expansion, manual topic expansion, or manually modified versions of automatically expanded topics. Many different techniques were effective, with research just beginning in this new area.
- The use of passage retrieval, subdocuments, and local weighting brings consistent performance improvements, especially in the adhoc task. Experiments this year show continued improvement coming from various methods of using these techniques to improve ranking.
- Preliminary results suggest that the extension of basic English retrieval techniques into another language (in particular Spanish) does not appear difficult. TREC-3 represents the first large-scale test of this portability issue.

Do these conclusions hold in the real world of text retrieval? Certainly the use of automatic construction of routers will work in any environment having reasonable amounts of training material. Of greater question is the transferability of the adhoc results. Two particular issues need to be addressed here. First, even though the topics in TREC-3 are shorter, they are still considerably longer than most queries used in operational settings. A couple of sentences is likely to be the maximum a user is willing to type into a computer, and it is unclear if the TREC topic expansion methods would work on these shorter input strings. Shorter topics may also need different techniques of passage retrieval and local weighting. TREC-4 will address this issue by using appropriately shorter topics.

The second mismatch of the TREC-3 results to the real-world is the emphasis on high recall in TREC. Requesting 1000 ranked documents and calculating the results on these goes well beyond average user needs. Karen Sparck Jones addresses this issue by looking at retrieval performance based only on the top 30 documents retrieved [Sparck Jones 1995, updated for TREC-3 in Appendix C to the proceedings]. An improvement of 20% in precision at this cutoff means that six additional relevant documents will be returned to the user, and this is likely to be noticeable by many users. Many of the techniques used in

TREC produced this difference; additionally some of the tools being investigated in TREC, such as the topic expansion tools, will make query modification much easier for the average user.

There will be a fourth TREC conference in 1995, and most of the systems that participated in TREC-3 will be back, along with additional groups. The routing and adhoc tasks will be done again, with different data and even shorter adhoc topics. In addition special new tasks (call "tracks") will be created to provide a focus to those areas of TREC that have been attracting more experimental interest. Six tracks will be tried.

- Interactive -- investigating searching as an interactive task by examining the process as well as the outcome.
- Multilingual -- working with non-English test collections (250 megabytes of Spanish and 25 topics, plus possibly Chinese and/or Japanese collections).
- NLP -- more focussed investigation of NLP in an IR environment, emphasizing the discovery and use of phrases for TREC-4.
- Multiple database merging -- investigation of techniques for merging results from the various TREC subcollections.
- Data corruption -- examining the effects of corrupted data (such as would come from an OCR environment) by using corrupted versions of the TREC data.
- Filtering -- evaluating routing systems on the basis of retrieving an unranked set of documents optimizing a specific effectiveness measure.

Acknowledgments

The author would like to gratefully acknowledge the continued support of the Software and Intelligent Systems Technology Office of the Advanced Research Projects Agency for the TREC conferences. Special thanks also go to the TREC program committee and the staff at NIST.

7. References

Harman D. (Ed.). (1993). *The First Text REtrieval Conference (TREC-1).* National Institute of Standards and Technology Special Publication 500-207, Gaithersburg, Md. 20899.

Harman D. (Ed.). (1994a). *The Second Text REtrieval Conference (TREC-2).* National Institute of Standards and Technology Special Publication 500-215, Gaithersburg, Md. 20899. Also a special issue of *Information Processing and Management*, 31(3) in press.

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

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.

Sparck Jones K. (1995). Reflections on TREC. *Information Processing and Management*, 31(3), in press.

Xerox TREC 3 Report: Combining Exact and Fuzzy Predictors

Hinrich Schütze Jan O. Pedersen Marti A. Hearst

Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
{schuetze,pedersen,hearst}@parc.xerox.com

1 Overview

We employed different processing techniques for the ad hoc and routing tasks, although both explored the combination of exact and fuzzy predictors.

For the routing task, we used a neural net classifier to estimate the probability of relevance of a new document to a given topic description. We experimented with a variety of document representations, finally settling on a hybrid feature set that combined a selected number of discriminating terms with a low dimensional local LSI [9] for generalization.

In the ad hoc task, we applied some of our content analysis techniques, in particular, automatic thesaurus construction and automatic topical segmentation of full-length texts. Working from the hypothesis that exact term match can be too restrictive, we used *thesaurus vectors* [13] to compute *context vectors* for full texts and text segments. We also used thesaurus vectors to decompose the terms contained in the topic descriptions into sets of *query factors*, where each query factor is intended to represent a semantically distinct component of the topic description. These factors constrain document search by imposing Boolean constraints as will be described below. We used *TextTiling* [6] to partition long documents into semantically motivated multi-paragraph segments called *tiles*. We adopted the logistic regression methodology of Cooper et al. [2] and Fuhr et al. [5] to model the probability of relevance given measured attributes of a query-document pair. This allowed us to combine standard predictors (such as number of match terms) with various assessments of tile match, query factor score, and other predictors.

For both tasks we first performed standard preprocessing (document parsing, tokenization, stop-list term removal) using the TDB system [3]. Our *terms* consisted of single words and two-word phrases that occur over five times in the corpus (where phrase is defined as an adjacent word pair, not including stop words). This process produced over 2.5 million terms, which were further processed as described below.

In the remainder of this report, Section 2 describes our routing experiments, Section 3 describes our ad hoc experiments, and Section 4 discusses our results and possible future experiments.

2 Routing

Figure 1 sketches the system architecture for routing. Our routing method combines a neural network classifier and a hybrid representation based on terms and local Latent Semantic Indexing (LSI) vectors [4, 9].

Routing is a statistical classification problem, although the classifiers that are typically employed have relatively restricted generalization capacity (e.g., logistic regression [2] and other linear methods). We explored the use of a more powerful family of classifiers: neural networks. They support both linear and non-linear architectures and can model complex functions (e.g., Boolean functions of arbitrary complexity).

The input to the classifier consists of two blocks of 100 units, one block for 100 highly discriminating terms, and one block for a local LSI vector. This representation allows for both exact term match (from the discriminating terms) as well as a fuzzy similarity measure (from the local LSI vector). For long documents, the terms and vectors are computed only for one subpart of the document, the segment with the highest vector space score (see Section 3 for a description of the segmentation procedure).

The highly discriminating terms were selected using a chi-square test. We created a subcollection of the 2000 documents closest to the topic description according to the vector space score (as computed Buckley et al. (1994) [1]). The topic description was Rocchio-expanded before running the selection procedure [1]. Five copies of the (unexpanded) topic were added to the 2000 documents resulting in a subcollection of 2005 items. Terms that occurred at least five times within

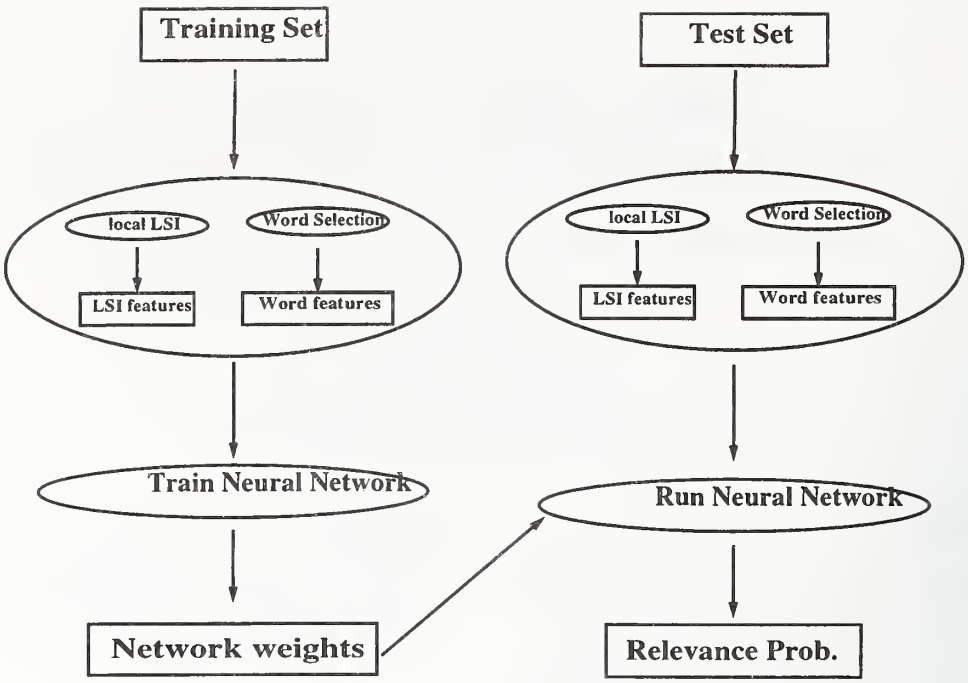


Figure 1: Architecture for routing.

this subcollection were ranked, using the chi-square test, according to how significantly they discriminated relevant from non-relevant documents. The top 100 terms were chosen as the input terms for the classifier.

We settled on chi-square as a selection criterion after initial experiments comparing it with terms selection according to raw frequency of occurrence and according to the ratio of relevant and non-relevant documents a term occurs in. These alternative measures were outperformed by the chi-square test.

Next, the top 1000 terms and the 2000 documents were used to define a 1000-×-2000 matrix that was decomposed using a singular value decomposition. The first 100 components for each document correspond to the second block of input units for the classifier.

For both linear and nonlinear networks, a different network is built for each topic description, and a single output unit indicates whether or not a given document is judged relevant to that network's topic. In the linear architecture, each input unit is connected to the output unit with no hidden units. This is roughly equivalent to logistic regression, although the fitting method differs (maximum likelihood for logistic regression versus backpropagation for the neural net). In the non-linear architecture, the input units are again connected to the output unit, but there is an additional layer of three hidden units between the input units and the output unit. Input units and hidden units are fully connected.

For each topic description, the classifier was trained with backpropagation [10] on two thirds of the 2000 preselected documents. The remaining third was used as a validation set. In order to avoid overfitting, the neural network was trained until the error on the validation set increased. To ensure that the classifier would be applied to a test set with a similar distribution as the training set, we considered only documents with a vector space score above a cutoff.

Table 1 gives results for the two classifiers and three input configurations on the TREC 2 data and TREC 3 data, respectively. For both collections, the best performance is achieved for a combination of terms and local LSI vectors. This result indicates that each of the two modes of representation, terms and local LSI, provides information about relevance that is not present in the other mode.

The results in Table 1 are less consistent with respect to the potential benefit of nonlinearity. The nonlinear classifier outperforms the linear one only in TREC 2 whereas it shows worse performance in TREC 3. A possible explanation is that the number of relevant documents for topics with few relevant documents is smaller in TREC 3 than in TREC 2. For example, 16 topics have less than 100 relevant documents in TREC 3. In contrast, there are only 6 such topics in TREC 2. It seems plausible that the kind of complex generalizations that a nonlinear classifier can learn will only be sound if there is enough training material, and this condition does not seem to hold in TREC 3.

Our routing results can be summarized as follows:

- Of the two document representations, terms and local LSI vectors, each contributes information that

TREC 2		TREC 3		
input	precision average	input	precision average	precision at 100
linear classifier		linear classifier		
terms	0.4253	terms	0.4109	0.473
local LSI	0.3901	local LSI	0.3963	0.473
terms + local LSI	0.4213	terms + local LSI	0.4207	0.484
nonlinear classifier		nonlinear classifier		
terms	0.4260	terms	0.4095	0.474
local LSI	0.3829	local LSI	0.3981	0.474
terms + local LSI	0.4359	terms + local LSI	0.4145	0.482

Table 1: Routing performance for TREC 2 (topics 51-100, 100 selected terms) and TREC 3 (topics 101-150, 200 selected terms). The term "average" refers to non-interpolated average precision. These results were obtained after the official NIST submission for TREC 3.

cannot be extracted from the other.

- There is evidence that a non-linear classifier performs better than a linear classifier if there is enough training information.

3 Ad Hoc

Documents are preprocessed for the ad hoc task by computing thesaurus vectors and segmenting long texts with TextTiling, topic descriptions are subdivided into query factors, and the various attributes (or predictors) are combined using logistic regression. This section describes this process in more detail. Figure 2 sketches the architecture for our ad hoc retrieval system.

3.1 Document Preprocessing

3.1.1 Thesaurus Induction

Our automatically constructed thesaurus maps each term to a vector that represents its pattern of cooccurrences with other terms throughout the document collection. The goal is to capture the similarity between terms as measured by the commonality of their neighbors. The technique is related to Latent Semantic Indexing [4], but relies on cooccurrence of terms rather than occurrence of terms in documents [12].

The thesaurus induction is motivated by a term-by-term matrix in which each entry of the matrix is a count of the number of times that the row term and the column term occur near each other in the corpus (where near is defined as within the same paragraph). To the extent that similar local context is a good measure of topical similarity, closeness in the row (or column) space of this matrix reflects topical similarity among terms. Since the full matrix is unworkably large, we approximate it in a staged computation. First, the matrix for a small subvocabulary of frequent terms is collected and the induced term similarity used to cluster these terms into basis classes. These basis classes are then used to compute a new matrix for an extended vocabulary. The process can be iterated until the full vocabulary is covered. The final matrix is reduced in dimensionality by the application of a singular value decomposition. For more details see [13].

Thesaurus vectors induce a similarity between terms which we use to factor queries. This similarity measure can be extended to sets of terms, and hence to documents, by averaging the included terms' vectors to form context vectors. Context vectors offer a fuzzy (or soft) match criteria for judging the similarity between a query and a document.

3.1.2 Motivated Segmentation

The second preprocessing step is concerned with the treatment of long documents (those comprised of more than two or three paragraphs). In previous work we have developed TextTiling, an algorithm for partitioning full-length text documents into coherent multi-paragraph units, called tiles [6]. These units are meant to represent the subtopic structure of the texts; according to evaluations done with human judges, subtopical discussions often span several paragraphs. TextTiling uses term frequency analyses to determine the extent of the tiles. The result of tiling is a representation of the original document augmented with tile boundary locations.

3.2 Topic Preprocessing: Factorization

A distinctive feature of our approach to the ad hoc task is that we overlay a conjunctive Boolean constraint on top of an additive similarity scoring scheme, which is intrinsically disjunctive in nature. That is, because the underlying scoring scheme is additive, there is no guarantee that a high scoring document will include a high

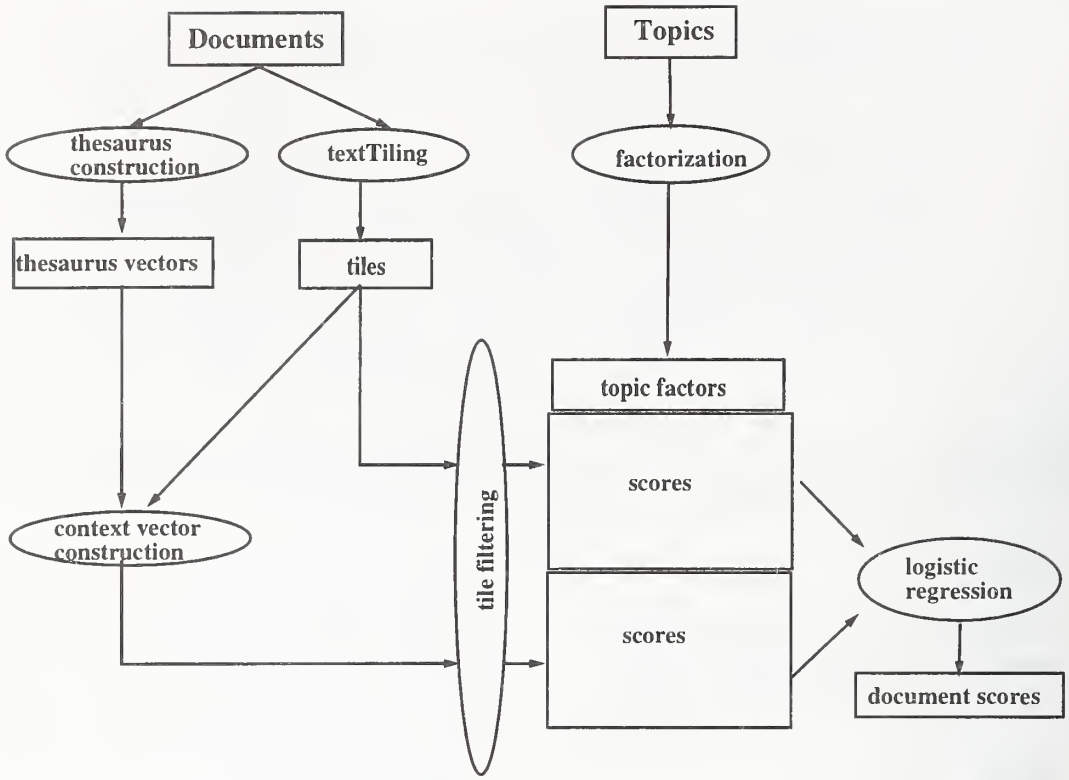


Figure 2: Architecture for ad hoc.

percentage of the terms in the query; it may be sufficient to include only a few with high intensity. This can be counterproductive if the meaning of the query requires disjoint sets of terms all to appear simultaneously. We explored both manual and automatic means to recover appropriate Boolean constraints. However, in the following we concentrate on automatic topic analysis since the manual experiment, described below, was unsuccessful.

Our automatic method, called topic *factorization*, subdivides the topic into a number of orthogonal *factors* or groups of words that are associated with independent themes of the topic. Factorization bears some similarity with extended Boolean retrieval [11] in that it combines Boolean and similarity-based retrieval. In contrast to [11], we induce the Boolean constraint automatically instead of relying on the user to define disjunctions or conjunctions. The key idea is to exploit the information about term relations in the thesaurus. Topics often contain several distinct groups of closely related terms. A case in point is Topic 51, which is about subsidies to the European aircraft industry. Two groups of terms or factors in this topic that are clearly delineated from each other and from other terms are terms related to subsidies and terms related to the aircraft industry. Consider two documents with the same number of terms matching Topic 51, the first with terms that are related to only one of these factors, the second with some terms related to the first factor and others related to the second factor. Intuitively, we want to give preference to the document that covers two of the crucial subthemes of the topic. (The importance of relating the themes of a query to the various subthemes of the documents being retrieved against are also discussed in [8].) In these experiments we attempted to achieve this by giving each document the retrieval status value of the factor that scores the lowest, thus imposing a conjunctive constraint on the query.

To find the factors of a topic automatically, words are represented by their thesaurus vectors and partitioned into a pre-determined number of classes via group-average agglomerative clustering. This step exploits the semantic information in the thesaurus to arrive at coherent groups of terms. Since topic descriptions in the TREC collection have similar length, we use a fixed number of four clusters per topic. For example, here are the four clusters for topic 51:

1. industrie airbus;industrie airbus aircraft;-consortium european;aircraft construcciones;-aeronauticas construcciones aerospatiale aeronauticas aeronauticas;s.a. british;aerospace boeing messerschmitt-boelkow-blohm messerschmitt-boelkow-blohm;gmbh mcdonnell;douglas mcdonnell aircraft;producer aerospace;plc s.a. aerospace aircraft u.s.;aircraft douglas plc producer consor-

input	TREC2		TREC3	
	precision		precision	
	average	at 100	average	at 100
documents	0.3018	0.444	0.2458	0.351
best tile	0.3038	0.452	0.2532	0.354
best tile + phrases	0.3154	0.469	0.2647	0.369
best tile + phrases + factors	0.3192	0.475	0.2718	0.379

Table 2: Scores for ad hoc runs on TREC 2 and TREC 3 data. These results were obtained after the official NIST submission for TREC 3.

tium gmbh

2. government u.s.;government dispute subsidy sanction federal;subsidy government;assistance agreement retaliation finance aid;loan aid dispute;trade loan controversy assistance;aid group

3. trade;dispute european;government trade;tension trade;controversy gatt european airbus;subsidy u.s. spanish;government trade;policy trade;gatt german aircraft;code british trade anti-dumping german;-british french anti-dumping;duty spanish tension french;german

4. duty federal assistance objection petition policy complaint countervail cite countervail;duty duty;-petition tariff loan;finance document narrative code

Although factor 4 is fairly nondescript, the other clusters do correspond to major themes present in the topic (which is about government subsidies to Airbus and the implications for U.S. companies): the aircraft industry (factor 1), government subsidies (factor 2), and international trade (factor 3). The intuition is that a relevant document should score highly against all selected query factors simultaneously. This imposes a conjunctive constraint that would otherwise not be present.

3.3 Combining Predictors

To combine scoring information for factors and tiles we adopted a probabilistic retrieval approach that models the logit of the probability of relevance given the query as a linear function of the available predictors [2, 5]. The model was fit using topics 51–100 as a training set and tested on topics 101–150 (the validation set). Since running the logistic regression on all 750,000-×-50 document-topic pairs is not feasible, we selected a subset of *privileged* tiles: for each topic description, only those tiles with the 2000 highest vector space scores were retained. The privileged tiles were used to select documents (or portions of documents) for further processing in each of the following ways:

- documents that contain at least one privileged tile
- the best privileged tile of a document
- the aggregate (= concatenation) of all privileged tiles of a document

Factors were used in the computation of *minmax* predictors: the minimum score for the n "best" factors of a topic. The "goodness" of factors was assessed either manually or automatically. In the manual method, one of us simply ranked the factors according to his intuition about how important they were relative to each other. In the automatic method, factors were ranked according to "soft" correlation with the center of the thesaurus space, i.e. according to the correlation of the context vector of the factor (the sum of the thesaurus vectors of its terms) and the vector representing the center of the space occupied by the thesaurus. The less descriptive a factor, the closer it is to the center of the thesaurus space. Terms like "outcome" and "attempt", that don't have strong topical characteristics, cluster near the center of thesaurus space. In other words, the more distant a factor's terms were from the thesaurus center, the higher it was ranked. The center of the thesaurus space was computed as the centroid of the 10,000 most frequent terms.

Given the privileged tiles and the ranked factors, the following predictors were computed in the experiments. Except for the first two, each was computed separately for the document as a whole, for the best tile (where best is defined according to the vector space score), and for the aggregate of the best tiles. We also computed these predictors separately for words and phrases.

1. number of tokens in query
2. number of tiles in aggregate
3. number of tokens (in document, tile or aggregate)
4. number of matches with query
5. sum over all term matches: square root document frequency (the occurrence count of that term in the document, tile or aggregate)
6. sum over all term matches: product square root query frequency (the occurrence count of the term in the query) and square root document frequency

pred. prob.	# matches:	2.50	5.50	7.00	9.00	12.00	29.00
63	emp. prob.	0	38	46	40	37	31
	# documents	2	64	254	800	1600	2326
24	emp. prob.	43	26	23	23	20	30
	# documents	51	296	1131	2079	2578	1301
16	emp. prob.	27	21	20	16	15	28
	# documents	177	1033	1876	2777	2632	998
11	emp. prob.	19	18	15	11	9	28
	# documents	605	2169	3290	3127	2007	797
8	emp. prob.	13	12	9	6	6	25
	# documents	2663	5266	4545	2563	1210	536
3	emp. prob.	3	4	3	1	1	11
	# documents	35116	9225	3205	1034	410	257

Table 3: Predicted and actual probability of relevance. Documents are binned according to number of matches and predicted probability. For each cell, the empirical probability and the number of documents in the cell are given.

7. sum over all term matches: product square root document frequency and log collection frequency (the number of documents the term occurs in)

8. minimum of soft correlations with 3 best factors (automatic selection)

9. minimum of soft correlations with 3 best factors (manual selection)

We ran extensive tests to find the optimal combination of predictors over the validation set. In general, it was found that adding predictors to the basic Cooper-style predictors (5–7) did not improve performance. Since many of the predictors are highly correlated (e.g., predictor 5 for tiles and for documents capture similar information), this result is not surprising in many cases. For some combinations we were disappointed by the failure of predictors to make a difference. For example, the manually selected factors did not do better than the automatically selected ones.

Table 2 displays results for predictors 5–7 for documents and separately for best tiles. Best tiles perform slightly better than documents. We found that simply adding predictors 5–7 calculated for phrases instead of words did not improve performance. However, if we fitted models separately for words and phrases and then combined them in a second-stage regression we did achieve better results (as quoted in the table). Finally, if we added predictor 8 to the second-stage regression mentioned above we achieved another improvement in performance.

4 Discussion and Future Work

In analyzing the ad hoc results, we consider three avenues of investigation:

- What do individual predictors contribute to the estimation?
- To what extent does the regression model estimate the probability of relevance correctly?
- What kind of interaction takes place between query factors and tiles?

To investigate the first two questions, we have found it useful to bin the set of documents according to their value on a particular predictor and according to estimated probability.[1] Table 3 gives an example from the "best tile" results of Table 2. The estimated probabilities of all 100,000 documents used in training (2000 for each of the 50 queries), are binned into six bins such that each bin has the same number of relevant documents. Similarly, the same procedure is applied to the number of matches of these documents, i.e. six bins are constructed such that each bin has the same number of relevant documents. The cross-product of the two binnings gives us 36 cells for which we can compute the actual probability of relevance by dividing the number of relevant documents in the cell by the total number of documents in the cell. Table 3 gives for each bin the center of the interval covered by it (probability for rows, number of matches for columns), and, for each cell, the empirical probability of relevance and the number of documents.

Our regression model assumes a linear relationship between predictor and logit of probability. In the lower rows of Table 3, this relationship holds to a good degree of approximation for documents with up to 12 matches. However, the assumption of linearity is grossly inadequate for the "outlier" bin labeled '29.00' (the last column). Since regression analysis is sensitive to outliers,

[1] This method of analysis was suggested to us by John Tukey.

this is an important problem to investigate. It may be the case that for this particular predictor, the model should be modified to include a transformation of the predictor "number of matches" or a special processing step for documents that have high match scores.

As for the second question (accuracy of estimation), the table shows that high probabilities are overestimated (top row) and that low probabilities tend to be underestimated (rows corresponding to predicted probabilities "11" and "8"). The incorrect estimation of the high probabilities is especially problematic since the evaluation measure of average precision depends mainly on the top part of the document ranking. The reason the top part of the ranking is less accurately estimated than the bottom part seems clear: there are many more non-relevant than relevant documents so that the logistic regression will concentrate its "capacity" on the low-probability end. To address this problem we plan to experiment with cost functions that are more closely tied to the evaluation measure.

For the third question, we are in the process of using a new user interface paradigm, called TileBars [7], to assess in more detail how the terms in the query factors are distributed in the longer texts. We suspect that the characteristics of the best subset of tiles to represent a long document's contents will vary with each query; for example, for some topic descriptions the best tile will work well, but for others, the set of tiles with the most query term overlap might be most appropriate.

Acknowledgments. We would like to thank David Hull and John Tukey for discussions of the material presented in this report and Mike Berry for making SVDPACK available to us.

References

[1] Chris Buckley, James Allan, and Gerard Salton. Automatic routing and ad-hoc retrieval using SMART: TREC 2. In Donna Harman, editor, *The Second Text REtrieval Converence (TREC-2)*. U.S. Department of Commerce, Washington DC, 1994. NIST Special Publication 500-215.

[2] William S. Cooper, Fredric C. Gey, and Aitoa Chen. Probabilistic retrieval in the TIPSTER collections: An application of staged logistic regression. In D.K. Harman, editor, *The Second Text REtrieval Converence (TREC-2)*, Washington DC, 1994. U.S. Department of Commerce. NIST Special Publication 500-215.

[3] Douglass R. Cutting, Jan O. Pedersen, and Per-Kristian Halvorsen. An object-oriented architecture for text retrieval. In *Conference Proceedings of RIAO'91, Intelligent Text and Image Handling, Barcelona, Spain*, pages 285–298, April 1991. Also available as Xerox PARC technical report SSL-90-83.

[4] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[5] Norbert Fuhr and U. Pfeifer. Probabilistic information retrieval as a combination of abstraction, inductive learning, and probabilistic assumptions. *ACM TOIS*, 12(1), Jan 1994.

[6] Marti A. Hearst. Multi-paragraph segmentation of expository discourse. In *Proceedings of the 32nd Meeting of the Association for Computational Linguistics*, June 1994.

[7] Marti A. Hearst. Tilebars: Visualization of term distribution information in full text information access. In *To appear in Proceedings of CHI 1995*, Denver, CO, May 1995.

[8] Marti A. Hearst and Christian Plaunt. Subtopic structuring for full-length document access. In *Proceedings of the 16th Annual International ACM/SIGIR Conference*, pages 59–68, Pittsburgh, PA, 1993.

[9] David Hull. Improving text retrieval for the routing problem using latent semantic indexing. In *Proceedings of SIGIR '94*, pages 282–289, 1994.

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In David E. Rumelhart, James L. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing. Explorations in the Microstructure of Cognition. Volume 1: Foundations.* The MIT Press, Cambridge MA, 1986.

[11] Gerard Salton, Edward A. Fox, and Harry Wu. Extended boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036, November 1983.

[12] Hinrich Schütze. Dimensions of meaning. In *Proceedings of Supercomputing '92*, 1992.

[13] Hinrich Schütze and Jan O. Pedersen. A cooccurrence-based thesaurus and two applications to information retrieval. In *Proceedings of RIAO*, pages 266–274, Rockefeller University, New York, 1994.

Document Retrieval and Routing Using the INQUERY System

John Broglio, James P. Callan, W. Bruce Croft and Daniel W. Nachbar
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, Massachusetts 01003, USA

The INQUERY retrieval and routing system, which is based on the Bayesian inference net retrieval model, has been described in a number of papers [5, 4, 10, 11]. In the TREC experiments this year, a number of new techniques were introduced for both the ad-hoc retrieval and routing runs. In addition, experiments with Spanish retrieval were carried out.

1 Description of Ad-Hoc Experiments

For the ad-hoc retrieval experiments, the major changes to the system were the incorporation of passage retrieval, query expansion using PhraseFinder [8], a new estimation technique for indexing probabilities, and new analysis techniques for the TREC topics.

1.1 Query Processing

The following description of query processing emphasizes the differences to the approach used in the previous evaluations, which is described in [5, 3].

The text in all parts of the TREC topics (topic, description, narrative) were treated in the same way. The University of Massachusetts JTAG tagger [12] was used to identify parts of speech for words in the query. Then sequences of nouns, or sequences of adjectives and following nouns, were selected for the #PHRASE operator. (In the past, we had included prepositional phrases which followed a "bare" headnoun. We eliminated these phrases because we found they were not useful.)

Analysis of phrase statistics has shown that phrases of length greater than two words were invariably too restrictive, especially when subphrases of these would have been useful in retrieval. We therefore used only two-word phrases. Sequences of more than two words were broken down into two-word subsequences. For example: *(crude oil price trend)* $\Rightarrow$ *(crude oil), (oil price), (price trend).* Using this modification of the original phrase extraction procedures, we were able to eliminate a number of special processing steps, and apply noun-phrase-extraction processing to all sections.

We weighted the sections of the query as follows; title = $3n$; description = $1n$, narrative = 1; where n is the number of terms in the section. This was accomplished simply by duplicating the title three times, the description once, and by using the #SUM of the narrative terms.

1.2 PhraseFinder

PhraseFinder is a technique for corpus-based query expansion [8]. Phrasal concepts are identified using a part-of-speech tagger, and the text contexts of these concepts (words in close proximity) are used to create concept documents. These documents, in turn, are used to create a concept-based INQUERY database. The query is then processed against this database to produce a ranked list of phrases. Phrases from this list are then added to the original query, with appropriate weighting. In [8], phrases were treated differently depending on whether they contained words that were in the original query ("duplicates") or were entirely new ("non-duplicates"). Previous experiments with TIPSTER/TREC databases and topics, which produced consistent effectiveness improvements, were used to determine the number of phrases and the weighting used.

After query processing created a query q from a topic, the query q was used to retrieve 30 associated phrases from a PhraseFinder database. The retrieved phrases were partioned into a ranked set of phrases already in q, and a ranked set of phrases not in q. Each phrase was weighted according to its rank in the set. "Duplicate" phrases were weighted by $1-(r-1)\cdot\frac{1}{30}$, where r was the phrase's rank. "Non-Duplicate" phrases were weighted by $0.6-(r-1)\cdot\frac{0.6}{30}$. Then the phrases and their weights were added to q to form an expanded query q'.

1.3 Passage Retrieval

In [2], we reported experiments that showed significant improvements in retrieval effectiveness are possible when document rankings based on the entire document text are combined with rankings based on the best passages in the documents. Overlapping passages of a fixed-length yielded better results than passages based on document structure (e.g. paragraphs).

In the ad-hoc document retrieval experiments, the expanded query (q' above) was applied to both documents and passages, as described in [2]. The final form of the query, q'', was

#wsum (1.0 2.0 #passage200 (q'') 1.0 (q''))

based upon the results reported in [2].[1]

1.4 Manual Modifications

These queries were generated by simulating some of the modifications a user might be expected to make to an initial query in an interactive environment. The starting point for this experiment was a version of the automatically produced queries (INQ101). These queries included phrases added automatically using the PhraseFinder as described above. Changes to these initial queries were limited to the deletion of spurious (in the user's opinion) words and phrases, modification of weights based on perceived relative importance, and adding proximity restrictions such as are often used in Boolean systems. The following is an example of the type of query produced.

[1]The first 1.0 is a scaling parameter that is used rarely. It was always 1.0 in our experiments.

```
! Document-level query
#WSUM (1.0 1.0  #SUM( tobacco #company  advertising
                      #PHRASE( young people) tobacco target young
                      #PHRASE( tobacco industry) young youth children
                      teenagers adolescents #phrase(high school students)
                      #phrase(college students) )
           3.0 #PHRASE( tobacco industry)  1.0 aims
           1.0 advertising   1.0 young
           3.0  Tobacco  3.0 company  3.0 advertising   1.0 young
           3.0 #uw100( tobacco advertising #syn(young youth teenagers
                       adolescents students children))
   0.3 #3 ( tobacco industry ) 0.3 #3 ( tobacco company )
   0.3 #3 ( tobacco product ) 0.3 #3 ( ad campaign )
   0.3 minor 0.3 snuff 0.3 #3 ( tv commercial )
   0.3 #3 ( advertising campaign ) 0.3 #3 ( marketing executive )
   0.3 teenager 0.3 glamour 0.3 billboard  0.3 allure )
```

1.5 Estimation

INQUERY, like most statistical systems, relies on a *tf.idf* formula for estimating the probability that a document is about a concept. Unlike many systems, INQUERY starts with a default probability and then adjusts it based on evidence. The formula used in recent TREC and TIPSTER experiments to determine the belief due to the occurrence of query term Q in a document was:

$$\text{bel}_{\text{term}}(Q) = d_b + (1 - d_b) \cdot \left(d_t \cdot H + (1 - d_t) \cdot \frac{\log(tf + 0.5)}{\log(max\_tf + 1.0)} \right) \cdot \frac{\log(\frac{C}{df})}{\log(C)} \qquad (1)$$

where

tf	=	the frequency of term t in the document,
max_tf	=	the frequency of the most frequent term in the document,
df	=	the number of documents in which term t occurs,
C	=	the number of documents in the collection,
H	=	$\begin{cases} 1.0 & \text{if } max\_tf \leq 200 \\ \frac{200}{max\_tf} & \text{otherwise} \end{cases}$,
d_t	=	minimum term frequency component when a term occurs in a document,
d_b	=	minimum belief component when a term occurs in a document.

In this formula, d_b is the belief in a document when there is no evidence ("default belief"). In recent TREC and TIPSTER experiments, d_b and d_t were set to 0.4.

The penalty H was introduced prior to TREC-1 to counter a slight bias towards long documents. This penalty was effective, if not pleasing theoretically. It prevented INQUERY from being biased unduly towards long documents, but still allowed them to be retrieved. (Only four of the TREC-2 systems retrieved more relevant Federal Register documents than did INQUERY [7].)

This year the document retrieval ("ad-hoc") experiments were conducted using a new estimation technique. The estimation experiments involved new forms of concept probability

calculation. This probability is based on the *tf.idf* weights used in many systems, and these experiments incorporated factors such as document length. The penalty H was eliminated, the default belief (d_b) was based partly upon document length, and the default tf component (d_t) was based partly upon the frequency of the term. The adjustments to Equation 1 were:

$$H = 1.0 \tag{2}$$

$$d_t = 0.3 * \left(1.0 - \frac{df}{C}\right)^2 \tag{3}$$

$$d_b = \begin{cases} 0.4 - \frac{L}{1{,}000{,}000} & \text{if } (L < 400{,}000) \\ 0.0 & \text{otherwise} \end{cases} \tag{4}$$

where L = the length of the document (in words).
Our hypothesis was that the estimation technique should be more tolerant of not observing a query term in a short document than in a long document. Likewise, it should be more tolerant of not observing infrequent terms in a document.

These adjustments to the estimation formula were tested against more than just the TREC document collection. In experiments prior to TREC-3, they were found to yield small improvements at all levels of recall on the CACM (2 query sets), West FSupp (2 query sets), and NPL document collections.

2 Description of the Routing Experiments

The routing experiments incorporated two major changes. These were the inclusion of proximity-based features and the use of weights based on Rocchio [1]. Both phrase-level and paragraph-level proximities were considered. This means that significantly co-occurring words in both 5 word windows and 50 word windows were used as features in the automatically constructed queries. The Rocchio weights are based on averaging the *tf.idf* probabilities in both the relevant and non-relevant documents.

The INQ103 query set was created automatically from the original TIPSTER topics (100-150) and the relevance judgements from Volumes 1 and 2. The original queries were expanded using words and phrases from the relevant documents and reweighted.

The new features that were added to the queries were the 32 words with the highest weights in the relevant set, 10 proximities based on 5 word windows, and 20 proximities based on 50 word windows. Features were weighted using an approach based on the Rocchio formula developed for the vector space model [1]. The three primary differences between these weights and the standard Rocchio weights are:

1. The query is a weighted sum of three components,

```
#wsum(1.0 2.0 (original query) 4.0 (new word terms)
      1.0 (new proximity terms))
```

2. Since the top level of the query already contains the original query, the Rocchio computation only includes average weight in the relevant and non-relevant documents;

3. The term weights are computed using the weight functions INQUERY uses to estimate beliefs, and the the normalization uses document length statistics instead of the cosine normalization;

4. Although the entire relevant set is used, only the top-ranked n non-relevant documents are considered, where n is the number of relevant documents for the query, and the ranking is done using the original query.

The "manual" routing experiment (INQ104) was done by a weighted combination of the automatically produced queries with a manually produced set of queries generated for the ad-hoc experiments in the previous TREC experiments. No manual changes were made by examining relevant documents. The automatically produced queries were given twice the weight of the manual queries.

3 Description of the Spanish Experiments

The Spanish retrieval experiments focused on evaluating the effect of morphological processing. Only small modifications were required to apply INQUERY to a collection of Spanish documents. A stopword list was developed manually (1 week), document indexing had to be adjusted (1 week), a Spanish word stemmer had to be developed (5 weeks), and the graphical user interface had to be modified (6 weeks).

The word stemming algorithm is a Porter-stemmer in spirit [9]. Each stage of the Spanish algorithm is somewhat more complex than the English algorithm, because Spanish has more verb conjugation than does English, and the basic algorithm is augmented with a large table of exceptions, because Spanish has more irregular verbs than does English. Decisions about which verbs to handle with exceptions and which to handle with general rules were based upon Spanish verb data supplied by New Mexico State University.

The processing of Spanish topics to queries was accomplished by the same software used for English. Phrase recognition was disabled because we had no Spanish part-of-speech tagger. The stop-phrase heuristics were not translated to Spanish, so stop-phrases were not discarded.

4 Ad-Hoc Results and Discussion

Two sets of results, INQ101 and INQ102, were evaluated in the ad-hoc document retrieval evaluation.

The INQ101 results were based on completely automatic processing of the TREC topic statement into a query, automatic query expansion, use of passage-level and document-level evidence, and adjustments to INQUERY's estimation formula.

INQ102 was a semi-automatic experiment in which a user was allowed to edit the INQ101 query prior to running it. As in the past, the user was restricted to deleting query terms, grouping query terms with proximity operators, adjusting query term weights, and adding query terms from the Narrative. No other kind of modification was permitted.

The official results for INQ101 and INQ102 are summarized below.

Query Type	Average Precision			
	5 Docs	30 Docs	100 Docs	11-Pt Avg
INQ101	.64	.57	.44	.37
INQ102	.74 (+15.5%)	.63 (+11.0%)	.49 (+11.5%)	.42 (+15.5%)

Limited user modification of INQ101 produced a very significant 15.5% improvement in average precision. Most of this improvement appears to be due to 1) deleting useless query terms introduced by query processing or PhraseFinder, and 2) grouping query terms with proximity operators. Clearly there remains room for improvement in automatic query processing techniques.

After TREC-3, experiments were done to investigate the effect of the changes made this year. Our simple query processing improved precision very significantly at almost all levels of recall, as shown in the table below. The average precision obtained with query processing was 84.2% higher than the average precision obtained by using raw words alone.

Recall	Precision (50 queries)						
	Raw Words (RW)	RW W/O Stop & Not Phrases (SN)		SN With Field Weights (FW)		FW With #PHRASE op (PO)	
0	45.3	61.8	(+36.3)	78.8	(+74.0)	79.6	(+75.8)
10	28.2	40.1	(+41.9)	50.1	(+77.5)	57.0	(+102.0)
20	24.3	33.1	(+36.2)	40.3	(+66.2)	46.6	(+92.0)
30	21.6	29.7	(+37.3)	33.7	(+55.6)	40.1	(+85.5)
40	18.0	24.9	(+38.7)	28.0	(+55.6)	33.8	(+88.1)
50	15.7	21.3	(+35.7)	23.9	(+52.0)	27.7	(+76.5)
60	13.1	18.0	(+38.0)	20.3	(+55.8)	22.7	(+73.7)
70	10.6	14.7	(+37.8)	17.1	(+60.4)	18.4	(+72.8)
80	7.9	11.3	(+43.8)	13.1	(+66.9)	13.3	(+68.7)
90	4.5	7.2	(+60.3)	9.0	(+101.3)	9.4	(+110.4)
100	0.5	0.8	(+60.9)	0.9	(+99.8)	0.7	(+47.1)
avg	17.2	23.9	(+38.6)	28.7	(+66.3)	31.8	(+84.2)

Query expansion with PhraseFinder proved to be a good idea. As the table below shows, the inclusion of PhraseFinder terms in the query yielded a 9.6% improvement in average precision. Passage retrieval was also a good idea. Combining passage-level and document-level evidence produced a 15.7% improvement in average precision.

Combining PhraseFinder query expansion and passage retrieval led to the best performance of all, a 19.8% improvement over the baseline query processing. The improvement of the combination over passage retrieval alone was smaller than expected, suggesting that their are better ways of combining these two sources of evidence.

Recall	Precision (50 queries)						
	Query Processing (QP)	QP With PhraseFinder (PF)		QP With Passage (PS)		QP and PF and PS (All)	
0	79.6	79.4	(−0.2)	83.8	(+5.2)	80.6	(+1.2)
10	57.0	59.2	(+3.9)	61.7	(+8.1)	61.4	(+7.6)
20	46.6	50.3	(+8.0)	54.9	(+17.7)	54.1	(+16.1)
30	40.1	44.4	(+10.7)	47.2	(+17.6)	48.4	(+20.7)
40	33.8	37.4	(+10.7)	41.3	(+22.0)	43.0	(+27.1)
50	27.7	32.7	(+18.1)	34.0	(+22.8)	37.0	(+33.4)
60	22.7	27.6	(+21.6)	28.4	(+25.4)	31.6	(+39.4)
70	18.4	22.2	(+20.7)	23.2	(+26.5)	26.6	(+44.7)
80	13.3	16.8	(+26.7)	17.1	(+29.0)	20.6	(+55.5)
90	9.4	11.6	(+22.6)	11.6	(+22.7)	13.4	(+42.0)
100	0.7	1.2	(+73.5)	1.2	(+74.2)	1.9	(+177.0)
avg	31.8	34.8	(+9.6)	36.8	(+15.7)	38.1	(+19.8)

The change to the estimation formula, described above, had no significant effect on INQUERY's performance in the ad-hoc document retrieval evaluation. There was a +1.3% improvement in average precision over 11 recall points, but most of the improvement occurred at high (> 40%) recall.

5 Routing Results and Discussion

Two sets of results, INQ103 and INQ104, were evaluated in the document routing evaluation. The INQ103 results were based on completely automatic processing of the TREC topic statement and relevance judgements into a query. INQ104 was a combination of the INQ103 query and a query modified manually. The official evaluations are summarized below.

Query Type	**Average Precision**			
	5 Docs	**30 Docs**	**100 Docs**	**11-Pt Avg**
INQ103	.67	.58	.46	.38
INQ104	.62 (−1.9%)	.56 (−2.4%)	.45 (−3.0%)	.39 (+1.1%)

As in previous TREC experiments, the combination of automatic and manually-modified query sets was not helpful. This result is not surprising, given the improved weighting in INQ103 and the mediocre performance of the manually-modified query set in last year's TREC evaluation.

6 Spanish Results and Discussion

Two sets of results, SIN001 and SIN002, were evaluated in the ad-hoc Spanish document retrieval evaluation.

The SIN002 results were based on completely automatic processing of the TREC topic statement into a query. The processing performed was a subset of the processing applied

to English topics. For example, there was no noun-phrase recognition, and no stop-phrase removal. The various fields of the topic statement were weighted as in English.

SIN001 was a semi-automatic experiment in which a user was allowed to edit the SIN002 query prior to running it. As with English, the user was restricted to deleting query terms, grouping query terms with proximity operators, adjusting query term weights, and adding query terms from the Narrative. No other kind of modification was permitted.

The official evaluations are summarized below.

Query Type	**Average Precision**			
	5 Docs	**30 Docs**	**100 Docs**	**11-Pt Avg**
SIN002	.74	.69	.60	.51
SIN001	.78 (+5.4%)	.67 (−2.7%)	.53 (−11.6%)	.42 (−17.5%)

Limited user modification of SIN002 produced improvements at very low recall, but a significant 17.5% loss in average precision. The relative inexperience of the user making the changes may have been a factor.

One concern was whether the Spanish word stemmer, which was developed in about five weeks, helped or hindered performance. The table below shows that it helped, providing a 9.4% improvement in average precision on the SIN002 query set.

Although the results for Spanish ad-hoc document retrieval appear reasonable, there is much room for improvement. A Spanish part-of-speech tagger would enable more sophisticated query processing, as would a Spanish stop-phrase recognizer. In principle, there is no reason why the complete English language query processing and query expansion techniques can't also be applied to Spanish. However, one practical near-term obstacle will likely be retrieval of large numbers of unjudged documents, due to the small number of systems participating in the Spanish evaluation this year.

Recall	Precision (25 queries)		
	SIN002 Unstemmed	SIN002 Stemmed	
0	86.5	86.9	(+0.4)
10	71.7	75.2	(+4.8)
20	65.3	69.6	(+6.7)
30	59.8	64.8	(+8.2)
40	54.5	60.4	(+10.9)
50	48.1	54.2	(+12.8)
60	42.7	49.1	(+15.0)
70	37.2	42.3	(+13.6)
80	29.2	33.0	(+13.1)
90	16.4	22.7	(+38.7)
100	3.5	5.1	(+44.4)
avg	46.8	51.2	(+9.4)

7 Summary

The themes guiding our work this year remain unchanged from previous years: We believe in highly structured queries, sophisticated query processing, and in combining multiple sources

of evidence. The evaluation this year showed continued progress on each of these fronts.

INQUERY's heuristics for processing TIPSTER topics into queries continue to evolve and improve, providing a relatively high baseline from which to start considering other improvements. The ad-hoc document retrieval queries have become increasingly structured. Although no additional structure was introduced into the routing queries, their weights were improved by a switch to Rocchio weighting.

Two additional sources of evidence were introduced into the ad-hoc document retrieval queries this year: passage-level evidence, and query expansion using PhraseFinder. These provided significant improvements over the baseline query processing methods alone, and probably compensated to some extent for the loss of the "Concepts" field in the TREC topics.

The value of the Spanish evaluation was in showing how quickly INQUERY can be applied to new languages. Previous efforts have demonstrated that INQUERY can be used with Japanese [6] and Finnish, but they required significant effort. The Spanish effort required about seven weeks of work on the retrieval system, most of it to develop a Spanish word-stemmer.

Acknowledgements

We thank Lisa Ballesteros, Paul Booth, Stephen Harding, Tom Kalt, Zhihong Lu, and Jay Ponte for their assistance in the work described here. This research was partially supported by the NSF Center for Intelligent Information Retrieval at the University of Massachusetts, Amherst.

References

[1] C. Buckley, J. Allan, and G. Salton. Automatic routing and ad-hoc retrieval using SMART: TREC-2. In D. Harman, editor, *Proceedings of the Second Text REtrieval Conference (TREC-2)*. National Institute of Standards and Technology Special Publication 500-215, 1994.

[2] J. P. Callan. Passage-level evidence in document retrieval. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 302–310, Dublin, Ireland, 1994. Association for Computing Machinery.

[3] J. P. Callan and W. B. Croft. An evaluation of query processing strategies using the TIPSTER collection. In R. Korfhage, E. Rasmussen, and P. Willett, editors, *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 347–356, Pittsburgh, PA, June 1993. Association for Computing Machinery.

[4] J. P. Callan, W. B. Croft, and S. M. Harding. The INQUERY retrieval system. In *Proceedings of the Third International Conference on Database and Expert Systems Applications*, pages 78–83, Valencia, Spain, 1992. Springer-Verlag.

[5] W. B. Croft, J. Callan¡, and J. Broglio. TREC-2 routing and ad-hoc retrieval evaluation using the INQUERY system. In D. Harman, editor, *Proceedings of the Second Text REtrieval Conference (TREC-2)*. National Institute of Standards and Technology Special Publication 500-215, 1994.

[6] Hideo Fujii and W. B. Croft. A comparison of indexing techniques for Japanese text retrieval. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 237–246, 1993.

[7] D. Harman, editor. *The Second Text REtrieval Conference (TREC2)*. National Institute of Standards and Technology Special Publication 500-215, Gaithersburg, MD, 1994.

[8] Y. Jing and W. B. Croft. An association thesaurus for information retrieval. In *RIAO 4 Conference Proceedings*, New York, October 1994.

[9] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, July 1980.

[10] Howard Turtle and W. Bruce Croft. Inference networks for document retrieval. In Jean-Luc Vidick, editor, *Proceedings of the 13th International Conference on Research and Development in Information Retrieval*, pages 1–24. ACM, September 1990.

[11] Howard R. Turtle and W. Bruce Croft. Efficient probabilistic inference for text retrieval. In *RIAO 3 Conference Proceedings*, pages 644–661, Barcelona, Spain, April 1991.

[12] J. Xu, J. Broglio, and W. B. Croft. The design and implementation of a part of speech tagger for english. Technical Report IR-52, Center for Intelligent Information Retrieval, Department of Computer Science, University of Massachusetts, 1994.

NATURAL LANGUAGE INFORMATION RETRIEVAL: TREC-3 REPORT

Tomek Strzalkowski, Jose Perez Carballo and Mihnea Marinescu
Courant Institute of Mathematical Sciences
New York University
715 Broadway, rm. 704
New York, NY 10003
tomek@cs.nyu.edu

ABSTRACT

In this paper we report on the recent developments in NYU's natural language information retrieval system, especially as related to the 3rd Text Retrieval Conference (TREC-3). The main characteristic of this system is the use of advanced natural language processing to enhance the effectiveness of term-based document retrieval. The system is designed around a traditional statistical backbone consisting of the indexer module, which builds inverted index files from pre-processed documents, and a retrieval engine which 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-3 effort, the total of 3.3 GBytes of text articles have been processed (Tipster disks 1 through 3), including material from the Wall Street Journal, the Associated Press newswire, the Federal Register, Ziff Communications's Computer Library, Department of Energy abstracts, U.S. Patents and the San Jose Mercury News, totaling more than 500 million words of English. Since the TREC-2 conference, many components of the system have been redesigned to facilitate its scalability to deal with ever increasing amounts of data. In particular, a randomized index-splitting mechanism has been installed which allows the system to create a number of smaller indexes that can be independently and efficiently searched.

INTRODUCTION

A typical (full-text) 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 preprocessed user 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.

In term-based representation, a document (as well as a query) is transformed into a collection of weighted terms, derived directly from the document text or indirectly through thesauri or domain maps. The representation is anchored on these terms, and thus their careful selection is critical. Since each unique term can be thought to add a new dimensionality to the representation, it is equally critical to weigh them properly against one another so that the document is placed at the correct position in the N-dimensional term space. Our goal here is to have the documents on the same topic placed close together, while those on different topics placed sufficiently apart. Unfortunately, we often do not know how to compute terms weights. The statistical weighting formulas, based on terms distribution within the database, such as *tf.idf*, are far from optimal, and the assumptions of term independence which are routinely made are false in most cases. This situation is even worse when single-word terms are intermixed with phrasal terms and the term independence becomes harder to justify.

The simplest word-based representations of content, while relatively better understood, 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 the 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-3 data.

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". After the final query is constructed, the database search follows, and a ranked list of documents is returned.

There are several deviations from the above scheme in the system that has been actually used in TREC-3, as well as some important changes from TREC-2. First and foremost, we have 'graduated' from the category B (exploratory systems, about 1/4 of text

data) to the category A (full participation) mostly thanks to significant efficiency improvements in the NLP module. In particular, the BBN's part-of-speech tagger, which we use to preprocess the input before parsing, has been redesigned in time for TREC-3 so that it now adds no more than 5% overhead to the parsing time. We have also installed a new, more efficient version of NIST's PRISE system which cut the indexing time from days to hours. In order to keep memory usage within the limits of our resources, as well as to prepare the system to deal with practically unlimited amounts of data in the future, we devised a randomized index splitting mechanism which creates not one but several balanced sub-indexes. These sub-indexes can be searched independently and the results can be merged meaningfully into a single ranking. Finally, while the query expansion via the domain map is an important part of our system, it has not been used in TREC-3 runs. Our analysis of TREC-2 results revealed several problems with the query expansion scheme and we were in process of redesigning it, however, we were unable to test the revised approach in time for this evaluation, and thus decided to leave it out of TREC-3. We plan to have it in place for TREC-4.

Before we proceed to discuss the particulars of our system we would like to note 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 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. When parsing the TREC-3 collection of more than 500 million words, we found that the parser's speed averaged between 0.17 and 0.26 seconds per sentence, or up to 80 words per second, on a Sun's SparcStation10. 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).

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. [1] 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. Below is a small example of text before and after stemming.

[1] 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.

While serving in South Vietnam, a number of U.S. Soldiers were reported as having been exposed to the defoliant Agent Orange. The issue is veterans entitlement, or the awarding of monetary compensation and/or medical assistance for physical damages caused by Agent Orange.

serve south vietnam number u.s. soldier expose defoliant agent orange veteran entitle award monetary compensate medical assist physical damage agent orange

Please note that proper names, such as South Vietnam and Agent Orange are identified separately through the name extraction process described below. Note also that various "stopwords" (e.g., prepositions, conjunctions, articles, etc.) are removed from text.

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 unable to verify their effectiveness in indexing, mostly because of the tight schedule.

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*

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

```
[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.

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 192:

> References to the cost of cleanup and number of people and equipment involved without mentioning the method are not relevant.

The corresponding compound terms are:

> NOT cost cleanup
> NOT number equip
> NOT number people

Note that while this statement is negated, the negation is conditioned with the *without mentioning ...* phrase. Our NLP module is not able to represent such fine distinctions at this time.

NOMINAL COMPOUNDS

The notorious ambiguity of nominal compounds remains a serious difficulty in obtaining head-modifier pairs of highest accuracy. In order to cope with this, the pair extractor looks at the distribution statistics of the compound terms to decide whether the association between any two words (nouns and adjectives) in a noun phrase is both syntactically valid and semantically significant. For example, we may accept *language+natural* and *processing+language* from *natural language processing* as correct, however, *case+trading* would make a mediocre term when extracted from *insider trading case*. On the other hand, it is important to extract *trading+insider* to be able to match documents containing phrases *insider trading sanctions act* or *insider trading activity*. Phrasal terms are extracted in two phases. In the first phase, only unambiguous head-modifier pairs are generated, while all structurally ambiguous noun phrases are passed to the second phase "as is". In the second phase, the distributional statistics gathered in the first phase are used to predict the strength of alternative modifier-modified links within ambiguous phrases. For example, we may have multiple unambiguous occurrences of *insider trading*, while very few of *trading case*. At the same time, there are numerous phrases such as *insider trading case, insider trading legislation*, etc., where the pair *insider trading* remains stable while the other elements get changed, and significantly fewer cases where, say, *trading case* is constant and the other words change.

The disambiguation procedure is performed after the first phrase extraction pass in which all unambiguous pairs (noun+noun and noun+adjective) and all ambiguous noun phrases are extracted. Any nominal string consisting of three or more words of which at least two are nouns is deemed structurally ambiguous. In the Tipster corpus, about 80% of all ambiguous nominals were of length 3 (usually 2 nouns and an adjective), 19% were of length 4, and only 1% were of length 5 or more. The algorithm proceeds in three steps, as follows:

(1) *Assign scores* to each of the candidate pairs x_i+x_j where $i>j$ from the ambiguous noun phrase $x_1 \cdots x_n$. The score assigned to a candidate pair is the sum of the scores for each occurrence of this pair in any compound nominal within the training corpus. For each occurrence, the score is maximum when the words x_i and x_j are the only words in the phrase, i.e., we have unambiguous nominal $x_j x_i$, in which case the score is 1. For longer phrases, for non-adjacent words, and for pairs anchored at words toward the left of the compound, the score decreases proportionately.

(2) For each set $X_j=\{x_i+x_j \mid \text{for } i>j\}$ of candidate pairs *rank* alternative pairs by their scores.

(3) *Disambiguate* by selecting the top choice from each set such that its score is above an empirically established global threshold, it is significantly higher than the second best choice from the set, and it is not significantly lower than the scores of pairs selected from other sets X_i.

The effectiveness of this algorithm can be measured in terms of recall (the proportion of all valid head+modifier pairs extracted from ambiguous nominals), and precision (the proportion of valid pairs among those extracted). The evaluation was done on a small sample of randomly selected phrases, and the algorithm performance was compared to manually selected correct pairs. The following numbers were recorded: recall 66% to 71%; precision 88% to 91%, depending on the size of the training sample. In terms of the total number of pairs extracted unambiguously from the parsed text (i.e., those obtained by the procedure described in the previous section), the disambiguation step recovers an additional 10% to 15% of pairs, all of which were previously thrown out as unrecoverable. A sample set of ambiguous phrases and extracted head+modifier pairs is shown in Table 1.

Ambiguous nominal	Extracted pairs
oil import fee	oil import import fee
croatian wartime cabinet	croatian cabinet wartime cabinet
national enviromental watchdog group	national group enviromental group watchdog group
current export subsidy program	current program export subsidy subsidy program
gas operating and maintaining expenses	**gas operating operating expenses maintaining expenses

Table 1. Ambiguous nominals and extracted pairs.

EXTRACTING PROPER NAMES

Proper names, of people, places, events, organizations, etc., are often critical in deciding relevance of a document. Since names are traditionally capitalized in English text, spotting them is relatively easy, most of the time. Many names are composed of more than a single word, in which case all words that make up the name are capitalized, except for prepositions and such, e.g., *The United States of America*. It is important that all names recognized in text, including those made up of multiple words, e.g., *South Africa* or *Social Security*, are represented as tokens, and not broken into single words, e.g., *South* and *Africa*, which may turn out to be different names altogether by themselves. On the other hand, we need to make sure that variants of the same name are indeed recognized as such, e.g., *U.S. President Bill Clinton* and *President Clinton*, with a degree of confidence. One simple method, which we use in our system, is to represent a compound name dually, as a compound token and as a set of single-word terms. This way, if a corresponding full name variant cannot be found in a document, its component words matches can still add to the document score. A more accurate, but arguably more expensive method would be to use a substring comparison procedure to recognize variants before matching.

In our system names are identified by the parser, and then represented as strings, e.g., *south+africa*. The name recognition procedure is extremely simple, in fact little more than the scanning of successive words labeled as proper names by the tagger (*np* and *nps* tags). Single-word names are processed just like ordinary words, except for the stemming which is not applied to them. We also made no effort to assign names to categories, e.g., people, companies, places, etc., a classification which is useful for certain types of queries (e.g., *To be relevant a document must identify a specific generic drug company*). A more advanced recognizer is planned for TREC-4 evaluation. In the TREC-3 database, compound names make up about 8% of all terms generated. A small sample of compound names extracted is listed below:

right+wing+christian+fundamentalism
u.s+constitution
gun+control+legislation
national+railroad+transportation+corporation
superfund+hazardous+waste+cleanup+programme
u.s+government
united+states
exxon+valdez
dow_corning+corporation
chairman+julius+d+winer
new+york
wall+street+journal
mcdonnell+douglas+corp+brad+beaver
soviet+georgia
rebel+leader+savimbi
plo+leader+arafat
suzuki+samurai+soft_top+4wd
honda+civic
richard+j+rosebery
mr+rosebery
international+business+machine+corp
cytomegalovirus+retinitis
ids+financial+service+analyst+g+michael+kennedy
senate+judiciary+committee
first+fidelity+bank+n.a+south+jersey
eastern+u.s
federal+national+mortgage+association
canadian+airline+international

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.[2]

To cluster terms into similarity classes, we used a (revised) variant of weighted Jaccard'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 2.

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, we used $M \geq 5$.[3] 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 (GEW) and frequencies with *aerospace* and *pharmaceutical*, respectively):[4]

CONTEXT	GEW	freq/*aerospace*	freq/*pharmaceutical*
firm	0.58	9	22
industry	0.51	84	56
sector	0.61	5	9
concern	0.50	130	115
analyst	0.62	23	8
division	0.53	36	28
giant	0.62	15	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.[5]

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).[6] 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

[2] It would not be appropriate to predict similarity between *language* and *logarithm* on the basis of their co-occurrence with *natural*.

[3] 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*.

[4] 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).

[5] 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).

[6] 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.

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.[7] 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.[8] For example, the following were obtained 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*.

Query expansion is an important part of our system, but it wasn't used in TREC-3, mostly because we were in process of redesigning it. Following TREC-2 we found various problems with both the term clustering procedure as well as with the way the cluster were used to add new terms to queries. Details of these finding are discussed in TREC-2 final report.

CREATING AN INDEX

The limited amount of resources that we had available for indexing forced us to devise a method that splits the collection randomly and produces several sub-indexes. This method would allow us now to index even larger collections in reasonable times. The preliminary tests that we carried out in order to compare the performance of systems where the collection is split into N sub-indexes, for different values of N, suggest that a collection can be split into at least 7 sub-indexes without seeing any degradation in the performance. Given the results that we obtained from such tests as well as the fact that the tests were carried out using relatively small collections (about 150 Megabytes) we intend to perform more extensive testing as soon as possible.

One of the problems we had to face for TREC-1 and TREC-2 was that we did not have enough real memory to index the complete collection (category A) in a reasonable time . Even indexing only the collection for category B (550 megabytes for the ad-hoc experiments) used to take 2 weeks, or about 330 hours. This was more slow than the times that could be obtained by other versions of the PRISE system that were already available by that time. We used a slower version because we did not have then enough main memory to use the faster one. The faster version grows the word frequency tree in main

[7] 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.

[8] For TREC-2 we used $\sigma = 0.2$; δ varied between 10 and 100.

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 2. Selected clusters obtained from syntactic contexts, derived from approx. 40 million words of WSJ text, with weighted Tanimoto formula.

memory, and it is the physical memory that matters here, not the virtual memory, since a tree larger than the size of the real memory causes so many page faults that performance becomes unacceptably slow.

The version of the PRISE system that we used for TREC-3 is much faster than previous versions. According to the on-line documentation provided by NIST the old system would take about 67 hours to index 276 Megabytes of WSJ material while the new system takes less than 2 hours to index the same material. Still, we did not have enough main memory to use the new system to index the complete collection. Our solution to this problem was to split the collection into N sets of almost equal number of documents and create a separate sub-index for each set. In order to keep the N sub-indexes balanced with respect to each other (so that the term idfs are comparable across sub-indexes, for example) we split the collection randomly into N sets. This is done by assigning each document to one of the N sets selected at random. Our goal was to build N sets that would be as homogeneous as possible. At retrieval time the same query is submitted to each one of the sub-indexes and a separate list of ranked documents is obtained for each index. Since we expect idfs to be comparable across sub-indexes, it makes sense to compare the scores of documents belonging to different sub-indexes. The result of the query is then the set of documents with the highest scores chosen from the union of all lists of ranked documents.

In order to evaluate this technique we ran a series of experiments involving about 50000 records. We split that collection into N sets for several values of N (from 1 to 7) and made some measurements of parameters that we expected to be indicators of the degree of homogeneity (e.g., standard deviation of the total number of terms per index, standard deviation of the maximum idf, standard deviation of the number of unique terms, and others). As expected, these indicators showed a decreasing level of homogeneity as N grows larger. This information is summarized in Table 3.

For each value of N, we evaluated the performance of the system using a series of queries for which NIST had provided relevance judgments. For the weighting scheme we were using, and the small collection used for these preliminary experiments, we observed that the performance actually peaks at N = 4 (the average precision when N was 4 was about 7% better than when N was 1). We thought that these results were promising enough to justify the use of the technique described in order to index the complete collection but we intend to perform a much more careful and complete series of experiments as soon as the time and the resources are available. Table 4 summarizes the system's performance at various levels of index split with a subset of AP subcollection.

No. of indexes	Max-Mem MB	Max-idf %std	Uniq.terms Mean	Uniq.terms %std
1	81.9	0.000	921253	0.000
2	61.2	0.424	600869	1.006
3	54.7	0.902	438992	11.678
4	48.2	0.555	373249	3.095
5	46.0	0.986	314986	6.356
6	44.1	1.080	279261	7.318
7	46.8	2.432	247606	16.475

Table 3. Statistics of index splitting performed on a subset of Tipster AP88 subcollection consisting of 48,770 records (about 230 MBytes).

No. of indexes	Avg Prec %change	R-Prec %change	Recall %change
1	0.00	0.00	0.00
2	+4.04	+1.85	+1.11
3	+4.63	+0.72	+0.81
4	+7.04	+4.53	+2.59
5	+1.68	+4.08	+3.92
6	+5.68	+2.75	+4.29
7	+4.18	+4.45	+4.36

Table 4. Performance statistics for split index performed on a subset of Tipster AP88 subcollection consisting of 48,770 records (about 230 MBytes).

For TREC-3 we used 4 sub-indexes for the ad-hoc experiments (2200 Megabytes) and 2 for the routing part (1100 Megabytes). We chose these numbers because, in each case, it was the smallest number of sub-indexes that we could handle given our resources. A nice side-effect of this technique is that each index can be created in parallel on a different machine, making the total time required even shorter. The parameters of the 4-way split used in indexing the TREC-3 ad-hoc database are listed in Table 5.

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

Index No.	Postings MB	Dict. MB	Max-idf	Records	Uniq.terms
1	128.82	72.91	18.509	186557	2928737
2	129.34	72.26	18.492	184460	2909249
3	128.99	72.82	18.499	185297	2931791
4	128.27	71.26	18.498	185114	2874007

Table 5. Statistics of the 4-way split index created for ad-hoc database from Tipster Disks 1 and 2 (about 2 GBytes).

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.[9]

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.

[9] 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.

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.

Schematically, these new weights for phrasal and highly specific terms are obtained using the following formula, while weights for most of the single-word terms remain unchanged:

$$weight(T_i)=(C_1*log(tf)+C_2*\alpha(N,i))*idf$$

In the above, $\alpha(N,i)$ is 1 for $i<N$ and is 0 otherwise. The selection of a weighting formula was partly constrained by the fact that document-length-normalized *tf* weights were precomputed at the indexing stage and could not be altered without re-indexing of the entire database. The intuitive interpretation of the $\alpha(N,i)$ factor is given in the following section.

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. This weighting scheme was again used in TREC-3 runs.

'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 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:

Log(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). The 'hot spot' weighting is represented with the α factor in the term weighting formula given in the previous section.

SUMMARY OF RESULTS

We have processed the total of 3334 MBytes of text during TREC-3. The first 2162 MBytes were data from the Tipster/TREC disks 1 and 2 of which 550 Mbytes (Wall Street Journal subcollection) were previously processed for TREC-2; however, even this portion had to be partially reprocessed. The entire process (tagging, parsing, phrase and name extraction) took about 45 minutes per Megabyte, or just over 2 months on a Sun's SparcStations 10 (at times using an additional Sparc-2). Building a 4-way split index took about 0.6 minutes per Megabyte, or about 21 hours on the Sparc10. The final index size, including postings files and term dictionaries was 804 MBytes, and included approximately 2.9 million unique terms in each sub-index (that's single-word terms, syntactic word pairs and compound names) or nearly 16 (unique) terms per document.

The remaining 1172 MBytes were documents from the Tipster/TREC disk 3 of which about 300 Mbytes of San Jose Mercury articles were previously processed for TREC-2. This portion of the corpus was used to create the routing database. Natural language processing of this part required about 4 weeks on SparcStation 10, and about 10 hours of indexing time. The final size of the index was 428 MBytes, split into 2 sub-indexes of about 214 MBytes each. Each sub-index contained about 3.2 million unique terms, or more than 19 unique terms per record.

Note that in both cases the index size was at 37% of the initial size of the corpus. Given that the natural language processing has added an average 30% to the size of the input (i.e., for each Megabyte of text we obtained about 1.3 Megabytes of terms), the indexer compression ratio was actually 29%.[10]

Two types of retrieval have been done: (1) new topics 151-200 were run in the ad-hoc mode against the Disk-1&2 database, and (2) topics 101-150, previously used in TREC-2, were run in the routing mode against the Disk-3 database. In each category 2 official runs were performed, all fully automatic, with different set up of system's parameters. These runs were labeled *nyuir1* and *nyuir2*. The second run in the routing category includes an experimental use an automatic feedback program which uses the known relevance judgements for topics 101-150 with respect TREC-2 database, to automatically expand the search queries. Summary statistics for these runs are shown in Tables 6 and 7. We note that there is a significant (20%) improvement in precision, as well as a visible increase in recall over the base statistical run when phrasal terms are used. The increase is smaller for routing runs because the routing queries already contained manually prepared concepts fields (*<con>*). We also note the robust improvement of routing results when massive query expansion is performed based on the known relevance judgements for these queries with respect to the training database.

An example ad-hoc topic is shown below:

<top>
<num> Number: 189

<title> Topic: Real Motives for Murder

<desc> Description:
Document must identify a murderer's motive for killing a person or persons in a true case.

<narr> Narrative:
Most relevant would be a description of an intentional murder with a statement of the murderer's motive. An unintentional murder, such as in a charge of second-degree homicide, would be relevant if a motive is stated for an action which clearly led to the victim's death.
</top>

[10] This may be somewhat misleading, since many of the compound terms added by NLP were singletons which take little index space. The unprocessed text compression ratio may in fact be closer to 37%.

The reader familiar with previous TREC evaluations may notice that this query lacks the manually derived <con> field which listed important concepts relevant to the topic, some of which had not even occurred in the actual query. Performance comparison in database search performed during TREC-2 showed that search queries built from the concepts fields outperformed the queries based on narrative sections of the topics by as much as 25% to 30% in precision and up to 10% in recall. Nonetheless, it was felt that the results obtained with the use of the <con> field in the queries did not reflect accurately the capabilities of automatic IR systems in dealing with unprocessed input, therefore the field was dropped in TREC-3.

The table below shows the search query obtained from Topic 189 above with respect to one of the 4 sub-indexes making up the ad-hoc database. Note that the terms extracted from <desc> field are weighted doubly.

Query 189

term = motive+murder	idf = 18.509256	weight = 2
term = motive+murder	idf = 18.509256	weight = 1
term = murder+intentional	idf = 18.509256	weight = 1
term = state+motive	idf = 17.509256	weight = 1
term = death+victim	idf = 15.509257	weight = 1
term = motive+real	idf = 15.509257	weight = 1
term = motive+real	idf = 15.509257	weight = 1
term = charge+homicide	idf = 14.339332	weight = 1
term = unintentional	idf = 12.727898	weight = 1
term = kill+person	idf = 12.033524	weight = 2
term = kill+person	idf = 12.033524	weight = 1
term = second+degree	idf = 11.299804	weight = 1
term = homicide	idf = 10.531977	weight = 1
term = intentional	idf = 9.724622	weight = 1
term = motive	idf = 8.484118	weight = 1
term = motive	idf = 8.484118	weight = 1
term = motive	idf = 8.484118	weight = 1
term = murder	idf = 7.294331	weight = 2
term = murder	idf = 7.294331	weight = 1
term = murder	idf = 7.294331	weight = 1
term = murder	idf = 7.294331	weight = 1
term = victim	idf = 6.775394	weight = 1
term = true	idf = 6.400079	weight = 1
term = degree	idf = 5.974225	weight = 1
term = death	idf = 5.846589	weight = 1

Note that many 'function' words have been removed from the query, e.g., *must*, *identify*, 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., *true* and *degree*, because these could not be uniformly considered as 'common' across all queries.

Run Name	base ad-hoc	nyuir1 ad-hoc	nyuir2 ad-hoc
Queries	50	50	50
Tot number of docs over all queries			
Ret	50000	50000	50000
Rel	9805	9805	9805
RelRet	5398	5978	5978
%chg		+11.0	+11.0
Recall			
0.00	0.6710	0.7653	0.7639
0.10	0.4444	0.5420	0.5429
0.20	0.3784	0.4465	0.4523
0.30	0.3298	0.3763	0.3814
0.40	0.2821	0.3271	0.3281
0.50	0.2274	0.2608	0.2613
0.60	0.1684	0.2031	0.2033
0.70	0.1112	0.1522	0.1519
0.80	0.0699	0.0990	0.0989
0.90	0.0147	0.0339	0.0332
1.00	0.0000	0.0029	0.0029
Average precision over all rel docs			
Avg	0.2271	0.2722	0.2735
%chg		+20.0	+20.0
Precision at			
5 docs	0.5160	0.5960	0.5880
10 docs	0.4680	0.5480	0.5580
15 docs	0.4427	0.5280	0.5253
20 docs	0.4280	0.5060	0.5070
30 docs	0.4113	0.4793	0.4827
100 docs	0.3138	0.3650	0.3644
200 docs	0.2489	0.2902	0.2907
500 docs	0.1623	0.1832	0.1832
1000 docs	0.1080	0.1196	0.1196
R-Precision (after RelRet)			
Exact	0.2807	0.3232	0.3231
%chg		+15.0	+15.0

Table 6. Automatic ad-hoc run statistics for queries 151-200 against Tipster Disks-1&2 database: (1) *base* - statistical terms only; (2) *nyuir1* - using syntactic phrases, names, and the new weighting scheme; (3) *nyuir2* - same as 2 but different parameters on the weighting scheme.

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	5000
Rel	9353	9353	9353	9353
RelRet	6011	6350	7203	7345
%chg		+5.6	+19.8	+21.2
Recall	(interp) Precision Averages			
0.00	0.7551	0.7401	0.7711	0.7881
0.10	0.4871	0.5003	0.5957	0.6243
0.20	0.4059	0.4255	0.4857	0.5117
0.30	0.3482	0.3719	0.4337	0.4492
0.40	0.2952	0.3330	0.3758	0.4005
0.50	0.2557	0.2723	0.3321	0.3531
0.60	0.2116	0.2327	0.2762	0.2922
0.70	0.1582	0.1765	0.2213	0.2411
0.80	0.0953	0.1094	0.1480	0.1541
0.90	0.0651	0.0691	0.0816	0.0918
1.00	0.0048	0.0077	0.0069	0.0117
Average precision over all rel docs				
Avg	0.2578	0.2743	0.3244	0.3422
%chg		+6.4	+25.8	+32.7
Precision at				
5 docs	0.5080	0.5280	0.6000	0.6080
10 docs	0.4520	0.4800	0.5560	0.5760
15 docs	0.4533	0.4600	0.5333	0.5560
20 docs	0.4410	0.4390	0.5200	0.5370
30 docs	0.4273	0.4273	0.4940	0.5133
100 docs	0.3418	0.3584	0.4098	0.4270
200 docs	0.2838	0.3063	0.3380	0.3484
500 docs	0.1874	0.2008	0.2260	0.2310
1000 docs	0.1202	0.1270	0.1441	0.1469
R-Precision (after Rel)				
Exact	0.3062	0.3135	0.3510	0.3635
%chg		+2.4	+14.6	+18.7

Table 7. Automatic routing run statistics for queries 101-150 against Tipster Disk-3 database: (1) *base* - statistical terms only; (2) *nyuir1* - using syntactic phrases, names, and the new weighting scheme; (3) *nyuir2* - same as 2 with the automatic relevance feedback, but some queries not fully expanded due to an error; (4) *nyuir2a* - nyuir2 rerun after TREC-3 with full feedback.

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 has improved to the point where it can be applied to real IR problems. 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.

At the same time it is important to keep in mind that the NLP techniques that meet our performance requirements (or at least are believed to be approaching these requirements) are still fairly unsophisticated in their ability to handle natural language text. In particular, advanced processing involving conceptual structuring, logical forms, etc., is still beyond reach, computationally. It may be assumed that these 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.

ACKNOWLEDGEMENTS

We would like to thank Donna Harman of NIST for making her PRISE system available to us. Will Rogers provided valuable assistance in installing updated versions of PRISE at NYU. We would also like to thank Ralph Weischedel and Constantine Papageorgiou 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 Projects Agency under Contract N00014-90-J-1851 from the Office of Naval Research, under ARPA's Tipster Phase-2 Contract 94-FI57900-000, and the National Science Foundation under Grant IRI-93-02615.

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.

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 Jose Perez-Carballo. 1994. ''Recent Developments in Natural Language Text Retirieval.'' Proceedings of the Second Text REtrieval Conference (TREC-2), NIST Special Publication 500-215, pp. 123-136.

Strzalkowski, Tomek. 1994. ''Natural Language Information Retrieval'' To appear in **Information Processing and Management.**

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.

Indexing Structures Derived from Syntax in TREC-3: System Description

Alan F. Smeaton, Ruairi O'Donnell and Fergus Kelledy
{asmeaton, rodonnel, fkelledy}@CompApp.DCU.ie

School of Computer Applications
Dublin City University,
Glasnevin, Dublin 9, IRELAND.

Abstract: *This paper describes an approach to information retrieval based on a syntactic analysis of the document texts and user queries, and from that analysis, the construction of tree structures (TSAs) to encode and capture language ambiguities. TSAs are constructed at the clause level and thus each document can yield many TSAs and each query may be represented by several TSAs. The TSAs from documents and from queries are then matched and their degrees of overlap between individual TSAs are computed and then aggregated to yield a score for each document, which is then used in ranking the collection. This paper presents the system description when benchmarking our retrieval strategy on category B of TREC-3, i.e. on c.550 Mbytes of the Wall Street Journal newspaper texts. The implementation is based on a two-stage retrieval where a statistically-based pre-fetch retrieval retrieves the set of WSJ articles for the more computationally expensive language based processing. The results of our retrieval system in terms of precision and recall are disappointing and an analysis of why is also included. Part of this analysis includes a direct comparison between our system and some mainstream IR approaches. In addition to performing ad hoc retrieval on texts in English, we have also performed ad hoc retrieval on texts in Spanish using a weighted trigram approach, and this is outlined and performance results given in an appendix.*

1. Introduction and Motivation

Since 1989 the IR group at Dublin City University has been working on a method to index and retrieve texts based on a domain-independent syntactically-based language analysis. Our thesis has been that the meaning or content of a segment of natural language (query or document) can be at least partly represented by the words used, combined with the structures underlying the syntax used in the language. While it is true that syntax and syntactic structure does not always imply content, and it is equally true to say that there will be many instances of syntactic ambiguity after a syntactic analysis to make uncertain what true content is, our research has tried to address the question of whether there is enough content indication from syntax to overcome these obvious drawbacks to our approach.

In recent years, several other researchers have tried to use syntactic level analysis in the indexing and retrieval of texts. One approach taken has been to use syntactic structure and syntactic relationships between words to derive normalised phrases as indexing terms. Examples of this approach are the CLARIT work (Evans et al, 1993) and earlier work of Fagan (Fagan, 1987). Syntax has also been used to identify head-modifier relationships between words (Salton *et al*, 1990) or to identify dependent word pairs (Strzalkowski and Carbello, 1993). In all these cases, syntactic ambiguity occurring in language is either ignored or normalised. An alternative to this has been to incorporate syntactic ambiguities into the indexing and

retrieval operations and to use alternative interpretations of content, as indicated by syntactic structure, during retrieval. Alternative syntactic interpretations of language are usually best encoded in some kind of structured representation. The TINA project at Siemens, Münich (Schwarz, 1990) and the COP project at Pittsburgh (Metzler and Haas, 1989) are two such examples.

In published experimental evaluations of the effectiveness of document retrieval when using syntax in the ways mentioned above, the first approach, deriving normalised representations, has been the most successful. In fact large-scale evaluations of approaches which encode syntactic ambiguities have been non-existent to date. In our previously published work we have used syntactic analysis to build structures into which we have encoded syntactic ambiguities. These TSAs have been evaluated in phrase to phrase matching and the results we have obtained show promise for future work (Sheridan and Smeaton, 1992). Since then we have scaled phrase-phrase matching up to query-document matching and here we present our first results.

The remainder of this paper is organised as follows. In the next section we briefly outline the language analyser we have used in our work. Following that we present the structures we derive from syntax, known as TSAs and we describe how they are matched individually and how these matches are combined to compute document scores. In section 4 we describe our approach to implementing a document retrieval algorithm on the *WSJ* database. Section 5 presents the official TREC-3 results we obtained. As part of the analysis of these results we re-ran our retrieval strategy on 3 other document rankings obtained from 3 mainline IR research systems and these comparative results are also included here. Section 6 presents some general comments about the validity of our approach to document indexing and retrieval. Finally, in addition to performing retrieval on the *WSJ* we have also performed retrieval on Spanish texts and the approach we have taken here is outlined in an appendix.

2. ENGCG Analysis

The language analyser used in our work was developed at the Research Unit for Computational Linguistics at the University of Helsinki as part of the CEC-funded SIMPR project (1989-1992). It is known as ENGCG and it performs a domain-independent morphosyntactic analysis or tagging of running English text. It has a two-level lexicon and it based upon a constraint grammar.

ENGCG works in 5 stages as follows. The first stage is the assignment of clause and sentence boundaries. This is followed by lexical and morphological analysis where each word is decomposed into all its possible morphological base forms, suffixes and prefixes. The lexicon has c.65,000 entries which recognise over 300,000 word forms. The lexicon also contains about 600 collocations or multi-word terms. As a result of this analysis, many words will have more than one tag so the next stage is a context-sensitive disambiguation to discard all and only the contextually illegitimate morphological readings. This is done using a constraint grammar with c.1,100 rules. In the next stage of the analysis, all possible syntactic functions are assigned to each word token and these tokens describe how a word affects and is affected by other words around it. These syntactic function labels are preceded by a '@' character and modifiers are denoted by a '>' or '<' depending on the direction of the modification. For example, the label '@AN>' indicates that the current word is an adjective (A) modifying a noun (N) somewhere to its right (>). In the final stage of the analysis, the syntactic function labels are disambiguated as much as possible by removing all and only the contextually illegal readings, according to a second constraint grammar of c.300 rules.

In the version we used here, 3% to 6% of words in general texts would be assigned more than one

tag after analysis but the *WSJ* texts contain many proper nouns so perhaps this figure may be higher for this data. By default a word not in the lexicon is treated as a proper noun. There are 32 possible function labels in the analysis output which we group into 6 categories namely heads, modifiers, verbs, adverbs, adjective and stopwords. Examples of stopwords would be determiners, auxiliaries, conjunctions, quantifiers, negation, etc.

In order to give some sense of the output of the ENGCG analyser, a sample of 1 Mbyte of *WSJ* text was analysed. This consisted of 153,198 words or 19,423 clauses, which were assigned an average of 1.138 syntactic labels per word. The occurrence of labels per category is given at the end of this paper and illustrates the kind of distribution to be expected.

Category	*Freq.*	*% Freq*
Modifier	47,098	27.0
Stopword	34,915	20.0
Head	37,590	21.6
Adverb	22,712	13.0
Adjective	9,731	5.6
Verb	22,331	12.8
Total	174,377	100.0

Table 2.1 Distribution of Label Occurrences in sample 1 Mbytes *WSJ* text

Further analysis of this sample text was then performed to determine which types of ambiguity are actually identified in the ENGCG output. We computed the occurrences of words assigned labels from more than one of our 6 categories and found that in the sample of 153,198 words, 7,344 are identified as either a modifier or a head, or put another way, of the 37,590 head category labels assigned, nearly 20% can also be modifiers. Analysis showed that this is due to the subject or object of a sentence also acting as a prepositional compliment. Of the 22,712 adverbs, 6,153 (27%) can also be modifiers which is due to occurrences of phrasal verbs.

In terms of syntactic rather than lexical ambiguities, we counted the number of instances of prepositional phrase attachment ambiguities and found that among the 19,423 clauses there were 4,569 instances. We also looked for ambiguity in complex noun phrases containing a conjunction whose role is ambiguous, i.e. conjoining modifiers or heads ("bearing cups and cones" being @NN> @OBJ @CC @OBJ) and we could find only 806 such occurrences, a surprisingly small figure. Even more surprising was the only 180 instances of conjunction among modifiers where the first modifier could also be a head ("hub and bearing components" being {@NN>,@OBJ} @CC @NN> @OBJ).

An example analysis of some running text, taken from WSJ870324-001 is given here for the sentence "Industry sources put the value of the proposed acquisition at more than 100 million.":

Industry *[industry] @NN>*
sources *[source] @subj*
put *[put] @+FMAINV*
the *[the] @DN>*
value *[value] @OBJ*
of *[of] @<NOM-OF*
the *[the] @DN>*
proposed [propose] @AN>
acquisition *[acquisition] @<P*
at *[at] @NOM @ADV*
more=than *[more=than] @AD-A>*
100 *[100] @qn>*
million *[million] @<P*

The example above shows, for each word in the input sentence, the base form in [] brackets and the syntactic label(s) assigned. The full ENGCG analysis also includes much morphological information on each word which is not shown above. We can also see multi-word lexical entries (more=than) and syntactic ambiguity in this sample.

One of the things which hindered our work was the processing speed of the parser we used which is an early deliverable from the SIMPR project and which runs at 8 to 10 words per second.

Lingsoft Inc have developed the speed but not the performance of this parser and taken it to market with a claimed processing speed of 500 words per second on a SUN SparcStation 10 model 30. The version we have is unsupported and we had no funds to upgrade or buy in support. In addition to speed, we also found some other problems with the analyser like the fact that it cannot handle calendar expressions as it does not use a calendar expression subgrammar, a plural noun cannot be a noun premodifier (industrial relations council ?), many ellipses are missed, an adjective cannot be a prepositional compliment (the bartender in my father's local ?), conjoined premodifier separated by commas are not correctly assigned. These kind of aberrations would be eliminated in the more recent version of the parser.

3. TSA Construction and Matching

In our original paper describing our TSAs (Sheridan and Smeaton, 1992) we generated one TSA per clause. These TSAs were quite complex in structure and hence in manipulations on them. We have since decided to reduce the complexity of TSA structures in the light of our findings re frequency of occurrence of different types of ambiguity. In the present format, each clause generates a list where each item in the list is either a verb, adverb, adjective, stopword or TSA for a noun phrase. For example, the full representation for the sentence given earlier is shown below and the representation for an entire document text would be a set of such lists, one per sentence.

In computing a score between a document and a query there are a large number of factors to be accounted for. Given that a document will be represented by a set of lists, one list per clause, and elements of these lists will be tree-structures, we must consider scoring a query-document match at the following levels:

- Leaf node level, covering matched base forms, matching between identical or related syntactic function labels
- Structural level, covering scores for inexact matches caused by ambiguities and combining node scores into one overall TSA score for clause matches, possibly normalising to account for clause length
- Document level, addressing how clause TSA scores are combined to produce document scores, possibly normalising by clause count.

We have made many experimental runs using TREC-2 queries and relevance assessments and the algorithm described here has been found to give best performance, though there is no guarantee it is optimal.

Overall, the document scoring algorithm works by matching each document TSA against each query TSA and adding the score for the best match to a running total. The next document TSA is then matched against all query TSAs and the score for the best-matched is added to the total and so on until all document TSAs have been processed. This approach presupposes that fragments of information content as represented in clauses, can be repeated in document texts but not in queries. In TREC queries a sub-topic of information need can be repeated within different fields of the query. For example in topic 101 the description field asks for "... *configuration, components and technology of the U.S.'s 'star wars' anti-missile defense system*" while the narrative field asks for "... *design and technology to be used in the anti-missile defense system advocated by ..., also known as 'star wars.'*". Effectively this same sub-topic is repeated in the narrative field but in a more descriptive manner. Our assumption is that where a sub-topic or clause appears in a document text it should be matched against one sub-topic of the query, the one it scores highest against. In aggregating TSA scores for a document we are thus treating TSAs as atomic

units but we have no notion of re-occurrence of TSAs in queries as they are too short. The same topic or TSA representation of that topic can re-occur and be re-scored where it occurs in a document text, where there are on average approximately 60 TSAs. The total score for the document is then normalised by dividing by the number of clauses (hence TSAs) in the document.

Computing the TSA score between two clauses is rather complicated and we will illustrate part of its operation by working through a match between a query clause **"lexical analysis within IR"** and a document clause **"effective IR systems which involve lexical and syntactic analysis of text"**. The TSAs for these are given in Appendix B at the end of the paper, in an abbreviated form. The leaf nodes also hold morphological information on words and the numbers in parentheses beside each word are labels to illustrate the sequence of the matching algorithm. Asterisk nodes illustrate a modification relationship with left child modifying right.

The TSA match starts with node 1 of the query and we find this is a modification relationship so we descend to node 3 and search the document TSA for a base form match which we find at node 12. We now compute a word-word match between these nodes. The document frequency of the baseform *analysis* is 3136 documents so its IDF weight is 2.713 and this is further weighted by a factor of 8 giving 21.9. This factoring comes from a set of rules determining that an adverb (@O-ADVL) and an object (@OBJ) are both significant labels in their respective clauses. The full set of such rules and weights comes from research for the TSA phrase to phrase matching.

We then return to query node 1 and descend to its left child, node 2 whose baseform is matched against node 10 in the document. The baseform frequency of the word *lexical* in *WSJ* texts is only 3 documents, so the IDF weight is 5.762, again weighted by a factor of 8 giving a word-word score of 46.1 as both are playing the same role (modifiers) within their respective clauses. The common ancestors of the matching nodes in respective TSAs are * (node 1) for the query and * (node 8) for the document, indicating both have the same modification relationships but in the residual structure between nodes 10 and 12 of the text TSA we find a conjunction among modifiers which is an indicator of ambiguity so the cumulative score to date (21.9+46.1=68.0) is further incremented by a bonus of only 5 (again from another rule set).

Moving on to query node 4 we find it has a right leaf child (5) and we search for *ir* in the document TSA and find it matches node 4. Its document frequency is 4 and its IDF weight is again multiplied by a factor of 8, the factor assigned to a match between a prepositional compliment (@<P, effectively the head of a prepositional phrase) and a premodifying adjective (@AN>) giving an additional weight of 45.1. The query term *within* does not occur in the document TSA so there is no contribution to the score there. Possible reductions in the overall degree of match caused by syntactic ambiguity due to propositional phrase attachment or ambiguities due to conjunctions in complex noun phrases, for example, would then be applied but they do not affect the occurrences in the present document TSA.

Finally, in addition to normalising query-document TSA scores by a within-document TSA count as described earlier, we also normalise TSA-TSA scores by dividing the word-word and syntactic structure scores by the number of nodes in the query TSA, 5 in this case, giving a final score of 23.6.

This worked example illustrates some inadequacies with our approach. For example, the term IR is an acronym but these are not handled in our retrieval. The most obvious problem with the approach, however, is its complexity which will obviously affect the implementation as described in the next section. A more subtle consequence of the complexity for retrieval effectiveness might be that the way we retrieve now is simply too complex and has lost

sight of the original motivation so the contribution of "structure from syntax implying content" may be too diluted in the overall retrieval.

4. Implementation of Document Retrieval

Because of the computational overhead of calculating query-document similarities using TSA-based matching we took a 2-stage approach to implementing retrieval. For each TREC query we used all fields given and we analysed the text using the ENGCG analyser. Words assigned only one of the syntactic function labels in our stopword category are discarded as are words whose baseforms occur in more than 10% of the documents in the collection. For the remaining words we calculate their IDF weight using their baseforms as indexing terms and we score documents using tf*IDF weighting on these baseforms. The top 1000 ranked documents are then used as input into the second stage of the retrieval process, TSA-based weighting. In earlier experiments we varied the size of the document set returned by this pre-search but we found that quantities above 1000 did not affect subsequent retrieval to any noticeable degree.

The tf*IDF weighting on baseforms which we use as a pre-search is implemented by using an inverted file. storage of the TSA structures was a bit more problematic however. We looked at using an OODBMS to store the TSA structures internally but found that we would have to decompose them into linear lists with extra fields as pointers and the storage overheads would then become too much of a burden. In the end we decided not to store TSA structures internally within GemStone but to use GemStone to manage TSA locations within documents. TSAs are in fact reconstructed on demand from an encoded representation of the ENGCG analysis, a process which is reasonably fast and which allowed us flexibility during our experimental runs if we decided to modify the TSA format.

A second consequence of the computational and storage overhead of our implementation is that we would have been unable to process the entire TREC-3 collection of 2 Gbytes so we elected to become category B participants. Our retrieval was implemented on 550 Mbytes of *Wall Street Journal* newspaper stories from 1986 to 1992, made up of 173,256 documents and c.27,000,000 words. Processing was done on a SUN SparcServer 690MP.

5. Experimental Results

The Dublin City University team submitted two official runs to be evaluated as part of TREC-3 ad hoc retrieval. These were given the codes DCUNL1 and DCUNL2 and are repeated from the proceedings below.

Recall	Precision *(DCUNL1)*	Precision *(DCUNL2)*
0.00	0.6431	0.5453
0.10	0.3973	0.3362
0.20	0.3383	0.2587
0.30	0.2931	0.2193
0.40	0.2321	0.1799
0.50	0.1907	0.1367
0.60	0.1479	0.1039
0.70	0.0999	0.0651
0.80	0.0610	0.0425
0.90	0.0299	0.0212
1.00	0.0097	0.0061
Avg Precision	0.1966	0.1514
P at 30 docs	0.2933	0.2433

Table 5.1 : Comparison of Results for DCUNL1 and DCUNL2

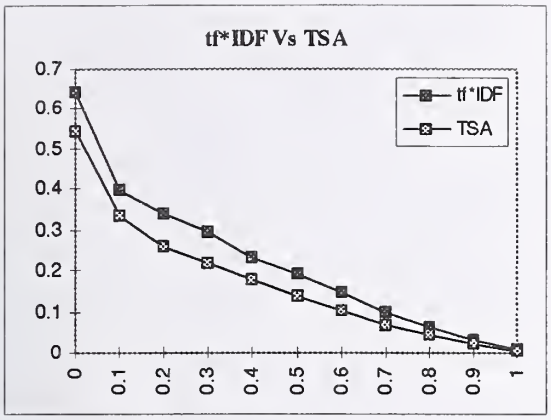


Figure 5.1 : Comparison of Results for DCUNL1 and DCUNL2

In comparison to results obtained by other TREC-3 groups, both category B and A, our results are disappointing. The most disappointing aspect of our results, however, is that our baseline prefetch run (DCUNL1) is actually **better** than our TSA-based run (DCUNL2). This is really quite unexpected. In experimental runs with TREC-2 queries we found that the TSA-based re-ranking improved retrieval slightly at the high precision end of the scale but averaged out as approximately equal to the tf*IDF weighting. In essence what we were finding in our pre-official runs was that our pre-fetch retrieval (tf*IDF weighting on ENGCG word baseforms) was not bringing in as many relevant documents as we wanted and our TSA-based retrieval was improving precision slightly and only at the high precision levels. In an attempt to determine if this was a fault of the quality of the pre-fetch we used, we contacted some other TREC-3 groups to see how our TSA-based retrieval would perform on top of other pre-fetch methods. One difficulty was that there are few category B groups in TREC-3 anyway so after discussion so we elected to use results provided to us using the following systems which are very heterogeneous in nature and hence should retrieve different document sets:

- Chris Buckley at Cornell sent us results he computed for TREC-2 topics (101-150) on category B data using the SMART system with official TREC-2 parameter settings. These results were used to provide a benchmark for TREC-2 category B participants and appeared in the TREC-2 proceedings (pp10).

- Sue Dumais at Bellcore did a special run of LSI-based retrieval on category B data and TREC-2 topics (101-150) where singular valued decomposition was used to reduce the term space. The parameter settings used here were determined from the full category A dataset.

- Jaime Callan at UMass provided results of a run with the INQUERY system on category A data but with only the category B *WSJ* articles extracted from that ranking. The parameter settings here were from the official UMass TREC-2 submission.

In each of these three cases we ran our TSA-based retrieval on the top 1000 document rankings and the results we obtained are presented below:

Recall	Precision	Precision
	Cornell	*TSA*
0.00,	0.8076	0.5736
0.10,	0.6292	0.4347
0.20,	0.5455	0.3461
0.30,	0.4901	0.2821
0.40,	0.4424	0.2478
0.50,	0.3654	0.2097
0.60,	0.3056	0.1788
0.70,	0.2497	0.1405
0.80,	0.1814	0.1052
0.90,	0.0986	0.0577
1.00,	0.0165	0.0051

Table 5.2 : Comparison of Results for TSA and Cornells

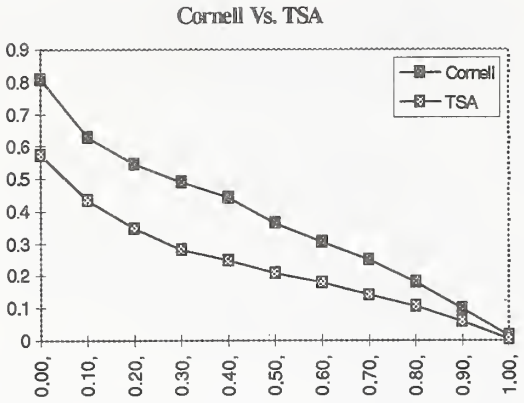


Figure 5.2 : Comparison of Results for TSA and Cornells

Recall	Precision	Precision
	LSI	*TSA*
0	0.7004	0.5721
0.1	0.5124	0.3777
0.2	0.4112	0.3073
0.3	0.3428	0.2612
0.4	0.3003	0.2219
0.5	0.2528	0.1717
0.6	0.2123	0.1319
0.7	0.1672	0.0917
0.8	0.1242	0.0642
0.9	0.0735	0.0339
1	0.0062	0.0043

Table 5.3 : Comparison of Results for TSA and LSI

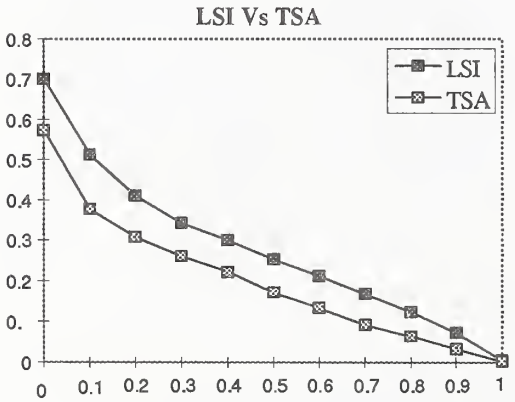


Figure 5.3 : Comparison of Results for TSA and LSI

	Precision	Precision
	INQ	*TSA*
0	0.8002	0.6066
0.1	0.6736	0.4487
0.2	0.5747	0.38
0.3	0.5042	0.3335
0.4	0.4565	0.3017
0.5	0.3984	0.2715
0.6	0.3463	0.2309
0.7	0.2652	0.1659
0.8	0.171	0.0828
0.9	0.0579	0.0195
1	0.0245	0.0087

Table 5.4 : Comparison of Results for TSA and Inquery

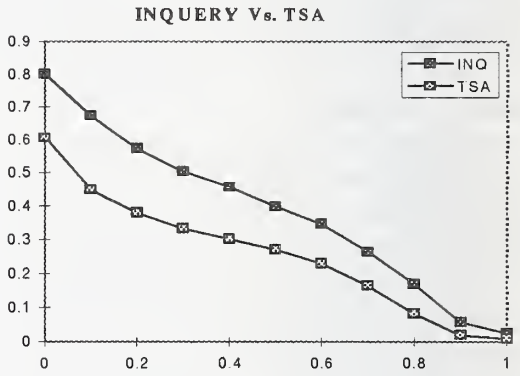


Figure 5.4 : Comparison of Results for TSA and Cornells

As can be seen from the results presented above, we have consistently and significantly managed to disimprove the results of pre-fetch runs from other IR approaches. These results conclusively confirm that the approach of using syntax to determine structural relationships between words and to use them in the way we have described as part of an information retrieval strategy, does not work.

6. Conclusions

There are a variety of reasons why our results are so poor, some of which we postulate here. In the

first place it seems that we score highly, those documents which have query terms in a syntactic relationship in documents which is similar to the relationship in the query, however results suggest that documents are relevant to queries for some other reason besides sharing structural syntactic relationships. A second reason for our poor performance could be that the version of the ENGCG language analyser we used is of poor quality; certainly it is slow ! After experimentation, we found that the score decrement due to syntactic ambiguity is quite low which suggests that this ambiguity does not really matter to retrieval, yet it is a mainstay of our approach. Finally, it has been suggested that the type of language used in TREC topic descriptions is very different to that used in document texts, interrogative vs descriptive language, and this suggests that the two language types should be treated differently, something which we did not do.

As a result of discussions at TREC it has been suggested that instead of piggy-backing TSA-based weighting **on top of** tf*IDF weighting, we could have had better results had we developed a TSA-based retrieval strategy which was totally independent of any kind of term weighting. It has been shown by a number of groups in TREC-3 and elsewhere (UMass, ETH, etc) that combining the results of more than one **independent** retrieval strategy into one overall global document ranking can bootstrap the performance of the individual retrieval strategies. Our TSA-based retrieval could have retrieved documents not retrieved by the term weighting strategy and thus could have been usefully combined, though this could be said of any approach to indexing and retrieval which is independent of any others.

The real contribution of our work, we feel, is the fact that we have at last tried to use structural aspects of syntactic relationships, as part of retrieval. The fact that we have failed to improve retrieval performance and have in fact considerably disimproved retrieval performance tells us that this line of research is not worth pursuing further.

Appendix A: Spanish TREC-3

In addition to taking part in ad hoc retrieval for English texts, the DCU team also took part in ad hoc retrieval from Spanish texts using a weighted trigram approach. Trigram-based retrieval has been tried in TREC-2 by Cavnar (Cavnar, 1993) in both adhoc and routing and the results which have been obtained are poor by TREC-2 standards, though Cavnar believes much better results are obtainable using this approach. After some variations and cul-de-sacs, the following are the steps taken in our official, submitted retrieval runs:

- After ftp-ing and decompressing Spanish, punctuation and numeric characters are discarded and the remaining text is converted to lowercase. Letters in the Spanish character set include upper and lowercase versions of the 26 used in English (a-z, A-Z) plus upper and lowercase versions of *ñ, á, é, í, ó, ú* and the double letter combinations *ll, ch* which can be treated as single characters. These extra characters beyond the 26 a to z are represented in our system by uppercase A to Z equivalents (N, A, E, I, O, and U). Other groups running retrieval on Spanish texts discarded all accents which, given the inconsistent use of accents in our TREC-3 Mexican newspaper texts, may have been the best approach. Such inconsistent use of accents is not so common in Spanish Spanish texts

- The stoplist used in Porter's stemming algorithm was translated into Spanish. These include multiword stopwords which were then treated as several single stopwords. A frequency count of words from a sample of 10 Mbtes of the text was generated and presented to our Spanish colleagues and they added further entries to our stoplist yielding a total of 245 entries. Documents were then stripped of these stopwords and this reduced the size of the documents by c.35%.

- Trigrams, 3 letter overlapping substrings, are then generated from the remaining words which are padded with one space before and after each word. For example, the word sueño (meaning dream) generates the trigrams _su, sue, ueN, eNo, No_, where "_" denotes a space and N denotes ñ.

In analysing the text, after stopword removal, we found a total of 15,861 unique trigram occurrences. The 20 most frequently occurring of these were as follows:

res	_co	ent	nte	ado
_de	_re	est	iOn	_pr
sta	_in	_es	_ca	_se
con	ciO	aci	_di	ica

What is interesting about this is that the most frequently occurring trigram (res) occurred 57,605 times while the 20th most frequently occurring (ica) occurred 54,856 times, so there was a very "flat" and non-Zipfian distribution among the most frequently occurring terms. This does not auger well for a retrieval strategy based on highly weighting the more discriminating terms, or trigrams in our case.

- The same process is then applied to Spanish queries. In our official run we used the topic description and narrative fields and we did not do any processing to remove any noise text.

- An inverted file of trigram occurrences is generated (which is 160 Mbytes in size) to speed up subsequent processing and to determine the frequency of occurrence of trigrams in the document collection.

- A score is calculated for each document which is based on *tf*IDF* weighting as

follows. For each trigram generated from the query, the document frequency was calculated and query trigrams occurring in more than α% of the documents were discarded. The remainder were sorted by increasing document frequency and an IDF weight calculated for each. Document scores were computed by processing query trigrams in order of increasing document frequency (lower IDF weight). Postings lists per trigram had previously been sorted by decreasing within-document frequency and as the algorithm worked through query trigrams, successively less of the postings lists were used in computing document scores thus effectively eliminating processing those with lower within-document frequencies. Finally, in computing document scores we kept accumulators for only a fixed number of documents, the first μ encountered.

- Documents are then ranked by these scores and the top 1000 per query are returned for evaluation.

Our system is implemented on a SUN SparcServer and returns a document ranking in 10-15 seconds for a TREC-3 topic description. In our official submission we chose parameter settings for α and μ (50% and 6800 respectively) which worked best for an implementation on English texts indexed by word stems rather than Spanish texts indexed by trigrams. This was a consequence of having no known queries or relevance assessments with which to work.

After the TREC-3 results and relevance assessments became known, we re-ran our experiments a number of times varying α from 5% to 50% in increments of 5% and for each of these varying μ from 1000 to 7000 in increments of 1000. At this point these would have been termed unjudged runs in that the retrieved documents may or may not have been judged by relevance assessors but nonetheless we obtained significant improvements, a consequence of our bad starting points, not the quality of our subsequent retrieval performance. The average precision values for each run show the best parameter settings for Spanish TREC-3 are 20% and 4000 accumulators. What is more important is that we have improved our official results considerably as the table below shows:

Recall	Precision *Official run*	Precision *Post-Official run*
0.00	0.5483	0.5279
0.10	0.1050	0.3590
0.20	0.0509	0.2608
0.30	0.0449	0.2187
0.40	0.0388	0.1768
0.50	0.0256	0.1362
0.60	0.0080	0.1082
0.70	0.0032	0.0853
0.80	0.0032	0.0565
0.90	0	0.0317
1.00	0	0
Avg Precision	0.0411	0.1612

Table : Results for Spanish TREC Official and Post-Official

The conclusion from our work on Spanish TREC-3 is that our performance in terms of the performances of other groups taking part in Spanish TREC-3 are worse than using weighted quadgrams which are worse than using any kind of crude stemming algorithm and term weighting strategies. This suggests that even though the character set may be larger than for English, trigram based retrieval may not be best. Certainly, trigram based retrieval would probably be good at handling word-word variants as would be generated from OCR and spelling errors but for straightforward retrieval, alternative approaches seem more promising. A more complete description of our work on Spanish TREC-3 is available via ftp (email one of the authors for details).

Acknowledgements: We would like to acknowledge the support of the Commission of

the European Communities under the VALUE program, project CCM-448, "Re-Implementation of SIMPR TSAs" and Dublin City University, in developing this work. We are also grateful to Sue Dumais (Bellcore), Chris Buckley (Cornell) and Jamie Callan (UMass) for either performing runs especially for us, or for providing us with some of their earlier results, and to Bill Richardson and Ciara Walsh from the School of Applied Languages at DCU for helping us with Spanish TREC.

Bibliography

(Cavnar, 1993) "N-Gram-Based Text Filtering for TREC-2", W. Cavnar, *in: The Second Text REtrieval Conference (TREC-2), D.K. Harman (Ed), NIST Special Publication 500-215*, 1993.

(Evans et al, 1993) "Design and Evaluation of the CLARIT-TREC-2 System", D Evans and R Lefferts, *in: The Second Text REtrieval Conference (TREC-2), D.K. Harman (Ed), NIST Special Publication 500-215*, 1993.

(Fagan, 1987) "Experiments in Automatic Phrase Indexing for Document Retrieval: A Comparison of Syntactic and Non-Syntactic Methods", J.L. Fagan, Technical Report 87-868, (Ph.D. Thesis), Department of Computer Science, Cornell University, September 1987.

(Metzler and Haas, 1989) "The Constituent Object Parser: Syntactic Structure Matching for Information Retrieval", D.P Metzler and S.W. Haas, *ACM Transactions on Information Systems*, 7(3), 292-316, 1989.

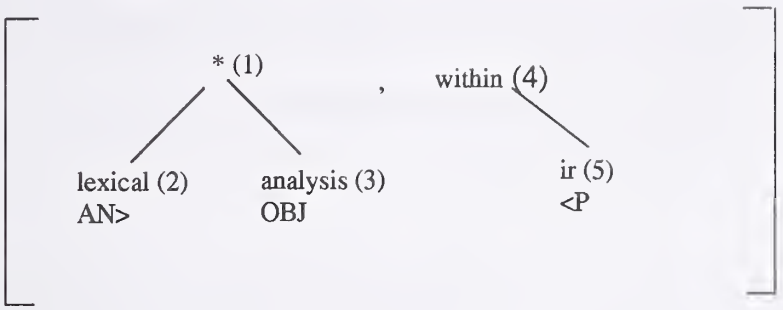
(Salton *et al*, 1990) "On the Application of Syntactic Methodologies in Automatic Text Analysis", G. Salton et al., *Information Processing and Management*, **26** (1), 73-92, 1990.

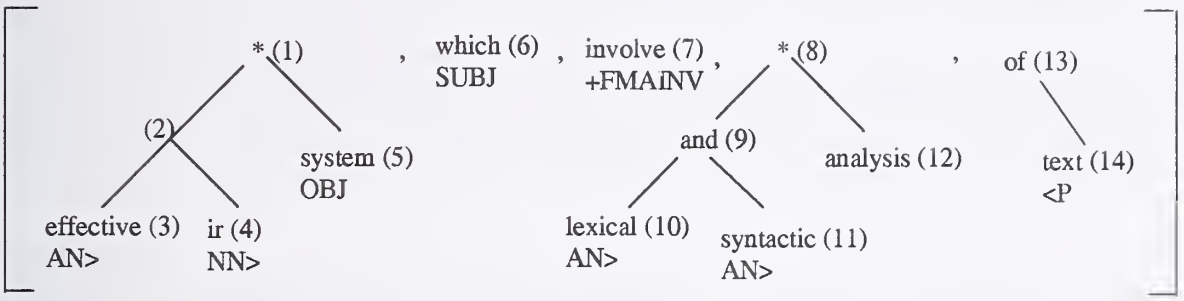
(Schwarz, 1990) "Content Based text Handling", C. Schwarz, *Information Processing and Management*, **26** (2), 219-226, 1990.

(Sheridan and Smeaton, 1992) "The Application of Morpho-syntactic Language Processing to Effective Phrase Matching", P. Sheridan and A.F. Smeaton, *Information Processing and Management*, **28** (2), 1992.

(Strzalkowski and Carbello, 1993) "Recent Developments in Natural Language Text Retrieval", T. Strzalkowski and J Carbello, *in: The Second Text REtrieval Conference (TREC-2), D.K. Harman (Ed), NIST Special Publication 500-215*, 1993.

Appendix B:Sample TSAs





TSAs for sample query and document clauses

Automatic Query Expansion Using SMART : TREC 3

Chris Buckley*, Gerard Salton, James Allan, Amit Singhal

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 TREC 3, performing runs in the routing, ad-hoc, and foreign language environments. Our major focus is massive query expansion: adding from 300 to 530 terms to each query. These terms come from known relevant documents in the case of routing, and from just the top retrieved documents in the case of ad-hoc and Spanish. This approach improves effectiveness from 7% to 25% in the various experiments.

Other 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. Using an overlapping text window definition of "local", we achieve a 16% improvement.

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.[13]

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

*Department of Computer Science, Cornell University, Ithaca, NY 14853-7501. This study was supported in part by the National Science Foundation under grant IRI 93-00124.

terms actually 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.[14, 12] 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 non-zero 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

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

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.[2, 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:[12]

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 and TREC 2 ([2, 1]) are used for TREC 3. Any pair of adjacent non-stopwords is regarded as a potential phrase. 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. 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 sub-vector 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:

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

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

TREC 3 Approaches

One way to improve effectiveness is to better represent the information need by adding useful terms to the query. The classical example of this is relevance feedback, where terms occurring in known relevant documents are added to the query.

The relevance feedback process can be divided into two phases: query term selection and query term weighting. Our basic approach to relevance feedback heavily emphasizes query term weighting. Proper weighting allows us to massively expand the query by adding any term for which we have any evidence of usefulness. Experiments show that effectiveness improves linearly as the log of the number of added terms, up to a point of diminishing improvement [3]. This point of diminishing returns for the TREC environment seems to be about 300 terms.

How can so many terms be added, when it is known that many of them are poor terms and have no connection with relevance? One contributing factor is simply that the good terms tend to co-occur non-randomly within the relevant documents (as opposed to the rest of the collection) and the poor terms tend to co-occur randomly. Massive expansion establishes a background "noise" similarity due to random poor term matches. The good documents escape the noise due to several good terms co-occurring within the document.

Some other expansion methods (eg naive thesaurus lookup) do not share the above property. When expansion occurs inappropriately, several connected poor words are added. Attempting to expand the word "bank" for instance, one might add several financial terms, which may reinforce each other and cause financial documents to be retrieved. This would result in poor retrieval if "bank" were referring to the side of a river. The poor terms for this query expansion are not co-occurring randomly and have a much greater effect on the final similarity.

Expansion by hundreds of terms occurring in known relevant documents worked so successfully in routing of TREC 2, that it was decided to use the same expansion techniques in the ad-hoc portion of TREC 3. In the ad-hoc environment, there are no known relevant documents. Instead, the top retrieved documents are all assumed to be relevant for the purposes of expansion and weighting. If many of the top documents are relevant, then the process achieves the same effect as relevance feedback. If none of the top documents are relevant, then the expansion is likely to have a very negative effect as the refashioned query will emphasize the same mistakes that caused the poor initial retrieval. The end result is thus likely to be a mixture of improvements for many queries, but deterioration of results for others.

The idea of treating the top documents as being relevant in the absence of any real relevance judgements is not a new one. It has probably been done dozens of times in the past (eg, it was a standard Cornell information retrieval class project in the early 1980's). In general, at least in the Cornell experience, it helped some queries but the negative results predominated on the standard small test collections. What makes the approach successful in the TREC environment is the combination of better initial retrieval, and the collection characteristics of TREC. There are many more relevant documents per query within TREC, and those documents are longer than in the small test collections. So there is more of a chance for terms from the relevant retrieved documents to meaningfully distinguish themselves from the terms in the non-relevant documents. Other groups in TREC 2 were able to take advantage of this situation and improve performance, noticeably UCLA and CMU [6, 7].

Another focus of our work the past few years, both within TREC and outside it, has been trying to take advantage of local similarities between a small part of the document and the query. We've shown that local similarities can be used very effectively to ensure that terms in common between document and query are being used in the same semantic sense. However, while this semantic disambiguation is important in other environments [16], the very long and rich TREC queries provide enough global context to disambiguate without going to a local level. Our efforts to improve effectiveness using local disambiguation using sentences and short paragraphs did not work in TREC 1 and TREC 2. For TREC 3, we lengthen our local contexts, and treat the local passage as being a mini-document. Adopting the approach of UMass [17, 4, 5], we define our local contexts to be a set of overlapping text windows, each of fixed size. This avoids the length and normalization problems that adversely affected our approach in TREC 2.

System Description

The Cornell TREC experiments use the SMART Information Retrieval System, Version 11, and are run on a dedicated Sun Sparc 20/51 with 160 Megabytes of memory and 18 Gigabytes 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 rewrite 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 of 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.

Routing Experiments

Our routing experiments in TREC 3 are only slightly different from those carried out for TREC 2. The basic routing approach chosen is the feedback approach of Rocchio [11, 15, 1]. 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 which terms are to be considered part of the final vector.

The investigations of TREC 2 and elsewhere [3] suggest that the decision of which terms to add is not a hard decision: just add all terms occurring in relevant documents that can be efficiently handled. We sort all terms occurring in the relevant documents by the number of relevant documents in which they occur, with ties being broken by considering the highest average weight in the relevant documents. We then add the top 300 single terms and top 30 phrases to the original query and reweight according to the Rocchio formula above with A,B,C parameters being 8,16,4. This forms the queries for our CrnlRR run.

Query-by-Query Variations

While the massive expansion Rocchio approach works well for most queries, examining past individual query results reveals that for about 15% of the queries, not expanding works better than massive expansion[3].

Run	Best	$\geq$ median	$<$ median
CrnlRR	1	44	5
CrnlQR	2	37	11

Table 1: Comparative Routing Results

	Run	*X.Y*	*A.B.C*	R-prec	Total Rel	recall-prec
1.	no fdbk	0.0	8.0.0	3461	5975	2985
2.	no expand	0.0	8.8.4	3698	6342	3163
3.	CrnlRR	300.30	8.16.4	4064	7134	3699
4.	CrnlQR	varies	varies	4013	7215	3725

Table 2: Routing evaluation

Our second official run, like our second official run of TREC 2, is an attempt to choose feedback parameters on a per-query basis to avoid expanding on those queries where no expansion might be appropriate. We also want to examine other feedback approaches for those queries with no, or little, expansion. In TREC 1 and TREC 2 it was noticed that the probabilistic approaches, e.g., the classical probabilistic formula [10] and Dortmund's RPI formula [8, 9], did better than the Rocchio approach if there was little expansion. Perhaps a choice among feedback methods would improve effectiveness; our TREC 2 results suggested just changing expansion amounts on a per-query basis would yield only a small improvement.

We examine seven different approaches:

1. : Original query, no expansion or reweighting.
2. : Probabilistic weights, no expansion.
3. : RPI model, no expansion
4. : RPI model, expansion by 30 single terms
5. : Rocchio, expansion by 30 single terms and 10 phrases
6. : Rocchio, expansion by 500 single terms
7. : Rocchio, expansion by 500 single terms and 30 phrases, with A,B,C parameters being 8,32,4

We ran each of these approaches using the 50 queries of the TREC 3 routing task, learning on D1 and testing on D2. This determined which approach should be used for which queries in the routing task. The final queries for the CrnlQR run are formed by using the best approach on each query, and learning from the full D12 set of relevance judgements.

Routing Results

Both CrnlRR and CrnlQR do quite well in comparison with other TREC 3 routing runs (Table 1). These comparative results are not quite as good as in TREC 2, suggesting that some other groups might have caught up to us.

Evaluation measures in Table 2 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). Just re-weighting the query terms without adding any terms according to Rocchio's algorithm gives a 6% improvement. Both reweighting and massively expanding gives a 24% improvement.

The run CrnlQR is actually quite disappointing. Like our initial attempts at per-query variations in TREC 2, we get very little improvement over using massive expansion for all queries. Table 3 shows that

	Approach	Expansion Sing.phrs	Rocchio *A.B.C*	Num Queries	Num better than CrnlRR
1.	Orig. query	0.0	n.a.	3	1
2.	Prob	0.0	n.a.	4	0
3.	RPI	0.0	n.a.	9	3
4.	RPI	30.0	n.a.	2	1
5.	Rocchio	30.10	8.16.4	3	1
6.	Rocchio	500.0	8.16.4	9	5
7.	Rocchio	500.30	8.32.4	20	15

Table 3: Routing Approach Variation

of all the feedback variations tried, the only ones that consistently do better than the CrnlRR Rocchio expansion by 330 terms, are the queries which use Rocchio and expand by even more terms. The low expansion approaches did better on their queries when learning on D1 and testing on D2, but tended to do worse on their queries when learning on D12 and testing on D3. This suggests that there is no inherent property of the semantics of an individual query that can predict whether massive expansion will work. Instead, it suggests the effectiveness of massive expansion depends on the properties of the documents, in both the learning set and the test set.

Ad-hoc Results

The first of Cornell's two ad-hoc runs, CrnlEA, is very similar to the Rocchio routing run, CrnlRR. The initial query is expanded and reweighted using Rocchio's feedback approach. The major difference is that there are no known relevant documents from which to draw the expansion terms. Instead, an initial retrieval is done, and the top 30 documents are all assumed to be relevant for the purposes of expansion and reweighting. While this is certainly not as good as having real relevance judgements (especially if the initial retrieval obtains no relevant documents), these terms should still have some connection to relevance.

The initial query is expanded by 500 terms and 10 phrases. In the future, perhaps more phrases should be chosen. However, in this initial experiment having many phrases would complicate the analysis of what is actually happening. The *A.B.C* parameters of the Rocchio equation are set to 8.8.0. These parameters weight the original query terms higher than in standard relevance feedback, and disregard occurrences among the non-relevant documents. The parameters were chosen after a small set of trial runs using the first 150 queries on D12.

The second of Cornell's ad-hoc runs, CrnlLA, is this year's local/global run. At retrieval time, each document is assigned a similarity based upon both the document's global similarity to the query, and upon the similarities of smaller parts of the document to the query. For this experiment, the parts of the document are defined to be text windows of 200 words in length. One set of text windows starts at the beginning of the document, with a new window every 200 words. Another set of text windows on that document starts 100 words into the document, with a new window every 200 words. The two sets of overlapping windows ensure that every semantically coherent chunk of text of length less than 100 words will be included whole in at least one text window.

The text of each local window is indexed and weighted with binary term weights (SMART-nomenclature "bnn" weights). The weights in the text windows do not need to be normalized since the text windows are almost all of the same length. An "idf" factor does not need to be included in the local document weights since it will be included in the query weight of any matching term. A pure "tf" factor that gives a weight proportional to the number of times a term occurs in the text window will over-weight common words. Thus the "bnn" weighting scheme would seem to be appropriate.

The question of how to combine a global similarity with the local similarity of a document has yet to be resolved. Work done in preparation for TREC 2 strongly suggested that the result should be some combination of the global similarity with the best local similarity of the document (as opposed to, say, an

Run	Best	$\geq$ median	$<$ median
CrnlEA	3	38	9
CrnlLA	0	49	1

Table 4: Comparative Ad-hoc results

average of local similarities). Other work showed that the values of both global and local similarities are query dependent. A good local similarity for one query may be a poor local similarity for another query. This suggests some sort of query relativization factor may be needed. Several functions were tried in preparation for TREC 3; the one used for the run CrnlLA is

$$\text{FinalSim} = \text{GlobalSim} + 2 * \text{GlobalSim} * \text{LocalSim}/\text{BestLocalSim} \quad (3)$$

where

- GlobalSim is the global similarity between an "ltc" weighted query, Q, and an "lnc" weighted document, D.
- LocalSim is the highest similarity of any "bnn" weighted text window of D with the "ltc" weighted query Q.
- BestLocalSim is the highest LocalSim for any examined document for this query Q.

The CrnlLA retrieval procedure to return rankings for 1000 documents is to

1. Perform a global search retrieving the top 1750 documents.
2. For each retrieved document,
   (a) Fetch the original document,
   (b) Break it into text windows,
   (c) Index and weight each text window separately
   (d) Calculate the similarity of each text window to the query.
   (e) Set the document's LocalSim to the highest of these similarities.
3. Set BestLocalSim to the highest LocalSim among the 1750 documents
4. Use Equation 3 to calculate a final similarity for the 1750 documents.
5. Rank the final similarities and return the top 1000.

The expansion run, CrnlEA, and the local/global run, CrnlLA, are very different but each perform well when compared against other systems. Both approaches perform at or above the median in most queries, as can be seen in in Table 4.

As could be expected, CrnlEA is somewhat inconsistent, performing extremely well on some queries, but dipping below the median on several others. Presumably this is related to the quality of the initial search, though this has not yet been tested. CrnlLA is almost always above the median, but was never the highest rated run.

Table 5 gives the results of several evaluation measures for CrnlEA, CrnlLA, and a simple "lnc.ltc" vector run. Each of the TREC 3 approaches gives substantial recall-precision improvement over the pure vector run (20.3% for CrnlEA, and 16.2% for CrnlLA). However, they get this improvement in very different fashions.

Run	Recall-Precision	Total Rel Retrieved	Precision 5 docs	Precision 100 docs
lnc.ltc	2842	6531	5530	3780
CrnlEA	3419	7267	5760	4168
CrnlLA	3302	6808	6800	4216

Table 5: Ad-hoc results

CrnlEA is a recall oriented approach. It shows a very mild .023 improvement in precision at 5 documents, but retrieves a very strong 736 more relevant documents than the vector run. CrnlLA, on the other hand, is a precision oriented approach. It shows a very strong .1270 improvement in precision at 5 documents, but then a much weaker increase of 277 relevant documents retrieved. It remains to be seen whether the strengths of these two very different approaches can be combined in one run.

Spanish

One of the fun side-tracks of TREC 3 is the Spanish experiments. About 200 megabytes of Spanish text and 25 Spanish queries were made available for runs in the ad-hoc environment. Our claim has always been that SMART is to a large extent language independent, as long as the language is based upon recognizable word tokens. TREC 3 Spanish presented a chance to test this claim.

Spanish SMART

Unlike other retrieval systems, SMART uses almost no linguistic knowledge. Enabling SMART to run well on Spanish text only required 3 subtasks.

1. Make SMART 8-bit clean.
2. Fashion stemming rules for Spanish.
3. Construct a stopword list of common Spanish words.

Extending SMART to handle 8-bit characters (e.g., the accented Spanish characters) instead of 7-bit ASCII was very simple. About 8 lines of code needed changing, plus a 128 entry table in the tokenizer giving the class of characters needed to be expanded to 256 entries.

After this was done, the Spanish document set was indexed without any stemming rules or stopwords. Simple stemming rules were then derived by looking at the sorted dictionary entries and guessing which lexicographically adjacent entries really represented the same words (guessing since the person doing this does not speak Spanish!). The final stemming rules were:

- Remove final "as", "es", "os", "a", "o", "e".
- Change final "z" to "c".

Initially the stopword list was composed of the 800 most frequently occurring words in the collection. This was later trimmed to 342 words by asking a native Spanish speaker to prune the list.

The Spanish collection was then re-indexed using the new stemming rules and stopword list, and was ready for use. It was, however, somewhat disconcerting to type the first query "This is a test", and retrieve a large set of English documents dealing with standard tests! The explanation turned out to be a partial file of English documents that had somehow crept into the distributed collection.

The total time to make SMART Spanish ready was about 5-6 person-hours.

Run	Best	$\geq$ median	$<$ median
CrnlES	11	8	4
CrnlVS	1	13	9

Table 6: Comparative Spanish Ad-hoc results

Run	Recall-Precision	Total Rel Retrieved	Precision 5 docs	R-Precision
CrnlES	5692	2439	7917	5578
CrnlVS	5301	2402	7500	5328

Table 7: Spanish Ad-hoc results

Spanish Ad-Hoc Runs

The two Cornell Spanish runs are CrnlVS, a simple "lnc.ltc" vector run, and CrnlES, a massive expansion run. Both procedures are described above in the main-line ad-hoc description; aside from the different database names there is no difference in the scripts which run the experiments.

Table 6 shows the two runs both do very well, though the expanded run is significantly better. CrnlES has the best results on 11 out of the 23 Spanish queries with relevance documents. (But remember that many fewer groups submitted Spanish runs, so our "share" of best results is expected to be higher.)

The results of the standard evaluation measures show extremely good retrieval effectiveness in Table 7. However, many of these values are artificially high. Unlike the mainstream ad-hoc and routing pools of judged documents, the Spanish pool was very small and narrow, and it's clear that a lower percentage of relevant documents were judged, thus somewhat inflating the recall figures. Comparative evaluation between runs should still be valid though, even between judged and unjudged runs. For example, CrnlES is definitely better than CrnlVS even though CrnlVS was a judged run and CrnlES was not.

After the actual conference, additional relevance judgements on the Spanish TREC runs were made by NIST. The top 150 documents from every Spanish run were judged (as opposed to the judgement of 100 documents from one run of each participant, which is what Table 7 was based upon.) Table 8 show that the additional judgements had very little effect on the final results, despite the addition of 50% more relevant documents to the total judged pool.

Efficiency

Efficiency issues are becoming increasingly important in these TREC experiments as retrieval methods become more complicated and expensive. Thus it is important to have at least some discussion of efficiency within a paper like this.

SMART is a reasonably fast system. It indexes documents at a rate of about 600 megabytes per hour. Simple vector retrieval runs can be quite fast. Calculating the similarities for the CrnlVS run took much less than 2 seconds for all 25 queries together (keeping track of the top 1000 documents for each query was

Run	Recall-Precision	
	Old Judgements	New Judgements
CrnlES	5692	5697
CrnlVS	5301	5013

Table 8: Spanish with New Judgements

Methodology	Run	Recall-Precision	Improvement over Previous Year
TREC 1	ntc.ntc	2067	-
TREC 2	lnc.ltc	2842	38%
TREC 3	CrnlEA	3419	20%
TREC 3	CrnlLA	3302	16%
TREC 4	???	????	> 20%?

Table 9: Runs of queries 151–200 on D12

done rather inefficiently and took much longer!). But the more complicated retrieval methods take anywhere from 9 seconds per query (CrnlRR) to 189 seconds per query (CrnlLA).

Luckily, in actual practice the execution times of the complicated methods can be cut down drastically. The massive query expansion approaches will benefit greatly from optimization efforts such as those discussed in our TREC 1 work. Some of the effectiveness increase of the massive query expansion will have to be traded back in order to get reasonable efficiency, but the results of TREC 1 show the effectiveness cost will not be prohibitive.

The other very time consuming approach of ours is the local/global matching (CrnlLA). The re-indexing of the local parts of a document can be done off-line and stored. When this time savings is combined with the decreased time due to a user asking for a reasonable number of documents (instead of 1000), retrieval time should be not much more than double an ordinary vector search. This should be quite feasible in practice, depending on the particular constraints of a site, of course.

Comparison with TREC 1 and TREC 2

It is difficult to determine how much systems are improving from TREC to TREC since the queries and the documents are changing. For example, in TREC 3 the "Concept" field of the queries was removed. These terms proved to be very good terms for retrieval effectiveness in TREC 1 and TREC 2; thus the TREC 3 task without them is a harder task. To get a handle on how much SMART has improved in the past two years, Table 9 presents the results of running our TREC 1 and TREC 2 systems on the TREC 3 ad-hoc task. SMART has been improving at a rate of over 20% per year so far, and given our work since we submitted the TREC 3 runs, we would expect that improvement rate to continue at least another year.

Conclusion

Automatic massive query expansion proves to be very effective for routing. Conventional relevance feedback techniques are used to weight the expanded queries. Once again, however, the option to choose feedback approaches on a per-query basis doesn't help significantly, where the choice is based on what worked in the past for this query.

Massive query expansion also works in general for the ad-hoc experiments, where expansion and weighting are based on the top initially retrieved documents instead of known relevant documents. In the ad-hoc environment this approach may hurt performance for some queries (e.g., those without many relevant documents in the top retrieved set), but overall proves to be worthwhile with an average 20% improvement.

Incorporating both global and local similarity information in the final ranking is useful, with improvements of 16%. Care needs to be taken, though, both in the definition of a local part of a document (making all parts equal length helps the weighting task enormously), and in the combination of the local and global similarity.

SMART is very easily adaptable to at least some foreign languages, even without knowledge of the languages. Performance of SMART appears to be just as good in the foreign language as in English, though this is tough to judge.

References

[1] Chris Buckley, James Allan, and Gerard Salton. Automatic routing and ad-hoc retrieval using SMART : TREC 2. In D. K. Harman, editor, *Proceedings of the Second Text REtrieval Conference (TREC-2)*, pages 45–56. NIST Special Publication 500-215, March 1994.

[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] Chris Buckley, Gerard Salton, and James Allan. The effect of adding relevance information in a relevance feedback environment. In W. Bruce Croft and C.J. van Rijsbergen, editors, *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 292–300, New York, July 1994. Springer-Verlag.

[4] James P. Callan and W. Bruce Croft. An evaluation of query processing strategies using the TIPSTER collection. In Robert Korfhage, Edie Rasmussen, and Peter Willett, editors, *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 347–355, New York, June 1993. Association for Computer Machinery.

[5] W. Bruce Croft, James Callan, and John Broglio. TREC-2 routing and ad-hoc retrieval evaluation using the INQUERY system. In D. K. Harman, editor, *Proceedings of the Second Text REtrieval Conference (TREC-2)*, pages 75–84. NIST Special Publication 500-215, March 1994.

[6] E. Efthimiadis and P. Biron. UCLA-Okapi at TREC-2: Query expansion experiments. In D. K. Harman, editor, *Proceedings of the Second Text REtrieval Conference (TREC-2)*, pages 279–290. NIST Special Publication 500-215, March 1994.

[7] D. Evans and R. Lefferts. Design and evaluation of the CLARIT-TREC-2 system. In D. K. Harman, editor, *Proceedings of the Second Text REtrieval Conference (TREC-2)*, pages 137–150. NIST Special Publication 500-215, March 1994.

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

[9] 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.

[10] S.E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, May-June 1976.

[11] 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.

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

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

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

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

[16] 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.

[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.

Comparison of Fragmentation Schemes for Document Retrieval

Ross Wilkinson
ross@cs.rmit.oz.au

Justin Zobel
jz@cs.rmit.oz.au

Department of Computer Science, RMIT
GPO Box 2476V, Melbourne, Victoria, Australia 3001

Abstract

Information systems usually rank whole documents to identify which are answers. However, it may in some circumstances be more appropriate to rank fragments to identify which documents are answers. We consider methods of fragmenting and examine the retrieval effectiveness achieved by each method.

1 Introduction

Document structure can be used to improve retrieval effectiveness.[2,4–6] When retrieving only titles or abstracts, document size is highly constrained, as is the content, so that each document is likely to consider only one topic. Thus the whole of the document is likely to be equally relevant to the query. In contrast, a longer document such as a technical article or news report is likely to address several topics.

However, the usual description of a document as a set of weighted terms can lead to the fine structure of a document being completely lost, since the presence a highly relevant fragment will be obscured by other, irrelevant material. It is therefore preferable to match the fragments themselves. This leads to two key problems: what should the fragments be, and how should relevance of fragments be used to determine the relevance of whole documents.

In this paper we consider the first of these questions, and examine several different strategies for dividing documents into fragments. Our conclusion is that the best retrieval is achieved when fragments are constructed to be of similar length, and that there may be a small advantage to fragments that are part of a logical unit such as a section. For the purposes of this discussion, we take the view that if one fragment of the document is of interest, the whole document should be identified as an answer. That is, we have not considered how information about relevance of fragments could be combined to give relevance information about documents.

2 Document fragmentation

There have been several proposed methods for forming fragments. Obvious choices are sentences, paragraphs and pages. The minimum and maximum size of these fragments is another factor that can be explored. Salton and colleagues[1,5] have demonstrated that the use of individual sentences can help determine the relevance of whole documents. In last year's TREC experiments, we demonstrated that pages of 1,000 bytes are useful fragments for retrieval.[8] In order to ensure that poor breaking of a document did not occur, Callan[2] demonstrated that overlapping of fragments can be successful. We have also investigated the explicit structure that can be provided in documents by SGML markup. Having fragmented using this structure, we investigated how the similarity of these fragments could be combined with the structural information to determine the relevance of whole documents,[6,8] with mixed results.

Innovative strategies have been pursued by other researchers. Hearst and Plaunt[3] have shown how structure can be discovered and has advocated queries based on content in context. Mittendorf and Schauble[4] showed how hidden Markov models can be used to discover passages that can be used as retrieval fragments.

We compare several approaches: retrieval based on whole documents; retrieval based on sentences; retrieval based on paragraphs, of a range of fixed maximum lengths; retrieval based on pages, that is, sequences of paragraphs of similar length; and retrieval based on sections, of a range of fixed minimum and maximum lengths.

3 Experimental design

The database used for these experiments was the set of 45,820 documents that comprise the Federal Register (FR) subset of the TREC collection. This set of documents is a good test-bed as it shows a wide variation in size, and documents have an explicit structure, marked up in SGML. The document collection has 70 million words in total, and so average document length is about 1,500 words.

In all experiments, the documents were stemmed but not stopped, and were stored and retrieved using the MG text retrieval system.[7] This system implements a cosine similarity measure and is returns a ranked list with associated similarities. The similarity measure is given by

$$C(q,d) = \frac{\sum_{t \in q \wedge d}(w_{q,t} \cdot w_{d,t})}{\sqrt{\left(\sum_{t \in q}(w_{q,t})^2 \cdot \sum_{t \in d}(w_{d,t})^2\right)}}$$

with

$$w_{x,t} = \log(f_{x,t} + 1) \cdot (\log(N/f_t) + 1)$$

where $f_{x,t}$ is the frequency of t in x, N is the number of documents in the collection, and f_t is the number of documents containing t.

In the case of retrieving documents using fragments, the frequency information is based on the number of fragments containing each term, and a document's similarity is determined by the similarity of the highest ranked fragment.

The query set for these experiments is the subset of 54 of the TREC topics 51-150 that have relevant FR documents. These topics have the SGML mark-up removed and are stopped and stemmed. The standard TREC relevance judgements have been used. After processing, each query had on average 90 words.

When dividing documents into sentences, we only considered sentences within the TEXT or LP fields. A sentence break was registered by any of ".", ", ", ".\n", "\n\n", "?", or "!". In the case of a ".", there was a check to ensure that the previous character is not uppercase, to stop initials or acronyms from causing sentence breaks. Using this definition, there were 3,780,437 sentences.

To break documents up into paragraphs, an SGML tag or "\n\n" was taken to represent the end of a paragraph. Pages are defined to be sequences of paragraphs such that each sequence exceeds, but is is as close as possible to, the required page size. Using a page size of 1,000 bytes, there were 367,141 pages.

Documents were split into sections based on the documents' internal markup. The documents contained several tags that defined internal structure. We found that 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 340,287 sections.

Our experiments investigate the interplay between size and type of fragment. We have previously determined that 1,000 bytes produces a good size for page based retrieval. Maximum paragraph sizes of 100, 500, 1,000, and 5,000 bytes were investigated. Short paragraphs were formed by splitting. We also conducted a similar investigation of section size, testing the effect of imposing minimum sizes and maximum sizes separately, and then combined. Minimum sizes ranged from 100 to 5,000 bytes and maximum sizes from 500 to 10,000 bytes.

4 Results

Our results are shown in Tables 1 to 4. In each table we have shown average precision for the 54 queries at various numbers of documents retrieved; thus for example the first column of numbers in each table is precision at 5 documents. Note that if a document had several fragments with high similarity, then only the highest ranked one was considered, and in general more than 200 fragments must be ranked in order to find the 200 highest ranked documents.

Table 1 describes the effectiveness of retrieving paragraphs of varying maximum size compared with the baseline of retrieving whole documents. We also compare division into pages and sentences.

Table 2 describes the effectiveness of retrieving sections, and then imposing a minimum number of bytes on the sections. Table 3 shows the results of applying a maximum. Thus in the first case we agglomerate sections, in the second case we split sections.

Table 4 shows the best of the methods to date, as well as describing the consequences of putting both minimum and maximum constraints on the size of sections.

It is worth reiterating that these experiments are designed to investigate appropriate fragmentation methods for retrieval, not to determine an appropriate retrieval formula. There has been no attempt to combine evidence, use structural

	Precision at number of documents						
Experiment	5	10	15	25	30	50	200
Documents	0.244	0.228	0.196	0.170	0.147	0.120	0.056
Sentences	0.128	0.092	0.075	0.071	0.063	0.054	0.030
Para-max-100	0.170	0.126	0.114	0.097	0.078	0.060	0.030
Para-max-500	0.093	0.080	0.070	0.064	0.056	0.047	0.026
Para-max-1,000	0.075	0.064	0.059	0.053	0.053	0.043	0.026
Para-max-5,000	0.077	0.063	0.058	0.055	0.054	0.042	0.025
Pages	0.300	0.235	0.206	0.182	0.152	0.117	0.052

Table 1: Comparison of informal fragmentation methods

	Precision at number of documents						
Experiment	5	10	15	25	30	50	200
Sections	0.137	0.113	0.102	0.096	0.078	0.060	0.037
Section-min-100	0.174	0.144	0.131	0.118	0.094	0.078	0.043
Section-min-500	0.233	0.209	0.183	0.159	0.135	0.111	0.053
Section-min-1,000	0.263	0.217	0.200	0.178	0.148	0.117	0.054
Section-min-5,000	0.252	0.233	0.214	0.181	0.157	0.124	0.055

Table 2: Comparison of section fragmentation with minimum size

information, filter documents against minimum similarity criteria.

5 Analysis and conclusions

The first set of experiments show that pages are better units of retrieval than either paragraphs or sentences. Ensuring that paragraphs are not too large makes little difference. The paging method performs significantly better than document ranking in the case of precision at at 5 documents retrieved; but otherwise, although paging has a numerical advantage, these methods cannot be separated by the Wilcoxon signed pairs test.

Ensuring that sections are not too small is clearly beneficial. Even imposing a minimum size of 5,000 bytes—which is quite large—leads to significantly better performance at most levels of retrieval. Curiously, ensuring that sections are no bigger than 1,000 bytes also gives improved retrieval performance; however, this improvement is significant only when retrieving 15 documents, and performance is not as good as with the other methods discussed above. It is interesting to compare having a maximum size for sections and a maximum size for paragraphs: the former gives better retrieval than the latter at almost all numbers of documents retrieved.

A comparison of the best methods shows that there is no significant difference between page-level retrieval and section-level retrieval with a minimum size constraint. Document-level retrieval also works well. The significance of these results is that, for the FR collection, there is only minimal performance gain to be had by fragmenting and using a simple document ranking formula. However, it is interesting to observe that, using fragments as a basis for retrieval, the longer relevant documents are more likely to be identified as answers. That is, use of fragments helps eliminate bias in the cosine method against long documents.

Moreover, in this paper we have concentrated on appropriate fragmenting schemes for ranking. There is clearly potential to better exploit this fine structure. Of particular interest to us is that the section fragmentation now performs as well as any other fragmentation method, but provides the opportunity of using type information, as well as structural information, in order to improve document retrieval.

Acknowledgements

We would like to thank Daniel Lam and Tim Shimmin for their assistance with the experi-

	Precision at number of documents						
Experiment	5	10	15	25	30	50	200
Sections	0.137	0.113	0.102	0.096	0.078	0.060	0.037
Section-max-500	0.159	0.143	0.120	0.110	0.093	0.068	0.033
Section-max-1,000	0.177	0.155	0.141	0.123	0.102	0.075	0.039
Section-max-5,000	0.167	0.139	0.122	0.105	0.084	0.063	0.036
Section-max-10,000	0.167	0.130	0.119	0.105	0.083	0.064	0.036

Table 3: Comparison of section fragmentation with maximum size

	Precision at number of documents						
Experiment	5	10	15	25	30	50	200
Documents	0.244	0.228	0.196	0.170	0.147	0.120	0.056
Pages	0.300	0.235	0.206	0.182	0.152	0.117	0.052
Section-min-1,000	0.263	0.217	0.200	0.178	0.148	0.117	0.054
Section-max-1,000	0.177	0.155	0.141	0.123	0.102	0.075	0.039
Section-1,000-5,000	0.285	0.226	0.200	0.180	0.146	0.117	0.053

Table 4: Comparison of best fragmentation methods

ments described here. We would like to acknowledge the valuable environment of the Multimedia Database Systems group at the Collaborative Information Technology Research Institute. This work was supported by the Australian Research Council and the Centre for Intelligent Decision Systems.

References

[1] J. Allan, C. Buckley, and G. Salton. Automatic routing and ad-hoc retrieval using SMART: TREC 2. In *Proc. Text Retrieval Conference (TREC)*, Gaithersburg, Maryland, 1994. NIST Special Publication 500-215.

[2] P. Callan. Passage-level evidence in document retrieval. In *Proc. ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 302–309, Dublin, Ireland, 1994.

[3] M.A. Hearst and C. Plaunt. Subtopic structuring for full-length document access. In *Proc. ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 59–68, Pittsburg, 1993.

[4] E. Mittendorf and P. Schäuble. Document and passage retrieval based on hidden markov models. In *Proc. ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 318–327, Dublin, Ireland, 1994.

[5] G. Salton, J. Allan, and C. Buckley. Approaches to passage retrieval in full text information systems. In *Proc. ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 49–58, Pittsburg, 1993.

[6] R. Wilkinson. Effective retrieval of structured documents. In *Proc. ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 311–317, Dublin, Ireland, 1994.

[7] I.H. Witten, A. Moffat, and T.C. Bell. *Managing gigabytes: Compressing and indexing documents and images.* Van Nostrand Reinhold, New York, 1994.

[8] J. Zobel, A. Moffat, R. Wilkinson, and R. Sacks-Davis. Efficient retrieval of partial documents. *Information Processing & Management.* (To appear).

Information Retrieval Systems for Large Document Collections

Alistair Moffat* Justin Zobel†

Abstract

Practical information retrieval systems must manage large volumes of data, often divided into several collections that may be held on separate machines. Techniques for locating matches to queries must therefore consider identification of probable collections as well as identification of documents that are probable answers. Furthermore, the large amounts of data involved motivates the use of compression, but in a dynamic environment compression is problematic, because as new text is added the compression model slowly becomes inappropriate. In this paper we describe solutions to both of these problems. We show that use of centralised blocked indexes can reduce overall query processing costs in a multi-collection environment, and that careful application of text compression techniques allow collections to grow by several orders of magnitude without recompression becoming necessary.

1 Introduction

Practical information systems are required to store many gigabytes of data while supporting rapid query evaluation. It is common for a single site to be used as a repository for many separate collections, either centrally managed or stored on separate machines; as indeed is evidenced by the diverse collections in the TREC data. In such environments queries may be posed to the set of collections as well as to individuals in the set.

Queries to the joint collection could be resolved by having a single centralised index of all documents. But such an index inevitably duplicates information held in the index of the individual collections, and, even with the low space requirements possible when the index is stored compressed,[8] may become onerously large if there are many collections. Moreover, if the centralised index is large, access to it may become a bottleneck in query processing.

Another approach is to provide a centralised index that does no more than identify which collections are likely to contain matches, so that queries are resolved by dispatching them to the hosts of likely collections. Such an approach has the advantage of distributing query processing costs, but raises the question of how a collection can be identified as likely to contain answers. A possible method would be to index the vocabulary of each collection—equivalently, index each collection as if it were a single document—with the expectation that a collection with a highly ranked vocabulary is likely to contain answers. However our initial experiments indicated that there was almost no correlation between vocabulary rank and value of a collection. Similar problems are presented by querying of information over Internet, for which a comprehensive search would be prohibitively expensive.[3,11]

Intermediate between these extremes is the possibility of the centralised index being to blocks of fixed numbers of documents, with the expectation that a highly ranked block is more likely to contain highly ranked documents than is a lowly ranked block. We explore such blocking in this paper, with, for the TREC methodology,[10] mixed results. It does allow reduction in the size of the centralised index, but only at the cost of identifying large numbers of false blocks (that is, blocks that rank highly yet contain no highly ranked documents). But the TREC experiments are designed to be high recall, and to some extent the

*Department of Computer Science, The University of Melbourne, Parkville 3052, Victoria, Australia. Email: alistair@cs.mu.oz.au

†Department of Computer Science, RMIT, GPO Box 2476V, Melbourne 3001, Victoria, Australia. Email: jz@cs.rmit.oz.au

problems arise because we are attempting to identify 1,000 highly ranked documents. When only a few documents are required, as is typically the case in ad hoc queries, performance improves.

These experiments have led to some interesting observations. One is that there appears to be no problem with, for a query, combining similarity values from separate collections, even though the weights for the query terms vary. Another is that tiny perturbations in weights can substantially perturb rankings, since even the 1,000'th document in a ranking often has a similarity value that, numerically, is greater than 50% of the score achieved by the top ranked document.

The other problem we have investigated in the last twelve months is the design of efficient compression methods for the text stored in dynamic collections. Although modifications are not as likely in text databases as they are in traditional database systems, it is relatively common for text collections to be continuously extended by addition of new data. Compression techniques for static text databases[8] use semi-static models, computed by initial inspection of the data and thereafter fixed. Semi-static models are required because queries retrieve documents randomly from within collections, and decompression is only practical if all documents have the same model. New text in a dynamic collection can also be coded with a semi-static model computed for the previous data, but compression performance slowly degrades, and if compression efficiency is to be maintained the model should be periodically recomputed and the collection recompressed. This may only be necessary each time the database doubles in size, but for a large collection this might still be an onerous requirement.

To reduce the frequency with which recompression must take place the constraint that all documents have the same model can be slightly relaxed. It is mandatory that the codes allocated to words not change. However if space is left in the model for new codes, to be allocated to new words as they appear, then existing codes can be retained. We have experimented with such a tandem model of document compression, in which known words are allocated codewords based upon their frequency in some initial seed text, and novel words are also stored in the lexicon and are allocated ordinal codes. Introduction of a "recency" technique then allows collections to expand by a factor of 1,000 before compression degradation is evident. For example, a collection of 100 Mb might be safely allowed to grow to 100 Gb before any significant compression degradation occurs. The details of this system are described in full elsewhere;[9] below we provide an overview of the technique and summarise the results obtained.

2 Indexing of multiple collections

Diverse collections of data can be stored by a single information retrieval system, and some information needs can span multiple collections. This is exemplified by the TREC data, the first 2 Gb of which contains nine distinct collections. Some queries would properly be directed only a single collection, such as the more recent set of articles from the Wall Street Journal. But many information needs are dealt with in all of the collections, as is shown by the relevance judgements for the TREC topics.

We were interested to explore how multi-collection queries might be processed if the nine collections were stored separately. The approach we considered was use of a centralised index that could be used to dispatch each query to the sub-collections most likely to contain relevant answers. The returned similarity values would then be compiled into a single ranked list. The combined ranking would then be used as a basis for selecting documents for presentation as answers, which would be requested from the separate collections. Such an approach has obvious application to a distributed system, in which it is desirable to minimise processing costs, and might also be appropriate for more anarchic environments such as collections at separate nodes on Internet.

The experimental regime was as follows. Each collection had a separate index, listing for each word the documents containing the word and the in-document frequencies. The indexes also stored word frequencies, to allow computation of inverse document frequencies in each collection. Thus, for each collection, similarity values were computed only on the information within each collec-

tion, and global information was not available. It was therefore quite possible for a query term to have high weight in one collection and low weight in another. The ranking measure used was the cosine measure with logarithmic in-document frequency,[5] that is, similarity of query q and document d is given by

$$C(q,d) = \frac{\sum_{t \in q \wedge d} (w_{q,t} \cdot w_{d,t})}{\sqrt{\left(\sum_{t \in q} w_{q,t}^2 \cdot \sum_{t \in d} w_{d,t}^2\right)}}$$

where $w_{x,t} = \log(f_{x,t} + 1) \cdot \log(N/f_t + 1)$, $f_{x,t}$ is the frequency of t in x, N is the number of documents in the collection, and f_t is the number of documents containing t.

There was also a centralised index of blocks of documents, where the number B of documents per block was fixed. That is, the words of the kth document of a collection were recorded as occurring in block $\lceil k/B \rceil$ of the collection, and overall word frequencies were based on the number of blocks containing each word rather than the number of containing documents. In such a regime it is quite possible for a block to contain a combination of words but for no document in the block to contain that combination, a problem that becomes more acute as B is increased. We refer to a block as a *false match* if it is highly ranked but contains no highly ranked documents.

Query evaluation with centralised indexes can proceed as follows. The centralised index is used to find the blocks that are highly ranked with respect to the query. If r answers are required, at least $\lceil r/B \rceil$ blocks must be identified; but almost certainly highly ranked blocks will contain some irrelevant documents, so more than $\lceil r/B \rceil$ blocks should be fetched. Even fetching r blocks does not guarantee that the top-ranked documents will be retrieved, particularly when B is large, since a highly similar record might have its strength diluted by occurring in the same block as many records containing no query terms. Exploration of the number of blocks that should be fetched is an important theme of these experiments.

Once the blocks have been ranked, the query is dispatched to each collection with matching blocks. The whole of each such collection could be ranked against the query, so that the purpose of the centralised index is to heuristically select a subset of the collections for further query evaluation. We instead chose to evaluate a mechanism in which only the documents in matching blocks were considered, to reduce processing effort. Each collection c has r_c documents in its matching blocks; of these the top r similarity values must be returned to the central processor if $r \leq r_c$, or r_c similarity values otherwise. At the central processor the similarity values from each collection are merged to produce a single ranking of as many as $\sum r_c$ answers; of these the top r are then retrieved from the collections, collated into rank order, and presented to the user. This four-stage model of query processing is illustrated in Figure 1.

Consider the behaviour of the centrally indexed system as B is varied. With $B = 1$, the central index is ranking documents directly, but this ranking is not the same as would be given by querying the individual collections, as the $\log(N/f_t + 1)$ values will vary between collections. Simply requesting the top r answers may omit documents that would be ranked highly in their collection, so fetching $r' > r$ blocks (each of one document) may yield better retrieval. Even if $r' = r$ blocks are ranked, the similarity values returned by the collections will permute the ranking generated by the central index. Hence, this is already different from our previous approach to TREC, in which the nine subcollections were treated as a single monolithic whole.

Now suppose B is increased, to say 10. The size of the central index diminishes, as the number of blocks containing each word decreases as does the size of the interblock gaps, which in turn allows better index compression. But the new size will still be considerably greater than 10% of the old, because the blocks of 10 documents contains many more distinct terms than the old blocks of one document each. The increase in B also increases the possibility of false block matches, but potentially at least the number of blocks that needs to be fetched to find r answers falls, since a matching block can contain more than one answer. Again, varying r', the number of blocks considered, controls retrieval performance. However, since the number of similarity values returned from the collections to the index will be largely unchanged, overall processing costs at the central index should be substan-

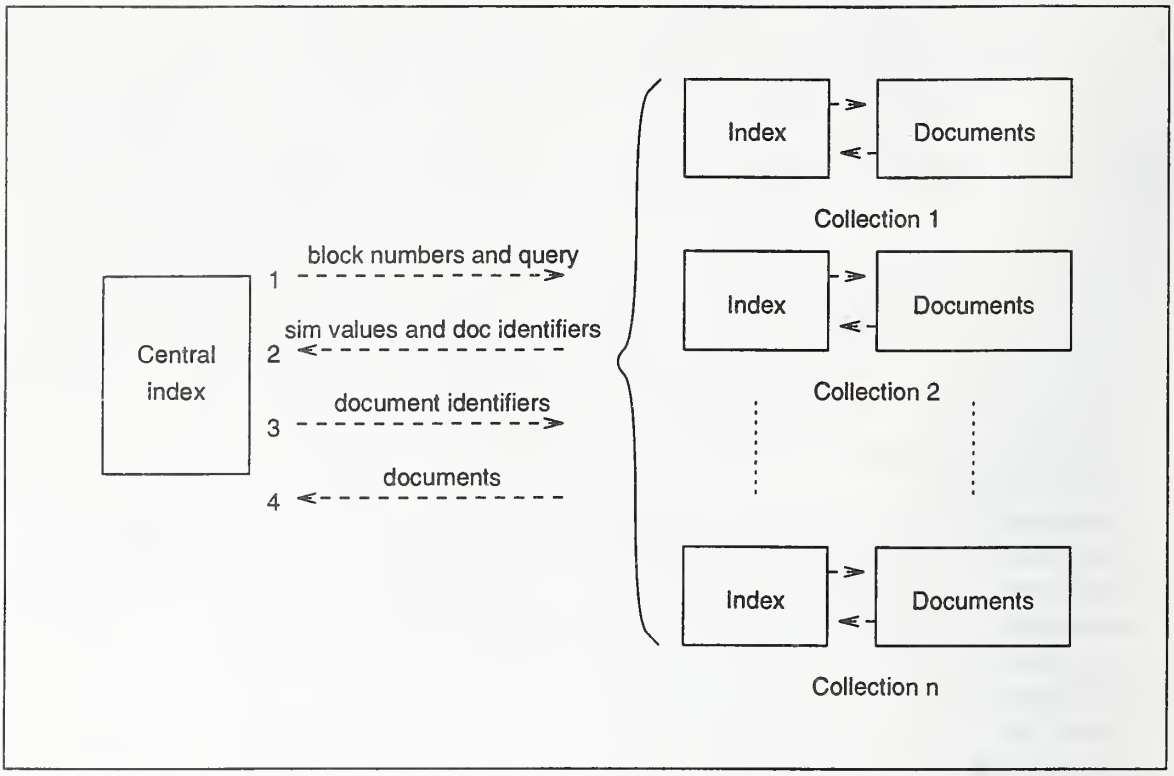


Figure 1: Architecture of a centrally indexed information retrieval system

tially reduced.

As B is increased further central index size and central processing costs continue to fall, but with diminishing returns. Larger blocks imply higher probability of false block match, so that, even if the number r' of blocks considered is decreased, the overall number of documents that must be ranked increases. We would expect that, for sufficiently large B, effectively the whole of each collection would have to be ranked to produce good retrieval effectiveness, somewhat defeating the purpose of a central index.

Following this thought experiment to the limit, when B is huge each collection is a single block. In this case the only benefit of the central index is that some collections might not need to be considered, if for example the rankings of the collections are widely separated. But the likelihood of false collection match seems very high, and we would expect there to be little benefit to such coarse level indexing.

Note that we have not in any way supposed that blocks are constructed according to relatedness of topic. If blocks consisted of similar documents then retrieval performance for a given r' should improve.

3 Experimental results

We have experimented with a centralised index by regarding the TREC data as consisting of nine separate collections, AP1, AP2, DOE, FR1, FR2, WSJ1, WSJ2, ZIFF1, and ZIFF2. The centralised index was implemented by using mg[8,13] to index the TREC data, but with document separators deleted to mimic aggregation of documents into blocks. Thus reindexing was required for each value of B tested. Topics 51–150 were transformed into mg input by removal of stopwords and SGML markup. The output of each query was a list of block numbers, which were transformed into lists of TREC document identifiers by Unix utilities. We had already precomputed for each collection a 10,000-document list of similarity values for each query; and so final lists of answers were computed by joining the query output to

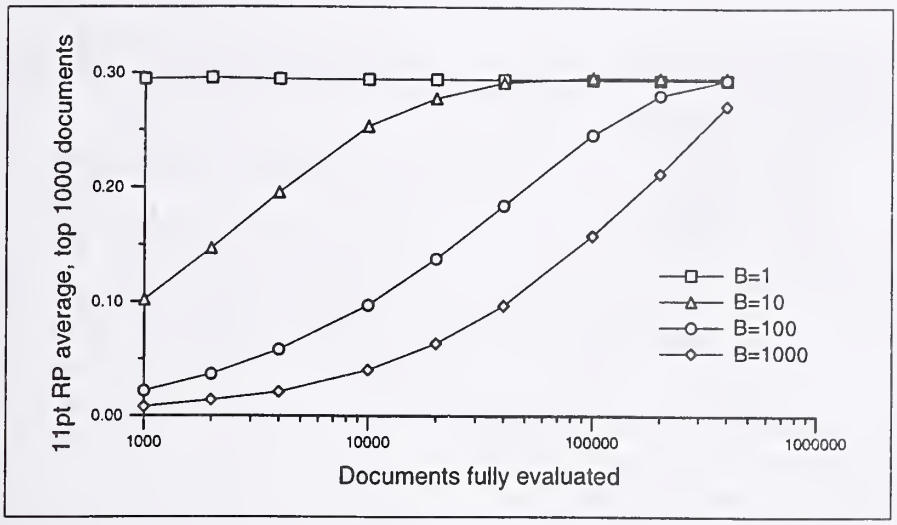


Figure 2: Recall-precision at different block sizes, r = 1,000, topics 51–150

these values. That is, the process of selecting answers from collections was simulated, and while this gives exact values for retrieval effectiveness, we can only estimate the CPU time required by this phase of query evaluation.

As a baseline, we also indexed the full TREC collection in the normal way—that is, as one monolithic collection—and computed retrieval effectiveness as an 11-point average assuming that 1,000 documents are returned as answers for each query. The result (queries 51–150) was an effectiveness of 29.4%.

We then experimented with the retrieval effectiveness achieved for retrieval of 1,000 documents per query and various values of B, as shown in Figure 2. The horizontal axis is the number Br' of documents considered, that is, B times the number of blocks identified as highly ranked. For $B = 1$, the central index is identical to that of our baseline, and as can be seen performance is almost independent of the number of documents fetched. There is a slight peak around $Br' = 2{,}000$, indicating that there may be a small advantage to ranking documents as members of individual (presumably homogeneous) collections compared to ranking them as members of one heterogeneous collection. We would expect such an effect to be more pronounced over larger numbers of collections, or sets of collections that are more diverse.

For higher B, false block match rapidly becomes a significant problem. For blocks of 10 records, even fetching 1,000 blocks (that is, $Br' =$ 10,000) results in a loss of retrieval effectiveness. For $B = 1{,}000$, the bulk of the collection must be processed to achieve good performance.

These problems are to some extent due to the experimental methodology. The TREC procedure of fetching 1,000 records is used to ensure high recall, so that there is some degree of certainty that most of the relevant documents have been located. But for ad hoc queries a much smaller number of answers are typically requested. To test the performance of a central index in this context, we have rerun the above experiments but assuming that only 10 answers are returned for each query. Results are shown in Figure 3. In this case the vertical axis records average precision after 10 documents are retrieved. As can be seen, there is a similar pattern of performance: a disproportionately large number of documents must be processed to achieve good performance when B is large. However in this case the total amount of effort required is reduced. For example, if a $B = 100$ index is being used, around 100 blocks and 10,000 documents must be ranked to achieve performance comparable to that attained by the $B = 1$ index examining just 10 blocks. But, if we equate volume of index inspected with ranking time, in the latter case all of the central index must be examined, plus 10/742,000'ths of the document indexes; while in the former case,

Blocking factor B	Index Size (Mbyte)
1	127.6
10	63.5
100	26.9
1,000	9.7

Table 1: Size of central inverted index

a central index one fifth the size must be examined, plus 10,000/742,000'ths of the component indexes, a net saving in index processing time.

Table 1 shows the sizes of the central indexes used in these experiments. The nine subcollection indexes totalled 118.4 Mb.

4 The 1994 experiments

This section summarises the two experimental runs that were submitted for assessment in 1994. For both of these runs topics 151–200 were transformed into queries by removal of SGML tags, removal of stopwords, and removal of all non-alphabetic characters. Detailed performance analysis of these two runs appears elsewhere in these proceedings.

Run citri1

This was our control run, and used a blocksize of $B = 1$ and extracted the top $r = 1{,}000$ documents according to the central index and then permuted them in the final ranking according to their similarity to the query using the collection weights. Based upon the performance when used with topics 51–150, we expected this to give almost identical behaviour to a ranking based solely upon the central index, using the global term weights.

Run citri2

In this run a blocksize of $B = 100$ was used, and the top 1,000 blocks were expanded in the collections for each query. Hence, the final ranking was the top 1,000 documents—according to the subcollection weights—out of the 100,000 documents contained in the 1,000 most highly ranked blocks according to the central index. If each block contains on average just one good candidate, overall retrieval performance should be good.

5 Dynamic text compression

It is attractive to compress the contents of an information retrieval system.[4,12] With an appropriate choice of coding scheme, such as a semi-static word-based model with Huffman coding,[1,2,6,7] not only can the space required to store the data be reduced to under 30% of the original, but the time required to retrieve data can actually be reduced because of lower data transfer times and smaller seek distances on disk. The disadvantage of this approach is that if the collection is dynamic and documents are to be appended the new text must be compressed with the existing model, so that the model parameters slowly become inappropriate and compression performance degrades.

To allow coding of any new words at all, it is necessary to leave probability space in the model. In the case of Huffman coding this is equivalent to explicitly reserving a family of codes for new words. A straightforward approach is to allocate a single escape code; a new word is then coded as an escape followed by some simple representation such as a length followed by ASCII characters. However, as the stored text grows such an approach quickly leads to progressively worse compression; for example, in a newspaper database, appended text represents a chronological progression of events, in which new words come into frequent use ("Cherynobyl" is an example in the Wall Street Journal) and the use of others declines ("Reagan", for example, is now relatively infrequent).

But given that code space has been reserved, an arbitrary number of new codes can be allocated, so long as each new code in turn leaves space for further codes to be added, and provided that, once assigned, no codeword is ever changed. Thus new words can, without risk of ambiguity, be inserted into the lexicon of known words and allocated a new code. In a sense, two channels of communication from encoder to decoder can be used. The first is the conventional stream of encoded text, as for all compression applications. The second channel is where we benefit from the

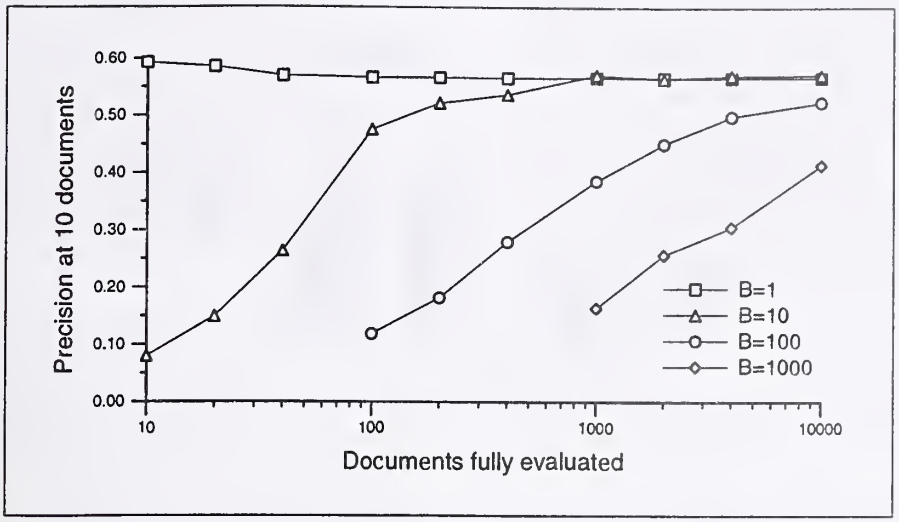


Figure 3: Precision at different block sizes, $r = 10$, topics 51–150

specialised nature of this application: it is the lexicon of terms and codewords assigned, and because no document can be accessed (and hence decoded) prior to its insertion, the scheme is robust.

We have had success with a tandem scheme on the above principles. In this scheme, a pass is made over all of the text that is initially available, a word model computed, and canonical Huffman codes allocated to the words. The code set also includes an escape code, calculated upon an estimate of the rate at which new words can be expected to appear. Then for new words an auxiliary lexicon is used, in which words are listed in order of first occurrence an assigned ordinal codes based upon some unambiguous method for representing infinitely large integers. During the compression process words in the primary lexicon are coded using their pre-calculated Huffman codes, while words in the auxiliary lexicon are coded as the escape code followed by their ordinal number in the auxiliary lexicon. Words novel even to the auxiliary lexicon are simply assigned the next ordinal code and installed, and that ordinal code then emitted. In particular, there is no need for novel words to be "spelt out" to the decoder as would be the case in a more conventional one-channel adaptive compression scheme.

Since any fixed code disadvantages words that are frequent in the appended documents (which might far outnumber the seed documents) but rare in the original seed text, we have added a further heuristic to allow the list of auxiliary words to be self-adjusting within any particular document. The codeword assignment must, of necessity, be fixed at the commencement of each document. However, once an auxiliary word has been seen once in a document it is likely that it will appear again a second time in that document. To exploit this recency effect we adjusted the ordering within the auxiliary lexicon after each use of it, swapping the word used to the next position at the front of the lexicon. This gives improved compression because any infinite encoding of the positive integers must assign arbitrarily long codewords to arbitrarily large integers, and so small integers must be assigned codewords of bounded length. Hence, novel words that appear more than once in any document are usually coded with a shorter codeword at their second and subsequent appearances than at their first.

The question then becomes one of selecting a coding scheme for new words. After experimenting with a variety of codes, we used a variable-length code parameterised by the size of the initial lexicon. The combination of all of these components allows the text to grow by a factor of 1,000 before compression performance significantly degrades, as is illustrated in Table 2, which shows compression performance on the 508 Mb

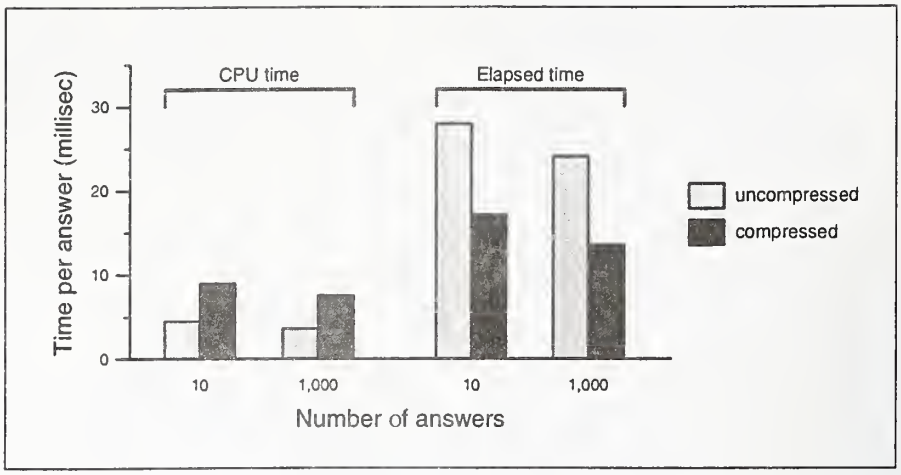


Figure 4: Speed of compressed and uncompressed retrieval systems, milliseconds per answer

Wall Street Journal (collections WSJ1 and WSJ2). In this table, the lefthand column is the factor by which the text has expanded since the model was computed and the codes allocated; thus, in the bottom line, the initial text is only about 32 Kb. The second column is the compressed size achieved, as a percentage of the original size, using a fixed Huffman code on characters to "spell" new words at each of their appearances; while the third column is the compression attained by the tandem model advocated here. The base compression, corresponding to an expansion factor of 1, is 28.4%.

Expansion factor	Character Huffman (%)	Tandem Huffman (%)
4	29.9	29.4
64	31.3	29.6
1,024	35.5	29.9
16,384	49.2	31.7

Table 2: Compression performance for the Wall Street Journal for varying initial text size

Decoding using the improved model is still fast, and it takes around 3–5 milliseconds to decompress the typical TREC document of around 3 Kb. This is only a small component of retrieval time, which is still dominated by disk access and transfer costs. Indeed, the 70% space saving brought about by compression means that not only are transfer times less when the data is compressed, but also that seek times are less. Figure 4 shows the time, in milliseconds per answer, to randomly retrieve 10 and 1,000 documents from a compressed and an uncompressed WSJ database respectively using the `mg` software. In each experiment a random list of ordinal document identifiers was generated and the system asked to fetch and present those documents in document number order, mimicking the effect of a Boolean query. Each of the points plotted is the average over 500,000 document retrievals, run on an otherwise idle machine and against a retrieval system stored on local disks. As can be seen, the CPU time required per answer is greater when the database is compressed, but the elapsed time—which includes disk seek and transfer costs—is less.

Further details of the compression scheme proposed for dynamic collections are described in the full paper.[9]

6 Conclusions

We have explored two hypotheses during the last twelve months. The first is that large agglomerations of data such as the TREC corpus can be usefully regarded as a collection of subcollections without substantial loss of retrieval performance. As an extension to this, we have also considered how the separate collections might be managed in

a distributed environment. Or results are mixed: effective information retrieval can be carried out by using a central index to refer queries on to the appropriate subcollections, but in the TREC framework only at the expense of inspecting a non-trivial amount of each subcollection. On the other hand, ad hoc queries involving a small number of answers can be efficiently partitioned, with a net saving in work and a useful distribution of workload.

The second hypothesis explored in 1994 is that compression can be effectively applied to the text of dynamic collections just as well as it can to static. Our results here show that, seeded with as little as 0.1% of the final WSJ collection, compression results comparable to those attained for the complete collection can be attained. That is, compression can be applied to dynamic collections with little need for periodic subsequent rebuilding of the database.

Acknowledgements

We would like to thank Tim Shimmin for programming support. This work was supported by the Australian Research Council, the Key Centre for Knowledge-Based Systems, the Centre for Intelligent Decision Systems, and the Collaborative Information Technology Research Institute.

References

[1] T.C. Bell, J.G. Cleary, and I.H. Witten. *Text Compression.* Prentice-Hall, Englewood Cliffs, New Jersey, 1990.

[2] A. Bookstein, S.T. Klein, and D.A. Ziff. A systematic approach to compressing a full-text retrieval system. *Information Processing & Management*, 28(6):795–806, 1992.

[3] C.M. Bowman, P.B Danzig, and M.F. Schwartz. Research problems for scaleable Internet resource discovery. Technical Report CU-CS-643-93, Computer Science Department, University of Colorado at Boulder, 1993.

[4] G.V. Cormack. Data compression on a database system. *Communications of the ACM*, 28(12):1336–1342, December 1985.

[5] W.B. Frakes and R. Baeza-Yates, editors. *Information Retrieval: Data Structures and Algorithms.* Prentice-Hall, 1992.

[6] R.N. Horspool and G.V. Cormack. Constructing word-based text compression algorithms. In J.A. Storer and M. Cohn, editors, *Proc. IEEE Data Compression Conference*, pages 62–81, Snowbird, Utah, March 1992. IEEE Computer Society Press, Los Alamitos, California.

[7] D.A. Huffman. A method for the construction of minimum redundancy codes. *Proc. IRE*, 40(9):1098–1101, September 1952.

[8] A. Moffat and J. Zobel. Compression and fast indexing for multi-gigabyte text databases. *Australian Computer Journal*, 26(1):1–9, 1994.

[9] A. Moffat, J. Zobel, and N. Sharman. Text compression for dynamic document databases. Technical Report TR-94-4, Collaborative Information Technology Research Institute, RMIT and The University of Melbourne, 1994.

[10] National Institute of Standards and Technology. *Proc. Text Retrieval Conference (TREC)*, Washington, November 1992. Special Publication 500-207.

[11] M.F. Schwartz, A. Emtage, B. Kahle, and B.C. Neuman. A comparison of Internet resource discovery approaches. *Computing Systems*, 5(4):461–493, 1992.

[12] D.G. Severance. A practitioner's guide to data base compression. *Information Systems*, 8(1):51–62, 1983.

[13] I.H. Witten, A. Moffat, and T.C. Bell. *Managing Gigabytes: Compressing and indexing documents and images.* Van Nostrand Reinhold, New York, 1994.

The Collection Fusion Problem

Ellen M. Voorhees Narendra K. Gupta Ben Johnson-Laird
Siemens Corporate Research, Inc.
Princeton, NJ
{ellen,gupta,ben}@scr.siemens.com

Abstract

This paper examines the feasibility of merging the results of retrieval runs on separate, autonomous document collections into an effective combined result. In particular, we examine two collection fusion techniques that use the results of past queries to compute the number of documents to retrieve from each of a set of subcollections such that the total number of retrieved documents is equal to N, the number of documents to be returned to the user. The fusion techniques are independent of the particular weighting schemes, similarity measures, and retrieval models used by the component collections.

Our official TREC-3 runs are fusion runs in which $N = 1000$; other runs investigate the effects of varying N. These results show that the precision averaged over the 50 queries is within 10% of the precision of an effective single collection run for a wide range of values of N.

1 Introduction

Data fusion techniques have been used in information retrieval to improve the effectiveness of a given query by merging the results produced by different formulations of the query in the context of a single database [1, 2, 3, 7]. In this paper, we look at a different fusion problem — the *collection fusion* problem. The goal of a collection fusion technique is to combine the retrieval results from multiple, independent collections into a single result such that the effectiveness of the combination approximates the effectiveness of searching the entire set of documents as a single collection. Since the collections are assumed to be independent, the fusion techniques must cope with arbitrary similarity measures and weighting schemes.

The TREC environment is an attractive vehicle for investigating the collection fusion problem for two reasons:

- The TREC collection is composed of distinct subcollections that are different sizes, cover diverse topics, and have different retrieval characteristics.
- Other groups participating in the TREC workshops treat the set of documents as a single collection. We can therefore measure how closely the results of our fusion strategies approximate effective single-collection results.

In another report [9], we used the portions of the TREC collection available after TREC-2 to develop and evaluate the collection fusion strategies that are described here. We show there that for modest numbers of documents to be retrieved ($N < 100$), the effectiveness of the fused result is usually within 10% of the effectiveness of a run in which the entire set of documents is treated as one collection (hereafter called a *single collection* run).

In TREC-3, we look at larger values of N, and, as a consequence, investigate methods for producing a total ranking in a fused result. Our official TREC-3 runs, `siems1` and `siems2`, represent the output from the two fusion techniques when $N = 1000$. The runs are completely automatic, ad hoc runs; Siemens did not do any routing or interactive runs in TREC-3.

The next section of the paper gives a formal definition of the collection fusion problem and describes our fusion strategies. The following section compares the effectiveness of the fused results to a particular single collection run. The final section summarizes our findings.

2 The Collection Fusion Problem

Consider a set of collections each of which is accessed through its information server. For a given query Q, each collection I has a certain number of relevant documents, denoted by n_Q^I. We assume when the server for I is presented with Q, it returns a list of documents sorted by decreasing similarity to the query. Clearly, the number of retrieved relevant documents increases as the number of retrieved documents increases until all n_Q^I relevant documents are retrieved. We denote the distribution of the relevant documents in the retrieved set (i.e., the ranks at which the relevant documents occur) by $F_Q^I(S)$, a function of the number of retrieved documents S. The collection fusion problem can now be formally stated as follows: Given a query Q, information servers $I_1, I_2, ...I_C$ and N the total number of documents to be retrieved, find the values of $\lambda_1, \lambda_2, ...\lambda_C$ such that $\sum_{i=1}^{C} \lambda_i = N$ and $\sum_{i=1}^{C} F_Q^{I_i}(\lambda_i)$ is maximum, i.e. the total number of relevant documents retrieved is maximized. Of course, in practice F_Q^I is not known and must be approximated.

There are two simple approaches to approximating the relevant document distributions. One approach is to assume each collection has as many relevant documents as the next and the relevant documents are identically distributed across the servers' rankings. In this case, retrieving an equal number of documents from each collection maximizes the number of relevant documents retrieved on average. In practice, this is a poor approximation because the different collections have different specialties and thus do not have equal numbers of relevant documents. Tests comparing the effectiveness of this uniform strategy versus a single collection run using topics 1-150 demonstrate that the uniform strategy degrades the single collection performance by over 40% [9].

The second approach is to assume the similarity values across collections are comparable and to select the cut-off levels such that the documents with the N greatest similarities across all collections are retrieved. This approach is viable if the similarity measures are indeed comparable. However, the incorporation of collection-dependent frequency counts in the document or query weights (such as idf weights) invalidates this assumption. This effect, noted by Dumais in her TREC work [6], is illustrated by the example in Figure 1. The example shows TREC topic 144 and the rankings produced by each of the five subcollections. For this query, the top similarities strategy retrieves only seven relevant documents when $N = 100$ while the uniform strategy retrieves fourteen. The problem is that both "United Nations" and "organization" are rare terms in the DOE subcollection, and thus irrelevant documents that contain these words have a higher similarity to the query than do the relevant documents in the AP subcollection.

Our approach to approximating the relevant document distributions is to learn them from the results of past queries. That is, training queries are used to build

TREC query 144 (abbreviated):

management effectiveness of the United Nations (UN) and related organizations

Individual collection retrieval results:

	AP	DOE	FR	WSJ	ZIFF
# rel in top 100	17	0	0	6	0
# rel in top 20	9	0	0	5	0
greatest similarity	.1549	.2032	.1235	.2368	.1637
# docs with sim > .1549	0	7	0	6	4

Text of DOE document with greatest similarity:

The relations of the IAEA with the United Nations, with subsidiary UN organs, with other UN specialized agencies, with intergovernmental organizations and with non-governmental organizations are briefly described.

Figure 1: Example in which retrieving the N documents with highest similarities across subcollections is very ineffective.

models of both the content and the search behavior of each collection. Once training is complete, previously unseen queries are answered by matching the new query's content to that of the training queries and using the associated models to compute the number of documents to retrieve from each collection.

2.1 Modeling Relevant Document Distributions

In the relevant document distribution fusion technique, we use the training queries to explicitly build a model of the distributions of the relevant documents in the retrieved set of each information server I, and use the models in a maximization procedure to obtain the number of documents to retrieve from each collection. We model the relevant document distribution of a query, q, by averaging the relevant document distributions of the k most similar training queries (i.e., the k nearest neighbors of q). We use the vector space model to compute the similarities among the queries. The vector space is built from the set of training queries, and the nearest neighbors of q are those training queries that have the highest cosine similarity with q.

The average relevant document distribution over k queries is computed by taking the average of the number of relevant documents retrieved by the set of queries after each document retrieved. Once the average relevant document distribution is computed for the current query for each collection, the distributions and the total number of documents to be retrieved are passed to a maximization procedure. This procedure finds the cut-off level, λ_i, for each collection that maximizes the number of relevant documents retrieved (the current maximization procedure simply does an exhaustive search). The computed cut-off levels are the number of documents selected from each collection.

The union of the top λ_i documents of each collection produces a set of retrieved documents, but does not impose a total order on those documents. (It does produce a set of partial orders since, by assumption, all of the documents retrieved from the same server are ordered relative to one another.) However, a total ordering is necessary to produce ranked output. We impose a total ordering probabilistically. To select the document for rank r, a collection is chosen by rolling a C-faced die that is biased by the number of documents still to be picked from each of the C collections. The next document from that collection is placed at rank r and removed from further consideration. Rankings produced in this way are guaranteed to respect the partial orders and give preference to collections that are the most likely to contain relevant documents. Figure 2 summarizes the steps of the relevant document distribution fusion process.

2.2 Query Clustering

The second fusion strategy does not form an explicit model of a collection's relevant document distribution. Instead, the system learns a measure of the quality of a search for a particular topic area on the collection. The number of documents retrieved from a collection for a new query is proportional to the value of the quality measure computed for that query.

As in the relevant document distribution approach, the query clustering fusion strategy uses query vectors in the vector space formed from the set of training queries. Topic areas are represented as centroids of query clusters. For each collection, the set of training queries is clustered using the number of documents retrieved in common between two queries as a similarity measure. The assumption is that if two queries retrieve many documents in common they are about the same topic. The centroid of a query cluster is created by averaging the vectors of the queries contained within the cluster. This centroid is the system's representation of the topic covered by that query cluster.

The training phase also assigns to a cluster a weight that reflects how effective queries in the cluster are on that collection. The weight is computed as the average number of relevant documents retrieved by queries in the cluster, where a document is "retrieved" if it was among the first L (a parameter of the method) documents.

After training, queries are processed as follows. The cluster whose centroid vector is most similar to the query vector is selected for the query and the associated weight is returned. The set of weights returned by all the collections are used to apportion the retrieved set such that when N documents are to be returned and w_i is the weight returned by collection i, $\frac{w_i}{\sum_{i=1}^{C} w_i} * N$ (rounded appropriately) documents are retrieved from collection i. For example, assume the total number of documents to be retrieved is 100, and there are five collections. If the weights returned by the collections are 4, 3, 3, 0, 2, then 33 documents would be retrieved from collection 1, 25 each from collections 2 and 3, none from collection 4, and 17 from collection 5. However, if the weights returned were 4, 8, 4, 0, 0 then 25 documents would be retrieved from each of collections 1 and 3, and 50 documents would be retrieved from collection 2. The weight of a cluster for a single collection in isolation is not meaningful; it is the relative difference in weights returned by the set of collections over which the fusion is to be performed that is important.

As in the previous method, once the number of documents to be selected from each collection is determined, a final ranking of the documents must be created. To investigate the effects of a different rank-

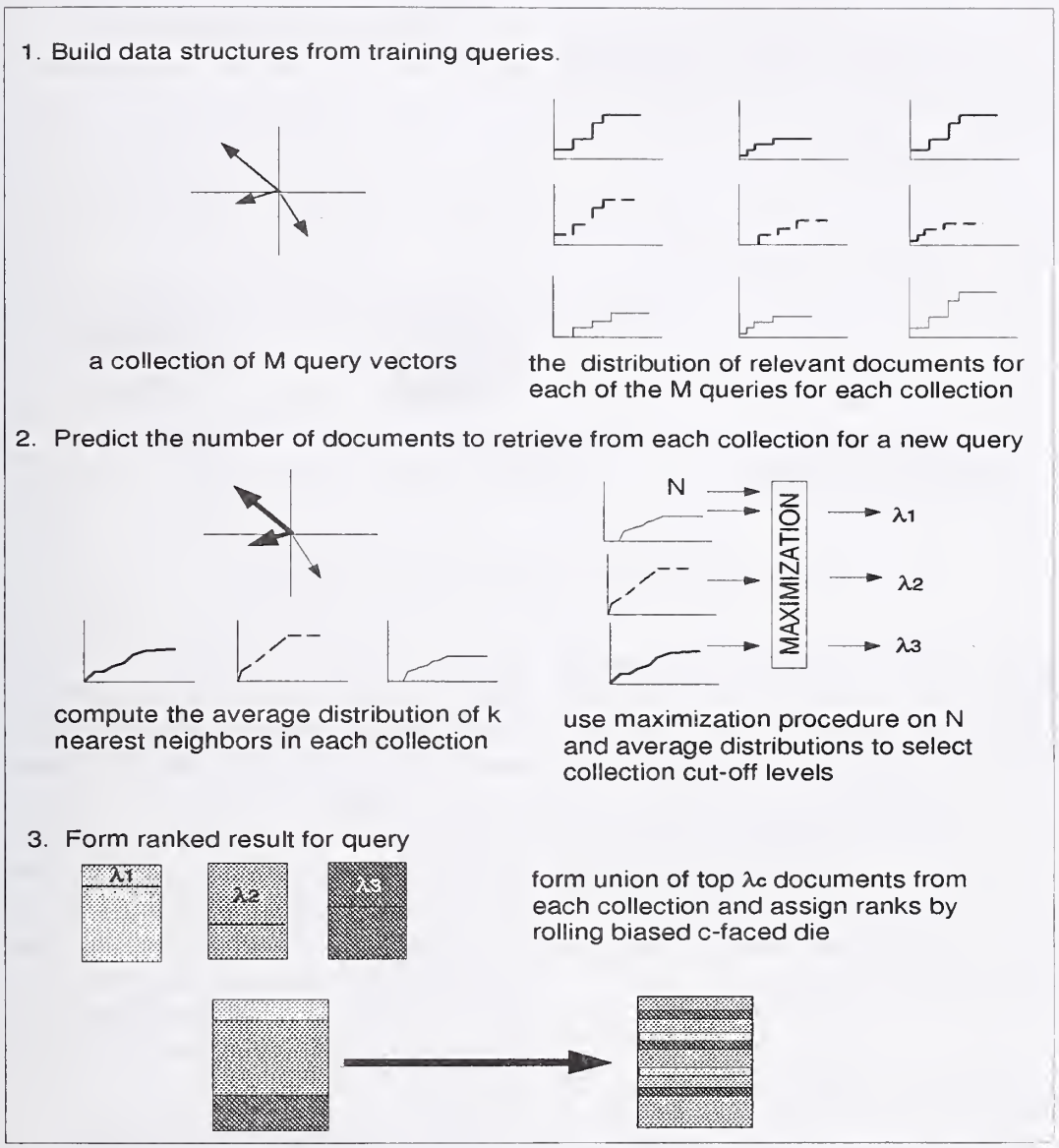


Figure 2: The relevant document distribution fusion strategy.

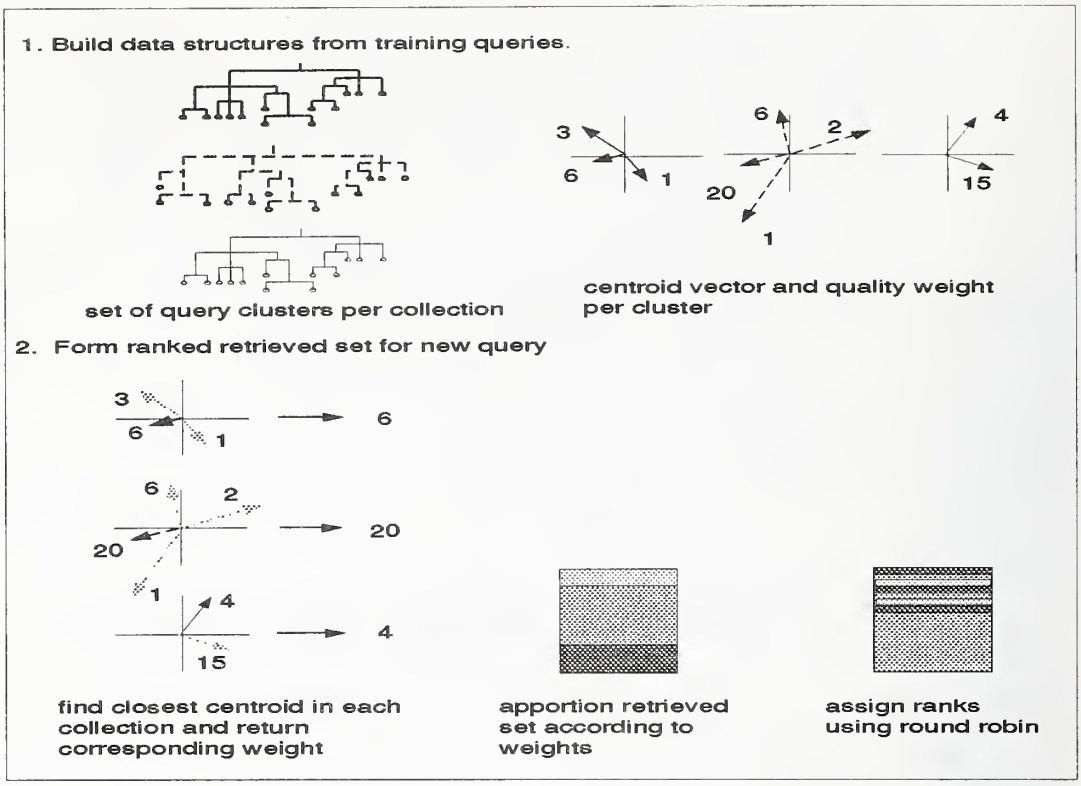


Figure 3: The query clustering fusion strategy.

ing method, for this fusion strategy we simply chose the documents round-robin style from each collection that had documents not yet inserted into the final ranking. While this total ordering also preserves each of the partial orders, it does not give preference to collections that are believed to have more relevant documents. Figure 3 summarizes the steps of the query clustering fusion strategy.

3 Retrieval results

The fusion results produced for TREC-3 were created using topics 1–150 as training topics and testing on topics 151–200. Queries were produced from the topics using the standard SMART [4] indexing routines on all of the fields except the Definition and Summary fields. The document set consisted of the documents on disk volumes one and two. Each of the five collections on those disks, AP, DOE, FR, WSJ, and ZIFF, was indexed separately, again using the standard SMART indexing routines. The retrieval runs on the component collections (that is, both the retrieval runs that were used for training and the runs that produced the rankings to be fused) used SMART's "`lnc`" weights for the document collections, "`ltc`" weights for the queries, and the inner product as the similarity measure (this is equal to the cosine of the vectors given the weights used) [5].

TREC run `siems1` was produced by the relevant document distribution fusion method. Four nearest neighbors were used to produce the average relevant document distributions. The nearest neighbors were selected using the cosine similarity metric on term-frequency weighted vectors.

Run `siems2` was produced by the query clustering fusion method. The training

queries were clustered using the Ward clustering method [8] with the reciprocal of the the number of documents retrieved in common in the top 100 documents as the distance metric. Final clusters were formed by cutting the resulting dendrogram at a distance of 1. Cluster centroids were created using term-frequency-weighted vectors produced from the "Concepts" field only. (Since the "Narrative" section of each topic begins with *A relevant document will contain...*, using the Narrative field to build the centroids caused clusters about document processing to absorb all the queries that were dissimilar to the other queries.) The parameter L, the number of documents used to create the weights associated with each cluster, was set to 100.

Table 1 shows how the two official TREC runs compared to the median of the other TREC submissions. The fusion techniques were radically different from the median values for precision after 100 and 1000 documents once: the relevant document distribution technique on Topic 151 obtained the worst results (5 relevant documents retrieved after 1000 documents). This topic concerns how inmates cope with prison overcrowding and is not very similar to any of topics 1–150. We did not control for the common phrasing used in the Narrative section in the relevant document distribution method, and as a result the four nearest neighbors of Topic 151 were Topics 65, 63, 35, and 33 which have to do with document processing, machine translation, information retrieval and the like. Using these queries to compute the number of documents to retrieve from each collection, the vast majority of the documents returned were from the ZIFF collection, which, unfortunately, has no known relevant documents about prison overcrowding. We hope to eliminate this effect by incorporating inverse query frequency weights (weighting terms inversely proportional to the number of training queries they appear in) when calculating nearest neighbors.

For comparison purposes, we ran queries 151–200 against a single collection consisting of all of the documents on disks one and two using `lnc-ltc` weights and inner product similarity. The non-interpolated average precision, the average precision after 1000 document retrieved, and the total number of retrieved relevant documents for the single collection run, `siems1`, and `siems2` is given in Table 2. The percentage difference of these measures for the fused runs over the single collection run is also included in the table. The number of retrieved relevant documents (and thus the average of the precision at 1000 documents) for the fused runs is comparable to the single collection run, especially for the query clustering fusion run. The non-interpolated average precision suffers in comparison to the single collection run, however, demonstrating that the single collection run is retrieving the relevant documents at smaller ranks. Clearly the round-robin method of imposing a total ordering on the final retrieved set is not very effective. Run `siems2` retrieved 600 more relevant documents over the 50 queries than run `siems1`, but has a smaller average precision.

Users are seldom going to want 1000 documents returned in a single reply, so to be useful the fusion techniques must approximate the behavior of a single collection run at small ranks. Table 3 gives the average precision of the single collection and fused runs when they are evaluated after 10, 30, 100, and 200 documents. The relevant document distribution fusion run (`siems1`) is within 11% of the single collection performance at these ranks, but the query clustering fusion run (`siems2`) is still harmed by the poor ranking algorithm.

Finally, earlier tests with the fusion

	Rel ret @ 100			Rel ret @ 1000		
	#<med	#=med	#>med	#<med	#=med	#>med
Rel doc dist (`siems1`)	25	3	22	25	4	21
Query clust (`siems2`)	41	2	7	24	5	21

Table 1: Number of queries whose precision after 100 and 1000 documents retrieved is less than, equal to, or greater than the median precision averaged over all TREC submissions.

	Non-interpolated average precision		Average precision at 1000		Total number relevant retrieved	
Single collection	.2643	—	.1218	—	6088	—
Rel doc dist (`siems1`)	.2088	−21%	.1076	−12%	5381	−12%
Query clust (`siems2`)	.1873	−29%	.1196	−2%	5981	−2%

Table 2: Effectiveness of fused runs compared to single collection run for 1000 documents retrieved.

	Eval at 10		Eval at 30		Eval at 100		Eval at 200	
Single collection	.5320	—	.4707	—	.3648	—	.2885	—
Rel doc dist (`siems1`)	.4720	−11%	.4187	−11%	.3268	−10%	.2607	−10%
Query clust (`siems2`)	.3940	−26%	.3467	−26%	.2820	−23%	.2434	−16%

Table 3: Average precision of official TREC runs evaluated at several small ranks.

	Eval at 10		Eval at 30		Eval at 100		Eval at 200	
Single collection	.5320	—	.4707	—	.3648	—	.2885	—
Rel doc dist, $N = 10$	.4400	−17%						
Query clust, $N = 10$	.4760	−11%						
Rel doc dist, $N = 30$	.4320	−19%	.4087	−13%				
Query clust, $N = 30$	.4560	−14%	.4247	−10%				
Rel doc dist, $N = 100$	.4800	−10%	.4320	−8%	.3340	−8%		
Query clust, $N = 100$	.4060	−24%	.3940	−16%	.3392	−7%		
Rel doc dist, $N = 200$	.5080	−5%	.4287	−9%	.3368	−8%	.2650	−8%
Query clust, $N = 200$	.4040	−24%	.3640	−23%	.3226	−12%	.2714	−6%

Table 4: Average precision for varying numbers of requested documents.

techniques suggest that they are more effective when smaller numbers of documents are requested [9]. We repeated the evaluations in Table 3 while varying N, the number of documents to be retrieved by the fusion algorithms (the official TREC runs used $N = 1000$). These results are given in Table 4. The diagonal entries in the table eliminate any ranking effect, and therefore the relative performance of the query clustering fusion technique is better there. The relevant document distribution fusion technique needs to request a moderate number of documents (100–200) to eliminate spurious effects caused by the volatility of the relevant document distributions of the training queries at very small numbers of documents retrieved. In general, the results in Table 4 provide evidence that these fusion techniques can approximate the effectiveness of a single collection run at the ranks that will be of interest to most users.

4 Conclusions

Our goal in TREC-3 was to explore the feasibility of combining the retrieval results of searches on independent collections into a single ranked list such that the effectiveness of the combined result is comparable to that of a single collection run. We have described two collection fusion techniques that are able to produce effective retrieved sets for the TREC-3 ad hoc queries. These techniques require only the ranked lists and relevance assessments of previous queries to produce the final retrieved set, and are applicable for a wide range of number of documents to be retrieved.

A user's perception of the quality of a result is strongly influenced by how well the documents are ranked, especially when large number of documents are returned. From the results presented here, it appears that the fusion techniques select enough documents from marginally relevant collections such that round-robin ranking (taking all documents ranked first in their respective collection, then all second, etc.) is not effective. Ranking documents by perceived quality of the collection from which they were drawn is much more effective.

While these fusion results are encouraging, it remains to be seen to what extent these results are applicable in retrieval environments other than TREC. For example, as Figure 4 demonstrates, a fusion strategy that is better than the single collection retrieval method for the TREC-3 ad hoc topics is simply to retrieve $N/2$ documents from the AP collection and the remaining $N/2$ documents from the *Wall Street Journal* collection. This method works well because there is only one topic, Topic 190, for which the majority of the relevant documents are not in the combination of those two collections [1]. Further experimentation is required to determine if our fusion methods will generalize to other environments.

References

[1] Brian T. Bartell, Garrison W. Cottrell, and Richard K. Belew. Automatic combination of multiple ranked retrieval systems. In *Proceedings of the 17th International Conference on Research and Development in Information Retrieval (SIGIR-94)*, July 1994.

[2] N.J. Belkin, C. Cool, W.B. Croft, and J.P. Callan. The effect of multiple query representations on information system performance. In Robert Korfhage, Edie Rasmussen, and Peter Willett, editors, *Proceedings of the*

[1] The majority of the relevant documents for Topic 190 is in Ziff. Both the relevant document distribution and the query clustering fusion methods take the majority of the documents they retrieve from the Ziff collection for Topic 190.

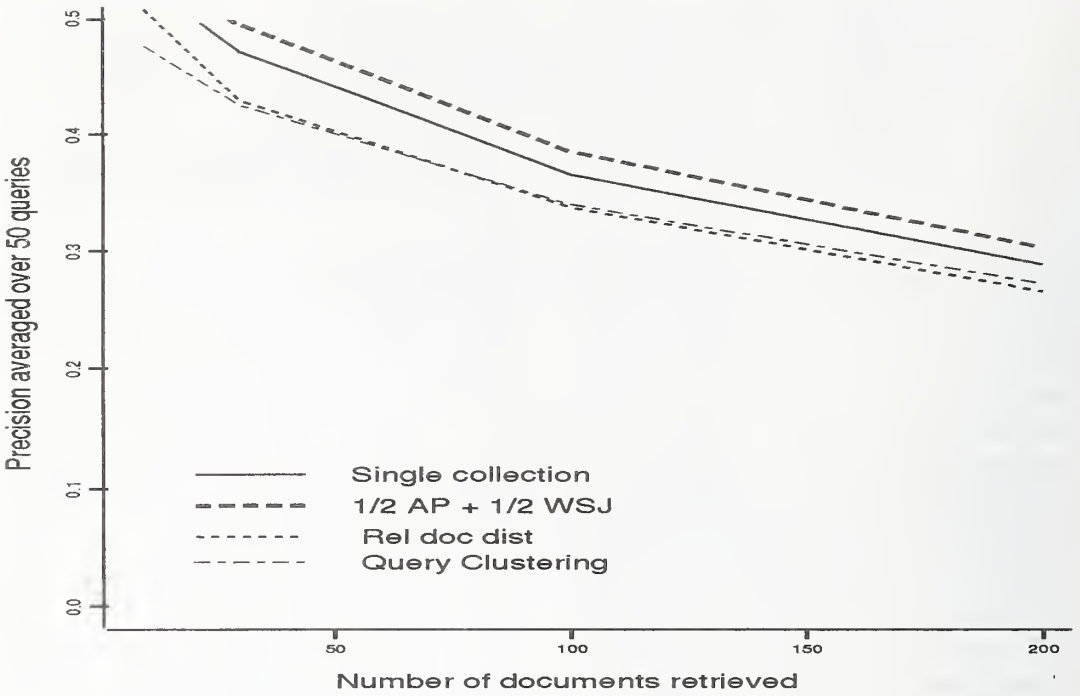


Figure 4: Effectiveness of four retrieval strategies.

Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 339–346, June 1993.

[3] N.J. Belkin, P. Kantor, C. Cool, and R. Quatrain. Query combination and data fusion for information retrieval. In Donna K. Harman, editor, *Proceedings of the Second Text REtrieval Conference (TREC-2)*, pages 35–44, March 1994. NIST Special Publication 500-215.

[4] Chris Buckley. Implementation of the SMART information retrieval system. Technical Report 85-686, Computer Science Department, Cornell University, Ithaca, New York, May 1985.

[5] 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.

[6] Susan T. Dumais. LSI meets TREC: A status report. In Donna K. Harman, editor, *Proceedings of the First Text REtrieval Conference (TREC-1)*, pages 137–152, March 1993. NIST Special Publication 500-207.

[7] Edward A. Fox and Joseph A. Shaw. Combination of multiple searches. In Donna K. Harman, editor, *Proceedings of the Second Text REtrieval Conference (TREC-2)*, pages 243–252, March 1994. NIST Special Publication 500-215.

[8] L. Kaufman and P. Rousseeuw. *Finding Groups in Data — An Introduction to Cluster Analysis*. Wiley, 1990.

[9] Ellen M. Voorhees, Narendra Gupta, and Ben Johnson-Laird. Learning collection fusion strategies. In preparation.

Combination of Multiple Searches

Joseph A. Shaw and Edward A. Fox
Department of Computer Science
Virginia Tech, Blacksburg, VA 24061-0106

Abstract

The TREC-3 project at Virginia Tech focused on methods for combining the evidence from multiple retrieval runs and queries to improve retrieval performance over any single retrieval method or query. The largest improvements result from the combination of retrieval paradigms rather than from the use of multiple similar queries.

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. In performing our TREC-3 ad-hoc retrieval experiments on the provided test collections, the results from both vector and P-norm [3] queries were considered in estimating the similarity 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. Our TREC-3 experiments built upon our TREC-2 experiments and focused more on determining where the improvements in combination were derived from rather than on evaluating different combination methods.

2 Index Creation

This section outlines the indexing done with the document collections 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 5000/125** 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 simply removed.

The resulting filtered text was tokenized, stop words were deleted using the standard 418 word stop list provided with **SMART**, plural removal stemming was performed, 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 **SMART** *ann* weighting scheme, defined as $term\_weight = 0.5 + 0.5 * \frac{tf}{max\_tf}$ proved to be the most effective in our TREC-2 experiments [5] and was used to evaluate all the queries in our TREC-3 results. The dictionary size for each collection was approximately 16 MB, while the document vector files ranged from 40 MB to 120MB (see Table 1).

3 Retrieval

3.1 Queries

All of the queries were created by the researcher 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), while two different sets of vector queries were created from the topics. The Title, Description and Narrative sections

Table 1: Collection statistics summary. Text, Dictionary and Document Vector sizes in Megabytes.

Collection	Text	Dict.	Doc. Vectors	Total Doc.s
AP-1	266	15.8	116.6	84678
DOE-1	190	15.7	95.3	226087
FR-1	258	15.7	50.9	26207
WSJ-1	295	16.0	120.4	98735
ZIFF-1	251	15.5	83.9	75180
D1	1260	N/A	467.1	510887
AP-2	248	15.7	107.1	79923
FR-2	211	15.5	40.1	20108
WSJ-2	255	15.9	101.7	74520
ZIFF-2	188	15.2	60.5	56920
D2	902	N/A	309.4	231471
Total	2162	N/A	776.5	742358

of the topics were used in the creation of all three sets of queries, while the P-norm query set and one of the vector query sets also contained a limited amount of additional terms added from the general knowledge of the query author to compensate for obvious omissions in the topic descriptions. The vector query set that included the addtional terms 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.

The five queries used for TREC-3 are similar in structure to our TREC-2 ad hoc queries, with the exception that one our TREC-3 vector queries contained terms that were not present in the topic descriptions, while the longer of our two TREC-3 vector queries did.

3.2 Individual Retrieval Runs

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 to simplify the merging of retrieval results across the various collections. The resulting similarity values were not based on collection statistics which would have differed for each collection. 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 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 2, and are equivalent to our TREC-2 runs with the exceptions in query construction noted above.

Table 2: 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 = 2.0

3.3 Combination Retrieval Runs

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. We explored several methods for combining the individual similarity values and found that simply combining the similarty values in a linear fashion– summing the similarity values–worked better than trying to select a given similarity value. This method of combination, called Comb-SUM in our TREC-2 report, [5] was used exclusively in our TREC-3 experiments.

4 TREC-3 Results

The procedure described above was used for our official TREC-3 ad-hoc results. We submitted two sets of results: one run labeled VTc5s which used the Comb-SUM method to combine all five individual runs, as per our official TREC-2 results, and one run labeled VTc2s which combined only the short vector query with the Pnorm query evaluated using a p-value of 1.5. The official results are reported in the last column of Table 3, in the rows labeled for the two runs.

Note that for all the collections the long vector query set containing terms not included in the topic performed better than the short vector query set. On a per-query

Table 3: Average Precision and Exact R-Precision for the five individual runs (Ad-hoc Topics 151-200).

Average non-interpolated Precision

Run	Disk 1					Disk 2				Both Disks
	AP	DOE	FR	WSJ	ZF	AP	FR	WSJ	ZF	
SV	0.2611	0.0320	0.0397	0.1957	0.0189	0.2355	0.0290	0.1811	0.0461	0.1340
LV	0.3139	0.0536	0.0547	0.2544	0.0459	0.2815	0.0357	0.2313	0.0588	0.1960
Pn1.0	0.3276	**0.0852**	0.0956	0.3240	0.0582	0.3038	0.0883	0.2840	**0.1019**	0.2062
Pn1.5	**0.3396**	0.0812	**0.1048**	**0.3435**	0.0599	**0.3201**	**0.0922**	**0.2915**	0.0974	0.2245
Pn2.0	0.3223	0.0758	0.1028	0.3283	**0.0651**	0.3120	0.0911	0.2894	0.0970	**0.2270**
VTc5s	0.3822	0.0855	0.1244	0.3866	0.0734	0.3604	0.1013	0.3322	0.1133	0.2914
Chg/Max	12.5%	0%	18.7%	12.5%	12.7%	12.6%	9.8%	14.0%	11.2%	28.4%
VTc2s	0.3944	0.0853	0.1125	0.3915	0.0732	0.3642	0.1005	0.3411	0.1192	0.3021
Chg/Max	16.1%	5.0%	7.3%	14.0%	22.2%	13.8%	9.0%	17.0%	22.4%	34.6%

Exact R-Precision

Run	Disk 1					Disk 2				Both Disks
	AP	DOE	FR	WSJ	ZF	AP	FR	WSJ	ZF	
SV	0.2996	0.0292	0.0406	0.2263	0.0225	0.2649	0.0217	0.2199	0.0336	0.2058
LV	0.3440	0.0562	0.0504	0.2833	0.0404	0.2907	0.0287	0.2427	0.0415	0.2607
Pn1.0	0.3500	0.0752	0.0903	0.3428	0.0575	0.3087	0.0751	0.2961	0.0799	0.2748
Pn1.5	**0.3528**	**0.0831**	0.0944	**0.3591**	**0.0623**	**0.3261**	0.0753	0.3043	0.0867	0.2855
Pn2.0	0.3453	0.0740	**0.0968**	0.3451	0.0554	0.3236	**0.0760**	**0.3082**	**0.0900**	**0.2895**
VTc5s	0.3947	0.0853	0.1181	0.3840	0.0633	0.3650	0.0824	0.3470	0.0938	0.3404
Chg/Max	11.9%	2.6%	22.0%	6.9%	1.6%	11.9%	8.4%	12.6%	4.2%	17.6%
VTc2s	0.4082	0.0875	0.1090	0.3938	0.0651	0.3728	0.0712	0.3487	0.0980	0.3538
Chg/Max	15.7%	5.3%	15.5%	9.7%	4.5%	14.3%	-5.4%	14.6%	13.0%	23.9%

basis, this held true for 39 of the 50 queries. Furthermore, the pnorm queries built from the long vector queries performed better on average than both sets of vector queries, though the improvement over the long vector query set was slight.

The VTc5s run shows a significant overall improvement over the five individual runs, and on a per query basis performed better than the best of the individual runs for 36 of the 50 topics. This matches the results obtained in our TREC-2 experiments. However, unlike our TREC-2 exerpiments, the VTc2s run combining only two of the five runs also performed significantly better than the best of the individual runs, and in fact performed better than the combination of all five runs overall and for several of the collections. On a per query basis, the VTc2s run performed better than the best of its two component runs for 42 of the 50 topics. However, the difference in overall performance between the two combination runs is not significant.

Further experiments involving all the possible combinations of two individual runs, as reported in Table 4 reveals further interesting trends. Combining two of the same type of runs, either both vector queries or two of the pnorm queries shows little improvement over the individual runs, and performs worse than the best of the two runs in many instances. However, combining one of the two vector queries with one of the pnorm queries always shows an improvement. This indicates that the primary source of improvements seen in the combination runs submitted for TREC-3 derives from the combination of retrieval paradigms and not simply from the use of multiple queries. This may be due to the similarity inherent in the five queries; combining two queries composed of two widely different sets of query terms may well result in significant improvements. But given a single set of query terms, it is still possible to achieve significant improvements by combining different retrieval paradigms.

5 Acknowledgements

This research was supported by the Virginia Tech Department of Computer Science and Computing Center. We also thank Russell Modlin, M. Prabhakar Koushik and Durgesh Rao for their collaboration during TREC-1.

Table 4: Average Precision for CombSUM runs combining two or three individual runs compared with combining all five individual runs. (Ad-hoc Topics 151-200).

Average non-interpolated Precision

Run	Disk 1					Disk 2				Both
	AP	**DOE**	**FR**	**WSJ**	**ZF**	**AP**	**FR**	**WSJ**	**ZF**	**Disks**
SV	0.2611	0.0320	0.0397	0.1957	0.0189	0.2355	0.0290	0.1811	0.0461	0.1340
LV	0.3139	0.0536	0.0547	0.2544	0.0459	0.2815	0.0357	0.2313	0.0588	0.1960
Pn1.0	0.3276	**0.0852**	0.0956	0.3240	0.0582	0.3038	0.0883	0.2840	**0.1019**	0.2062
Pn1.5	**0.3396**	0.0812	**0.1048**	**0.3435**	0.0599	0.3201	**0.0922**	**0.2915**	0.0974	0.2245
Pn2.0	0.3223	0.0758	0.1028	0.3283	**0.0651**	**0.3120**	0.0911	0.2894	0.0970	**0.2270**
SV-LV	0.3170	0.0473	0.0540	0.2568	0.0389	0.2849	0.0351	0.2310	0.0571	0.1865
SV-Pn1.0	0.3849	0.0894	0.1136	0.3734	0.0720	0.3479	**0.1079**	0.3264	0.1179	0.2826
VTc2s	0.3944	0.0853	0.1125	0.3915	0.0732	0.3642	0.1005	0.3411	0.1192	0.3021
SV-Pn2.0	0.3845	0.0831	0.1158	0.3871	0.0736	0.3608	0.0929	0.3351	0.1192	0.3004
LV-Pn1.0	0.3795	**0.0903**	0.1161	0.3766	0.0723	0.3516	0.0923	0.3296	0.1177	0.2941
LV-Pn1.5	0.3885	0.0844	0.1181	0.3966	**0.0775**	0.3654	0.0989	0.3429	0.1188	0.3104
LV-Pn2.0	0.3816	0.0823	0.1194	0.3890	0.0767	0.3634	0.0940	0.3395	0.1188	0.3100
Pn1.0-Pn1.5	0.3393	0.0846	0.0998	0.3405	0.0595	0.3191	0.0929	0.2940	0.0989	0.2183
Pn1.0-Pn2.0	0.3415	0.0823	0.1032	0.3449	0.0578	0.3216	0.0940	0.2909	0.0980	0.2236
Pn1.5-Pn2.0	0.3331	0.0797	0.1025	0.3353	0.0605	0.3194	0.0906	0.2916	0.0969	0.2267
SV-LV-Pn1.0	0.3953	0.0883	0.1155	0.3782	0.0751	0.3600	0.0961	0.3379	**0.1207**	0.3083
SV-LV-Pn1.5	**0.4056**	0.0841	**0.1253**	0.3947	0.0767	0.3704	0.0994	0.3488	0.1079	0.3204
SV-LV-Pn2.0	0.4029	0.0819	0.1241	**0.3987**	0.0706	**0.3722**	0.0969	**0.3490**	0.1090	**0.3219**
VTc5s	0.3822	0.0855	0.1244	0.3866	0.0734	0.3604	0.1013	0.3322	0.1133	0.2914

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 (TREC-1)*, D.K. Harmon (Ed.), National Institute of Standards and Technology Special Publication 500-207, Gaithersburg, MD, 319–328.

[5] Fox, E.A., Shaw, J.A. (1994). Combination of Multiple Searches. In *The Second Text REtrieval Conference (TREC-2)*, D.K. Harmon (Ed.), National Institute of Standards and Technology Special Publication 500-215, Gaithersburg, MD, 243-252.

[6] 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.

Okapi at TREC–3

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

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

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

1 Introduction

The sequence of TREC conferences has seen the City University Okapi IR system evolve in several ways. Before TREC–1 it was a very traditional probabilistic system comprising closely integrated search engine and interface, designed for casual use by searchers of bibliographic reference databases.

City at TREC–1

During the course of TREC–1 the low-level search functions were split off into a separate Basic Search System (BSS) [2], but retrieval and ranking of documents was still done using the "classical" probabilistic model of Robertson and Sparck Jones[7] with no account taken of document length or term frequency within document or query. Four runs were submitted to NIST for evaluation: automatic ad hoc, automatic routing, manual ad hoc and manual ad hoc with feedback. The results were undistinguished, although not among the worst. Of the ad hoc runs, the manual was better than the automatic (in which only the CONCEPTS fields of the topics were used), and feedback appeared beneficial.[1]

[1] We have only recently noticed that our TREC–1 (and probably also TREC–2) results would have been considerably worse had it not been that the system at that time could not handle documents longer than 64K, and so the longest few hundred documents in the database were truncated. The TREC–1 automatic ad hoc run redone on the full database (with cutoff at 200 documents) gives an 11–pt average of 0.10 (0.12), precision at 5 documents 0.37 (0.50); and at 30 documents 0.36 (0.42) (TREC–1 results in parentheses). This appears to be because the simple weighting scheme tends to favour long documents, particularly FR, few of which are relevant.

City at TREC–2

For TREC–2 the simple inverse collection frequency (ICF) term-weighting scheme was elaborated to embody within-document frequency and document length components, as well as within-query frequency, and a large number of weighting functions were investigated. Because of hardware failures few of the runs were ready in time, and City's official results were very poor. However, later automatic ad hoc and routing results, reported in [4, 5], were similar to the best official results from other participants. There were also some inconclusive experiments on adding adjacent pairs from the topic statements, and on automatic query expansion using the top-weighted terms extracted from the top-ranked documents from a trial search. Again, there was an interactive manual ad hoc run with feedback, but the results were far less good than City's best (unofficial) automatic run.

TREC–3

The emphasis in TREC–3 has been on

- further refinement of term-weighting functions
- an investigation of run-time passage determination and searching
- expansion of ad hoc queries by terms extracted from the top documents retrieved by a trial search
- new methods for choosing query expansion terms after relevance feedback, now split into:
  - methods of ranking terms prior to selection
  - subsequent selection procedures
- and the development of a user interface and search procedure within the new TREC interactive search framework.

The two successes have been in query expansion and in routing term selection. The modified term-weighting functions and passage retrieval have had small beneficial effects. For TREC–3 there were to be topics without the CONCEPTS fields, which had proved to be by far the most useful source of query terms. Query expansion, passage retrieval and the modified weighting functions, used together, have gone a long way towards compensating for this loss.

2 The system

Software

The Okapi software used for TREC–3 was similar to that used in previous TRECs, comprising a low level basic search system (BSS) and a user interface for the manual search experiments (section 7), together with data conversion and inversion utilities. There were also various scripts and programs for generating query terms, running batches of trials and performing evaluation. The main code is written in C, with additional material in awk and perl. The evaluation program is from Chris Buckley at Cornell.

Hardware

A single-processor Sun SS10 with 64 MB of core and about 12 GB of disk was used as the main development machine and file server. Batch processing was also done on two other Suns, a 4/330 with 40 MB and an IPX with 16. The SS10 is considerably faster than machines used for previous TRECs, particularly on disk I/O; this was important because the search-time passage determination procedure (section 4) was very greedy. In contrast to TREC–2, this time there were no very serious hardware problems.

Databases

Two databases were used: disks 1 & 2, and disk 3. In TRECs 1 and 2 all line and paragraph information was discarded. This time paragraph information had to be retained, and both for this reason and to improve readability for users of the interactive system most of the formatting of the source data was kept. (Some reformatting was done on the long lines of disk 1 WSJ.)

A 3-field structure was used, common to all source datasets. The first field was always the DOCNO and the third field contained all the searchable text, mainly the TEXT portions but also headline or title-like material for some datasets and documents. The second field was unindexed (and unsearchable) and so only (possibly) useful for display to users of the interactive system. It was empty except in the case of SJM, when it contained the DESCRIPT field; and the Ziff JOURNAL, AUTHOR and DESCRIPTORS fields.

3 Probabilistic model and basic procedures

3.1 Some notation

N: Number of items (documents) in the collection
n: Collection frequency: number of items containing a specific term
R: Number of items known to be relevant to a specific topic
r: Number of these containing the term
tf: Frequency of occurrence of the term within a specific document
qtf: Frequency of occurrence of the term within a specific query
dl: Document length (arbitrary units)
$avdl$: Average document length
BMxx: Best-match weighting function implemented in Okapi (see below)
k_i, b: Constants used in various BM functions (see below)

3.2 Weight functions

As in previous TRECs, the weighting functions used are based on the Robertson–Sparck Jones weight [7]:

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

which reduces to an inverse collection frequency weight without relevance information ($R = r = 0$). This is the BM1 function used in TREC–1.

In TREC–2 and the work that followed [4, 5, 6], we demonstrated the effectiveness of the following two functions:

$$w = s_1 s_3 \times \frac{tf}{(k_1 + tf)} \times w^{(1)} \times \frac{qtf}{(k_3 + qtf)}$$
$$\text{and} \quad k_2 \times nq \frac{(avdl - dl)}{(avdl + dl)} \quad \text{(BM15)}$$

$$w = s_1 s_3 \times \frac{tf}{(\frac{k_1 \times dl}{avdl} + tf)} \times w^{(1)} \times \frac{qtf}{(k_3 + qtf)}$$
$$\text{and} \quad k_2 \times nq \frac{(avdl - dl)}{(avdl + dl)} \quad \text{(BM11)}$$

s_i are scaling constants related to k_i (see [6]). nq is the number of query terms, and the "and" in front of the last component indicates that this document length

correction factor is "global": it is added at the end, after the weights for the individual terms have been summed, and is independent of which terms match.

In the course of investigating variant functions for TREC–3, we in effect combined BM11 and BM15 into a single function BM25, which allowed for a number of variations. The term frequency component is implemented as

$$\frac{tf^c}{K^c + tf^c} \tag{2}$$

with $K = k_1((1-b) + b\frac{dl}{avdl})$. Thus if $c = 1$, $b = 1$ gives BM11 and $b = 0$ gives BM15; different values of b give a mix of the two. The basis for BM11 was one of two possible models of document length (the "verbosity" hypothesis, [4]) which might be expected to exaggerate the document length effect; this is the justification for considering the mix.

A formula like equation 2, with $c > 1$, was suggested in [4], to give an s-shape to the function, as under some conditions the 2–Poisson model generates such a shape. Examination of a number of such curves generated by the 2–Poisson model suggested that c was related to K, and the formula $c = 1 + mK$, $m \geq 0$ was used in the experiments (in the event, m was largely ignored: see below). A scaling factor $s_1 = k_1 + 1$ was used, and the "global" document length correction was included. Also $s_3 = k_3 + 1$ was used, and where k_3 is given as ∞, the factor $s_3 \times qtf/(k_3 + qtf)$ is implemented as qtf on its own.

BM25 is referred to as BM25(k_1, k_2, k_3, b). It is always to be assumed that $m = 0$ unless stated.

The modified weight function seems able to give slightly improved results, at the cost of another parameter to be guessed. Non-zero m was not helpful. $b < 1$ can give some improvement. Values around 0.75 were usually used, sometimes with a higher k_1 than for BM11. Evaluation results for BM25 with various parameter values are not explicitly given in this paper.

3.3 Term ordering for feedback

In query expansion after relevance feedback in Okapi, terms from the relevant items are ranked according to some selection value which is intended to measure how useful they would be if added to the query. The formula usually used for this purpose (and in particular, the one used in TREC–1 and TREC–2) is the Robertson Selection Value (RSV), based on the argument in [8]. The formula given in that reference is $w(p-q)$ where w is the weight to be assigned to the term, p is the probability of the term occurring in a relevant document, and q is the probability that it occurs in a non-relevant document. For RSV, w is interpreted as the usual Robertson–Sparck Jones relevance weight $w^{(1)}$, p is estimated as r/R, and q is assumed to be negligible.

Given that the weighting function is now more complex, it seemed appropriate to consider some alternative interpretations. In particular, since within-document term frequency now figures in the weighting function, it should probably be part of the selection value (a good term is not just one which tends to occur in relevant documents, but one which tends to occur more frequently in relevant than in non-relevant documents). Although it is no longer obvious how to interpret the w in the $w(p-q)$ formula (since there is no longer a single weight for the term), a possible measure would keep the w as before, but reward terms that occur frequently in relevant documents by replacing r/R by $\sum_{\text{reldocs}} tf/R$ or rtf/R. This formula is referred to below as RSV2.

RSV2 seems to assume that the weight is a linear function of tf, as it would be with large k_1. However, as we have found that a smaller value of k_1 gives better performance, it seems likely that RSV2 is over-valuing large tf values. So we have also tried the (unweighted) average of RSV and RSV2, referred to as ARSV. Also, following earlier work by Efthimiadis [9], we have tried using r on its own as a selection value (referred to as the **r** criterion).

In the event, RSV2 and ARSV have not shown any advantage over RSV. The **r** criterion appears less good than the others.

In the past, this ranking of terms has been used to select the top n terms, where n is fixed (TREC–1) or variable between topics (TREC–2) (see section 6). Further development of these ideas, together with some results from early runs for TREC–3, suggested a more elaborate, stepwise term selection procedure.

3.4 Term selection and optimization

Theoretically, an alternative to term selection based on a ranking method such as those just described would be to try every possible combination of terms on training set, and use some performance evaluation measure to determine which combination is best. This is almost certainly not a practical proposition, but we have attempted a small step towards such optimization.

The principle was to take the terms in one of the rank orders indicated, and then to make a select/reject decision on each term in turn. This decision was based on one of the standard evaluation measures applied to the resulting retrieval: that is, a term was selected if its inclusion improved performance over that achieved at the previous step.

Although such a procedure is likely to be computationally heavy, it is not out of the question for a routing task. Full details of the procedure adopted are given in section 6.

4 Passage determination and searching

Some TREC participants have experimented with passage retrieval (e.g. [10]), with some success. In much previous passage retrieval work, however, passages are prespecified. The object of the City experiment described here was to investigate search-time *determination* of "good" passage(s) in each document by examining all, or many, of the possible sequences of text "atoms" (paragraphs, for example, or sentences).

There are at least three ways one could consider using passage retrieval.

- The retrieval status value of a whole document may be based on the score(s) of its best subdocument(s).
- In interactive searching the user could be presented (initially, or on request) with the best portions of a long document.
- In relevance feedback only the good portions need be used for feedback.

Only the first of these three uses has been tried in the present experiments.

Since the number of passages is nearly proportional to the square of the number $\mathcal{A}$ of text atoms in a document (and the total time to weight all passages is of order $\mathcal{A}^3$ unless the code is very carefully optimized[2]), it is not practical to use atoms which are too short in comparison with the length of a document. It was decided that the TREC atom should be a paragraph.[3] The Okapi database model was modified to incorporate paragraph and sentence information, and the TREC source disks reconverted in conformity with the new model. Paragraph identification was algorithmic, using indentation and/or blank lines in the source. Some of the more elaborate text structures, some of the FR documents for example, were not very accurately parsed; also, one-line paragraphs tended to become joined to the succeeding paragraph. Paragraph information for a document included length and offset, and the number of sentences in each paragraph. The mean length of a paragraph turned out to be not much more than 230 characters, with about 11 paragraphs in an average document and mean document length 2600 for both databases.

With this information it becomes possible to search any passage or sub-document which consists of an integral number of consecutive paragraphs. The system was set up so that the following could be varied:

- minimum number of atoms (paragraphs) in a passage (default 1)
- maximum number of atoms in a passage (default 20)
- number of atoms to "step" between passages (default 1).
- the weight functions depend on a notional "average document length" *avdl*; the true *avdl* (about 2600) is far too high for true weighting of short passages, so this parameter was sometimes reduced for the weighting of proper subdocuments only.

So as to avoid "passaging" documents with little chance of attaining a best passage weight in the top 1000, the first passage considered was the whole document. If this failed to come up to a certain threshold weight no further processing was done. By experiment, it was found that this threshold could be set to the weight of the 10000th whole document, where this was known, without losing more than a very small number of long documents with a good passage embedded somewhere. This reduced the number of documents considered by a factor of ten or more at the cost of a preliminary "straight" search for each topic. Finally, as a safety measure, it was also possible to set a maximum number of passages to be considered for a document. This was sometimes used, and it may have affected the final weights of up to about a dozen documents for some topics and conditions.

Results

A very large number of trials were done using topics 101–150 on the complete disk 1 & 2 database, first on single topics, then on topics 101–110, and finally on 101–150. Looking at individual documents suggested that the procedure behaved sensibly, but it proved difficult to obtain more than a small improvement over whole-document searching. Table 1 summarizes some results; see also table 3 for the effect of passage searching in combination with query expansion and table 7 for routing results. A minimum passage length of four paragraphs was a good compromise between speed and performance. Neither unlimited maximum passage length nor a fine granularity or large overlap gave more than a minimal improvement.[4]

In conjunction with query expansion, however, the improvement was considerably greater (see Section 5 and Table 3); it is not at all obvious why this should be so. For all the passage retrieval results given, the document weight was taken as the maximum of the weight of the best proper subdocument and the weight of the whole document.[5] We also tried linear combinations of

[2] If a maximum passage length is set this becomes $\mathcal{A}^2$

[3] The document with the most paragraphs is probably FR89119-0111, with about 8700. This can make about 3.9×10^7 passages of mean length 4350 paragraphs.

[4] In interactive searching it is unlikely that users would benefit from being offered passages longer than two or three screens.

[5] Where a maximum passage length has been set the whole document may not have been considered in the passage weighting.

Table 1: Automatic ad hoc results, passage retrieval, unexpanded queries: topics 151–200 TND only, disks 1 & 2

Passage min	step	max	*avdl*	AveP	P5	P30	P100	R-Prec	Rcl
1	1	8	1800	0.349	0.720	0.587	0.437	0.398	0.701
4	2	4	1800	0.346	0.720	0.586	0.439	0.391	0.692
4	2	4	2200	0.345	0.720	0.582	0.438	0.392	0.695
4	4	4	1800	0.344	0.724	0.583	0.437	0.386	0.691
4	2	12	1800	0.345	0.720	0.585	0.440	0.392	0.692
4	2	20	1800	0.345	0.716	0.585	0.440	0.392	0.692
4	2	24	1800	0.344	0.716	0.584	0.440	0.392	0.692
8	4	24	1800	0.342	0.728	0.589	0.434	0.387	0.687
8	4	8	1800	0.342	0.728	0.590	0.434	0.387	0.688
Non-passage result for comparison (official citya2)									
none				0.337	0.732	0.590	0.431	0.382	0.681
All runs BM25(2.0, 0.0, ∞, 0.75)									

best passage weight and document weight (as reported also in [10]); the best results from this were similar to those in Table 1, but achieved with different parameters.

5 Query expansion without relevance information

One of the experiments we did for the TREC–2 ad hoc was to attempt query expansion or modification without precise relevance information [4]. Query *modification* was done by reweighting the original query terms extracted from the topic statement on the basis of their distribution in the top-ranked documents retrieved in a trial search. There were no positive results from reweighting. For query *expansion*, the top documents from the trial search were used as the sole source of terms. Terms were selected in RSV sequence, with a limit on the number of non-topic terms. Any selected topic terms which occurred more than once in the topic statement were given a query term frequency component, with a value of 8 for k_3 (section 3.2).

A possibly–similar procedure appears to have been used with some success by at least one other TREC participant [11], but our best TREC–2 results showed only marginal and probably not significant improvement over the best from unmodified queries. Nevertheless, spurred by the relatively poor ad hoc results from topics with no CONCEPTS field, we decided to give it another try for TREC–3. This was unexpectedly successful.

Even if it has been considered, it may have been weighted with an *avdl* less than the true average document length, so the weight of the whole document considered as a passage may be less than its weight considered as a document.

Trial search and term selection

For all runs the trial search used the TITLE, NARRATIVE and DESCRIPTION fields of the topics with BM25(2.0, 0, ∞, 0.75). The top R documents were output and all terms other than stop and semi-stop terms extracted. These were F4-weighted on the basis of their occurrence in r of the R documents and the query term frequency adjustment applied. The resulting weight was multiplied by r/R to give an RSV value. The top T terms were then selected from the RSV-ordered list, subject to $RSV > 0$ and $r \geq 5$. Table 2 shows an example. (Table 4 illustrates a routing query for the same topic.)

Query expansion results

Table 3 gives a selection of ad hoc results. The official citya2 run is included for comparison: this used the same weighting method as citya1 but without expansion or passage-searching. Citya1 did better than citya2 on 35 of the topics. The results are fairly flat around 20–30 feedback documents and maxima of 20–40 additional terms. Passage searching improves results noticeably.

6 Automatic routing

Query term sources and weights

As in previous TRECs, for the official runs we used all the known relevant documents in the training collection (disks 1 & 2) as the sole source of query terms, ignoring the topic statements. After the official runs we repeated some runs restricting the term source to the subset of the officially relevant documents which appeared in the top 1000 documents retrieved by an ad hoc search. There have also been some experiments in which terms extracted from relevant documents have

Table 2: Topic 120, query expansion without relevance information: top 20 terms, $R = 30$, $k_3 = 8$

term	source	*qtf*	*n*	*r*	*wt*	RSV
terror	tit	6	8375	28	386	360
intern	tit	4	103519	25	144	120
privat	nar	3	37425	14	98	46
bomb	doc	0	9664	15	62	31
government...	nar	1	122323	23	40	31
threat	doc	0	15433	15	56	28
attack	doc	0	26544	16	49	26
state	doc	0	138351	22	35	26
militari	doc	0	33510	16	46	25
countri	doc	0	72086	19	40	25
act	nar	1	73037	19	40	25
offici	doc	0	117653	21	36	25
group	doc	0	124078	21	35	25
america...	doc	0	102703	20	36	24
consequ	nar	3	14062	7	98	23
libya	doc	0	1760	9	75	23
counterterror	doc	0	148	6	104	21
sponsor	doc	0	12950	12	53	21
iran...	doc	0	13455	12	52	21
econom	tit	3	64882	10	59	20
"Doc" means term does not occur in the topic statement.						

Table 3: Automatic ad hoc results, query expansion without relevance information: topics 151–200, disks 1 & 2

Feedback				Mean						
docs	max terms	r	passages	terms	AveP	P5	P30	P100	R-Prec	Rcl
30	40	> 4	lgth 1–8	51	0.401	0.752	0.617	0.478	0.421	0.741
30	40	> 4	lgth 4–24 by 1	51	0.401	0.740	0.625	0.476	0.422	0.739
(the above row is the official citya1)										
20	40	> 4	none	47	0.389	0.768	0.620	0.473	0.418	0.723
30	40	> 4	none	51	0.388	0.752	0.615	0.471	0.410	0.725
20	20	> 4	none	32	0.381	0.760	0.606	0.464	0.409	0.720
100	50	> 4	none	67	0.361	0.680	0.575	0.453	0.389	0.706
100	50	> 9	none	63	0.359	0.676	0.569	0.452	0.387	0.705
Unexpanded run for comparison (official citya2)										
0	0		none	34	0.337	0.732	0.590	0.431	0.382	0.681
All runs BM25(2.0, 0.0, 8, 0.75).										

been given additional weight if they occurred twice or more in the topic statement.

All non–stop and non–semi–stop terms were extracted, and given the normal $w^{(1)}$ weights (equation 1). Where a bonus was given for terms which occurred more than once in the topic statement this was done by multiplying the $w^{(1)}$ weight by $(k_3+1)\frac{qtf}{k_3+qtf}$ (see section 3).

Term ordering

Potential terms were first ordered according to some criterion based on their occurrence in relevant and nonrelevant documents and in the collection as a whole. The four criteria tried are described in section 3.3.

Obviously, for most topics there was a very large number of potential query terms. In previous TRECs we tried two methods for term selection from the ordered termlists. Both involved selecting the top T terms; either T was the same for all topics, or a value was chosen for each topic. Retrospective runs were done in which T was varied, from 3 upwards. Not surprisingly it was found that better results were obtained by choosing the best value for each topic, rather than the single value which gave the best average precision (for example) over all topics (see Table 4 in [4]). The former method (T optimized for each topic) was used for the cityr2 run in TREC–3 (Table 6), where T is between 3 and 100.

Table 4 illustrates the observation that, for a given topic, performance generally does not vary smoothly with the number of terms. This is part of the motivation for trying to discover more effective term ordering criteria. But the figures in Table 5 suggest that there is not much to choose between the criteria, at least when the same number of terms is used for each query. Further, when the same number of terms is used for each topic, there is very little difference in the averaged results as T increases from about 15 to 100 or more.

Table 6 shows that retrospective results can be improved by "individualizing" the number of terms selected for each topic.

"Optimizing" the queries

Since none of the term ordering criteria seems particularly effective, being swamped by the vagaries of individual terms in individual topics, it was decided to try some approach to the optimization of the term set for each topic with respect to some retrospective evaluation statistic, specifically a stepwise select-or-reject procedure as discussed in section 3.4. The procedure evolved after a number of informal trials (specifically to ensure that it would run in reasonable time, say an hour or two per topic) was as follows:

- termweights were not varied
- the top three terms were used, unconditionally, to start building the termset
- terms were considered one at a time, with no backtracking, in the sequence given by one of the ordering criteria
- after the first three terms, each successive term was added to the query and the query run (with a cutoff of 1000 documents) and evaluated against the training set; if the evaluation result satisfied some acceptance criterion relative to the result of the previous iteration the new term was retained, otherwise it was rejected
- the procedure ran until some stopping rule was satisfied (see below).

The stopping rule was triggered when one of the following conditions was satisfied:

- the number of terms in the set reached *maxterms*
- *maxbad* successive terms had been rejected
- the *lasttermth* term had been considered
- elapsed time exceeded *maxtime*

Acceptance criteria tried were increases in average precision or r-precision or recall. The most successful runs used average precision, with ties resolved on r-precision. Recall gave much more variability between topics, doing well on some and spectacularly badly on others. *maxterms* was initially set at 20, but since a majority of queries came out with the full 20 terms some later runs were done using a value of 30. *Maxbad* was always 8. *lastterm* was set so high (150) that it never caused the stopping rule to be triggered. *Maxtime* depended on the machine (and the time available), usually one or two hours per topic, although some runs with *maxterms* = 30 were given a higher value.

Automatic routing results

The "optimized" queries are much better than the other two types. Predictive and a few retrospective results for optimized queries are shown in Table 7. The procedure is computationally very demanding, sometimes taking several hours to produce a query on a Sun SS10 (excluding the time to extract and weight terms from the relevant documnets). At the time of writing work is in progress on a more efficient and perhaps sounder method of optimization, but no experiments have been done yet. The figures also suggest that there may be little difference in effectiveness between three of the four term-ordering criteria, but that the **r** criterion is less good.

Table 4: Topic 120, query terms extracted from relevant documents: effect of adding successive terms in ARSV order

Terms	wt	ARSV	AveP	P5	P30	P100	R-Prec	Rcl
terror	140	381						
airline	55	139						
secure	37	134	0.068	0.200	0.133	0.120	0.126	0.568
carrier	46	81	0.102	0.400	0.233	0.160	0.168	0.568
travel	48	71	0.126	0.600	0.267	0.200	0.210	0.632
intern	44	67	0.152	0.600	0.300	0.190	0.200	0.674
air	32	66	0.130	0.400	0.333	0.200	0.200	0.579
iran...	54	65	0.152	0.600	0.367	0.230	0.242	0.642
foreign	37	65	0.145	0.400	0.300	0.220	0.232	0.642
america...	35	65	0.145	0.400	0.333	0.230	0.242	0.632
libya	74	64						
bomb	59	56	0.125	0.400	0.233	0.190	0.190	0.653
flight	53	54						
faa	50	53						
passeng	58	53	0.102	0.400	0.233	0.230	0.232	0.611
pan	65	51						
airport...	51	48						
aviat	50	46	0.067	0.400	0.167	0.160	0.168	0.505
libyan	77	44						
europ...	39	42	0.088	0.400	0.200	0.180	0.179	0.579
(25 terms)			0.087	0.400	0.200	0.170	0.179	0.579
(30 terms)			0.088	0.400	0.133	0.190	0.200	0.579
(40 terms)			0.104	0.200	0.233	0.240	0.242	0.547
(50 terms)			0.117	0.400	0.267	0.220	0.232	0.600
(60 terms)			0.141	0.400	0.367	0.260	0.274	0.547
(75 terms)			0.121	0.600	0.300	0.210	0.221	0.537
(100 terms)			0.112	0.600	0.300	0.190	0.190	0.516
(125 terms)			0.129	0.600	0.300	0.220	0.210	0.526
(150 terms)			0.010	0.000	0.033	0.030	0.032	0.305

Table 5: Best routing results (retrospective) with same number of query terms for all topics

# terms	Criterion	AveP	P5	P30	P100	R-Prec	Rcl
18	RSV2	0.347	0.740	0.643	0.498	0.393	0.684
20	RSV2	0.351	0.724	0.656	0.497	0.398	0.681
30	RSV2	0.347	0.776	0.647	0.496	0.399	0.669
40	RSV2	0.353	0.756	0.653	0.510	0.400	0.674
50	RSV2	0.355	0.756	0.652	0.515	0.402	0.670
60	RSV2	0.356	0.764	0.655	0.516	0.400	0.667
75	RSV2	0.354	0.788	0.652	0.514	0.400	0.661
100	RSV2	0.346	0.788	0.651	0.509	0.394	0.652
40	RSV	0.351	0.780	0.655	0.511	0.396	0.669
50	RSV	0.356	0.784	0.667	0.516	0.397	0.668
60	RSV	0.354	0.800	0.654	0.513	0.395	0.664
75	RSV	0.350	0.824	0.663	0.510	0.393	0.657
15	ARSV	0.352	0.772	0.641	0.499	0.392	0.697
18	ARSV	0.346	0.724	0.630	0.497	0.398	0.684
20	ARSV	0.349	0.732	0.642	0.498	0.399	0.684
30	ARSV	0.346	0.756	0.655	0.498	0.397	0.666
40	ARSV	0.355	0.760	0.658	0.513	0.403	0.672
50	ARSV	0.354	0.752	0.649	0.513	0.397	0.670
60	ARSV	0.355	0.780	0.660	0.515	0.401	0.666
75	ARSV	0.355	0.800	0.659	0.511	0.399	0.660

Table 6: Best routing results using top T terms, T chosen to maximize AveP for a topic

Criterion	AveP	P5	P30	P100	R-Prec	Rcl
Predictive						
ARSV	0.371	0.660	0.584	0.457	0.393	0.752
RSV2	0.363	0.704	0.578	0.447	0.388	0.744
RSV	0.362	0.648	0.553	0.451	0.392	0.747
(the above row is the official cityr2 run)						
Retrospective						
RSV2	0.414	0.848	0.719	0.560	0.445	0.724
ARSV	0.410	0.840	0.715	0.561	0.442	0.723
RSV	0.409	0.856	0.707	0.562	0.441	0.719

Maxterms = 30 gives better results than *maxterms* = 20, and possibly a further small improvement might be obtained by setting *maxterms* still higher. A small topic term weight bonus ($k3 > 0$) appears to be beneficial. There was little difference between the weighting functions $BM25(2.0, 0.0, -, 0.75)$ and $BM25(0.8, -1.0, -, 1.0)$ (=$BM11$) (not shown in the table). Passage searching improves the results still further. Perhaps more interestingly, reducing the amount of training information by about 25% by using only the relevant records retrieved by one of the better ad hoc methods does not affect the results as much as might be expected; looking at individual topics, a few do substantially worse but some actually produce better results than with the full relevant set. (These are the rows marked "own rels" in table 7.)

7 Interactive routing

In comparison with TREC 1 and 2 where interactive searching was undertaken for ad hoc queries, TREC-3 routing queries constituted quite a different task and required different experimental conditions. The searchers were members of the City Okapi research team, who played the role of intermediaries. The official relevance judgements for the training document set served to simulate end-user relevance judgements in a realistic routing task.

The Appendix gives a factual description of the interactive system itself, the experimental conditions and the search process, as an addendum to the official system description provided elsewhere in these proceedings.[6]

7.1 The task and interactive process

The aim of the exercise was to generate an optimal query based on (a) information given with the topics (i.e. narrative, concepts and descriptions), and (b) terms extracted from relevant documents. The searchers made their own relevance judgements whilst interacting with the system using knowledge about the official relevance judgements. The interface was designed to facilitate query formulation rather than the creation of a set of relevant documents and searchers made use of the different information presented during the interaction to meet that end. One major improvement was that they no longer felt inhibited about examining documents at any stage in the search process, as they had previously under the 'frozen ranks' regime. A second was the ability to treat phrases as search terms, which were weighted as single terms and retained throughout the search, provided the relative weights were high enough. Thirdly, searchers were able to remove terms from term sets produced by automatic query expansion in order to eliminate 'noise' in the system generated term sets, e.g. numbers, proper names or other rare terms, which might be considered to have a disproportionately high weight.

[6] The weighting function used in the interactive system was BM11, as this was the best available at the time that system was implemented.

Initial query formulation

Search sessions consisted of three iterative phases. Firstly, in the initial query formulation phase the searcher could define different aspects of the topic with separate term sets and then join the sets to generate an initial query, from which a document set would be retrieved. The different commands and operators (define, join, adj) provided fine control over the elements of the search and to some extent enabled the searcher to structure the query. The 'adj' operator was used extensively to generate phrases: 232 times compared with the default operator BM11 (153 times).

Viewing results

The second phase, viewing results, involved the display of brief and full records. The brief record display gave a breakdown of the occurrence of query terms in the individual records and indicated the document source. The information on term occurrence was useful for multi-faceted queries, where the co-occurrence of two or more terms might be deemed important. However in most cases it simply provided a summary view since relevant terms could be combined in so many different ways. Likewise the document source served as background information but did not generally influence which full records were chosen for display.

In 75% of the searches the display of both the brief and full records was confined to the top 50 documents generated by the query; in only one instance did the scan go down to the 300+ level. As might be expected, the searchers' main objective was to achieve a reasonable precision amongst the top documents, rather than a high recall overall. However on occasions searchers did jump further down the ranking to check for more relevant documents, if the total number of officially judged relevant documents was known to be high.

Relevance judgements

Relevance judgements were made after viewing the full record, at which point the official relevance judgements were made available. However no distinction was made between documents not seen by the assessors and those definitely judged as not relevant, consequently documents tended to be read thoroughly even if marked with a 'no'.

Table 7: Some routing results with "optimized" queries

Conditions	AveP	P5	P30	P100	R-Prec	Rcl
Predictive (topics 101–150 on disk 3)						
maxterms = 30, $k_3 = 2$, passages (lgth 1–8)	0.430	0.728	0.615	0.480	0.449	0.791
maxterms = 30, $k_3 = 2$, passages (lgth 1–8), own rels	0.419	0.716	0.606	0.473	0.434	0.784
maxterms = 30, $k_3 = 2$, passages (lgth 4, o'lap 2)	0.426	0.724	0.611	0.480	0.449	0.788
maxterms = 30, $k_3 = 2$	0.425	0.724	0.603	0.483	0.447	0.788
maxterms = 30, $k_3 = 2$, own rels	0.417	0.716	0.610	0.475	0.436	0.775
$k_3 = 2$	0.412	0.692	0.605	0.474	0.436	0.779
maxterms = 30	0.414	0.744	0.621	0.482	0.443	0.762
maxterms = 30, own rels	0.405	0.696	0.602	0.467	0.428	0.754
passages (lgth 1–20)	0.415	0.716	0.621	0.477	0.439	0.779
	0.407	0.716	0.612	0.475	0.435	0.765
(the above row is the official cityr1 run)						
own rels	0.401	0.684	0.598	0.467	0.425	0.753
ARSV	0.406	0.692	0.599	0.476	0.438	0.770
RSV2	0.401	0.696	0.604	0.466	0.425	0.748
r	0.366	0.676	0.582	0.453	0.401	0.738
Retrospective (topics 101–150 on disks 1 & 2)						
passages (lgth 1–20)	0.500	0.944	0.789	0.618	0.502	0.794
	0.492	0.956	0.795	0.609	0.495	0.772
ARSV	0.481	0.928	0.769	0.609	0.490	0.768
RSV2	0.478	0.916	0.773	0.600	0.487	0.761
r	0.448	0.908	0.745	0.584	0.465	0.745
Ordering criterion RSV, $k_3 = 0$ and *maxterms* = 20 unless stated						
All relevant documents used (mean 239 per topic) except where "own rels" stated (176 per topic).						

Since official judgements were available, searchers were required to concentrate on selection of documents likely to be useful for term extraction. This reduced the conflicts experienced under TREC-2. The differences between official and searcher judgements are shown in Table 8.

Much of this difference can be accounted for by the fact that judgements were being made for a different purpose, but there were instances where the assessors appeared to have missed documents containing relevant sections interspersed with other material, or to have judged on the basis of simple term occurrence rather than query relevance.

Final query

In the third phase, usually after identification of 10 to 12 relevant documents, a new set of terms was generated by the system from the relevance feedback information. In most cases this constituted the final optimal expanded query, although the extracted term sets for 35 out of the 50 queries were modified by the searchers. After two or more iterations it became difficult to decide between similar term-sets. For 18 out of 50 topics, the last term set extracted was not the one chosen to form the final query. In two cases (topics 132 and 140) the initial user generated query produced satisfactory results without the need for term extraction and on at least one occasion the initial query was chosen as the final query in preference to the extracted term set.

7.2 Human intervention and the probabilistic models

In this round of TREC two features were introduced to provide more flexibility for interactive searching. The first allowed searchers to define phrases as query terms, which were treated as single terms in the term extraction process. The second provided searchers with the facility to delete candidate terms from an extracted term set. What effect this type of human intervention has on the probabilistic models is unknown. Some words occurred in derived term sets both as phrase components and as single terms, without any weight adjustment. In some instances searchers removed the single terms. Although searchers intuitively appeared to prefer to use phrases in formulating queries, the implications for the weighting functions need further consideration.

Similarly searchers were not aware of the full consequences of the deletion of individual terms from an extracted term set. One effect of extraction was to bring out more specific terms, including proper names. Searchers were sometimes doubtful about the potential

Table 8: Official vs Searcher Relevance Judgements

	Searcher			
Official	Y	N	?	Total
Y	573	78	0	651
N	154	446	2	602
Not Seen	48	86	0	134
Total	775	610	2	1387

value of such terms in routing queries, and tended to delete them in favour of more general ones. This highlights the artificiality of the task and the conflict of attempting to generate an optimal routing query which would be effective in another database and the very specific, often topical nature of some of the queries. Searchers were uncertain about whether to retain time-dependent names, events, and places which had been successful in a current context.

Another aspect of the weighting function which influenced human/system interaction relates to document length. The algorithm used this time brought short documents to the top of the list, with AP and WSJ sources being the most common. Such documents tended to be more homogeneous than those from other sources. This appeared to be a helpful property for both relevance judgement and term extraction.

7.3 Results

Output from the interactive system were queries like those shown in table 9.

Table 10 shows the results of applying these searches predictively and retrospectively. The predictive result should not be compared with the routing results (Table 7) because the routing queries were derived using a very large amount of relevance information, whereas the interactive queries had the benefit only of those few relevant documents found by the searchers. It is probably more fruitful to compare the result of applying the interactive searches retrospectively (i.e. the output which the searches would have obtained had they executed the final searches) with automatic ad hoc results. Since the searchers had access to complete topic statements the best comparison is with an automatic run using all topic fields.

8 Conclusions

8.1 Overview

In the course of participating in three rounds of TREC, the Okapi team has made very substantial progress. Internally, the system has been developed from an interactive search program into a sophisticated distributed tool for a wide variety of experiments. In terms of generally applicable research results, we have shown the benefits of continuing to work within the framework of the classical probabilistic model of Robertson and Sparck Jones. While the field of information retrieval continues to be strongly empirically driven (a tendency reinforced by the entire TREC programme), and any practical system has to make use of methods and techniques based on very different theories, arguments or observations, it remains possible for an effective system design to be guided by a single theoretical framework. Furthermore, even without such developments as regression analysis, the classical approach is capable of achieving performance levels comparable with the best systems in the world today.

8.2 Main conclusions from TREC–3 experiments

Term-weighting functions

The basic "rough model" methods developed for TREC–2, whose benefits were not apparent in the official results submitted to TREC–2 but emerged in subsequent experiments, have now been shown to be effective under the full rigour of the official TREC procedures. These methods allow the inclusion of within-document and within-query term frequency and document length into the Robertson–Sparck Jones relevance weighting model, and are applicable either with or without relevance feedback.

However, attempts at somewhat less rough models have shown only small benefit.

Passages

Run-time passage determination is feasible, if computationally expensive. In common with other investigators, we have shown some benefits for document retrieval, though not very large ones, from considering best-matching passages. It is likely that the most important uses of passage determination will be in interactive systems and in connection with feedback for query expansion.

Table 9: Topic 120: query from interactive search after relevance feedback

Term(s)	op	weight
guerilla		125
thailand		84
iraq...		66
holidai		63
iran...		58
econom consequ	adj	113
intern terror	adj	112
trade restrict	adj	110
travel		51
trade polici	adj	98
econom effect	adj	98
properti damag	sames	93
russia...		45
europ...		41
econom impact	adj	70
busi		23

Table 10: Interactive results

Conditions	AveP	P5	P30	P100	R-Prec	Rcl
Predictive						
BM11	0.250	0.560	0.445	0.345	0.302	0.648
(the above row is the official cityi1)						
Retrospective						
BM11	0.283	0.704	0.569	0.438	0.337	0.620
best automatic for comparison						
BM11, TCND topic fds	0.366	0.660	0.577	0.492	0.411	0.754
BM11, TND topic fds	0.294	0.600	0.517	0.435	0.350	0.659

Query expansion without relevance information

Somewhat to our surprise, query expansion based on the top ranked documents from an initial search, irrespective of relevance, proved to be of benefit with the shorter queries now in use. Furthermore, this technique combined effectively with passage retrieval.

Ordering and selection of expansion terms

We have not managed to improve on the term-ordering measures used in previous experiments. However, the stepwise selection or rejection of terms from the ranked list, although computationally expensive, proved very effective. This represents a return to our old friend, term dependencies.

Interactive searching

The reconciliation of the demands of interactive searching with the kind of controlled experiment represented by TREC has a long way to go. Although we have made a serious attempt at evaluating an interactive method within TREC rules, we do not believe that it is yet appropriate to try to compare interactive with non-interactive procedures.

8.3 Futures

The automatic methods developed for Okapi for TREC–3 depart somewhat from the principles on which Okapi was originally based, in that they involved some computationally heavy procedures (specifically those involved in query expansion for routing and in passage retrieval) which may not be feasible, as they stand, in a live-use system. One future line of work (within or outside TREC) will be to try to achieve similar levels of performance with simpler methods.

The scope for further performance improvements is debatable. It is possible that we and other TREC participants are approaching some limit of performance, or at least a point of diminishing returns. However, the real progress made over three years of TREC (after thirty years and more of research in information retrieval) does encourage the view that not all ideas have yet been exhausted. We have every expectation of further improvements in successive rounds of TREC and elsewhere.

In common with most other TREC participants, we have done much too little work on analysing individual or selected groups of instances (topics, documents, terms), to try to understand in more detail the circumstances under which our methods work or do not work. The time pressures of TREC participation and the scale of the operation do tend to discourage such analysis; at the same time, the TREC material provides a very great deal of scope for it, and there could be considerable benefits from it.

In many ways the most interesting and currently puzzling area is that of interactive searching. The apparent huge performance advantage of automatic over interactive methods may in various ways be an artifact of the methodology, but it most certainly deserves substantial further investigation. Given that most IR system use in the world today is interactive, the importance of achieving a better understanding of the phenomenon is hard to exaggerate.

References

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

[2] Robertson S E *et al.* Okapi at TREC. In: [1]. p21–30.

[3] *The Second Text REtrieval Conference (TREC–2).* Edited by D K Harman. Gaithersburg, MD: NIST, 1994.

[4] Robertson S E *et al.* Okapi at TREC–2. In: [3]. p21–34.

[5] Robertson S E and Walker S. Some simple effective approximations to the 2–Poisson model for probabilistic weighted retrieval. In: Croft W B and van Rijsbergen C J (eds). *Proceedings of the 17th Annual International ACM–SIGIR Conference on Research and Development in Information Retrieval, Dublin 1994.* Springer-Verlag 1994. (p232–241)

[6] Robertson S E, Walker S and Hancock-Beaulieu, M. Large test collection experiments on an operational, interactive system: Okapi at TREC. *Information Processing and Management* (to appear).

[7] Robertson S E and Sparck Jones K. Relevance weighting of search terms. *Journal of the American Society for Information Science 27* May–June 1976 p129–146.

[8] Robertson S E. On term selection for query expansion. *Journal of Documentation 46* Dec 1990 p359–364.

[9] Efthimiadis E N. A user-centred evaluation of ranking algorithms for interactive query expansion. *SIGIR Forum (USA), spec. issue.*, 1993, p146–59, (Paper given at: Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Pittsburgh, PA, USA, 27 June-1 July 1993)

[10] Callan J P. Passage-level evidence in document retrieval. In: Croft W B and van Rijsbergen C J (eds). *Proceedings of the 17th Annual International ACM–SIGIR Conference on Research and Development in Information Retrieval, Dublin 1994.* Springer-Verlag 1994. (p302–310)

[11] Evans D and Lefferts R. Design and evaluation of the CLARIT–TREC–2 system. In: [3]. p137–150.

Appendix: Addendum to System Description for Interactive Experiments

A System Description

A.1 Summary

Figure 1 (attached) shows a screen dump from a running system. The most important new functions in the TREC-3 interface were those for:

- User-controlled definition and manipulation of *term-sets*, reflecting the fact that our objective was to generate routing queries rather than sets of documents,
- Display of *brief records* giving an overview of a document-set and relevance judgements upon it, allowing searchers to assess the performance of the current query,
- Automatic retention and re-use of user-defined *phrases*, following new term extraction from relevant documents.

A.2 Interface style

The basic interaction was command-driven, but the interface was designed to run in an X-windows environment. One window was used for entering commands and receiving summary responses, another to show lists of brief records comprising a document set, and another to display a complete document. Brief record lists, and complete document displays, were piped through the Unix *less* utility, enabling repeated scrolling and rudimentary within-document searching. In displays of retrieved documents, any query terms are capitalized and surrounded by asterisks.

A.3 Usable features of the interface

The most important commands are described below, in roughly their expected order of use during a search session.

A.3.1 Define

Define a term-set using one or more key-words and operators. The default operator is standard Okapi BM11. Two other useful operators are:

- ADJ: adjacency, used to generate phrases. The presence of intervening stop-words is ignored.
- SAMES: words must occur in the same sentence.

Boolean operators AND, OR and NOT are available, but unlikely to be useful in the current context. The "define" command causes a numbered term-set to be created, whose details are retained by the interface client software. It reports back on the number of documents matched by the term-set, but does *not* generate a permanent document-set.

A.3.2 Join

Join two or more term-sets. This enables the creation of complex queries, comprising, for instance, two or more ADJ expressions. Again, no permanent document-set is created.

A.3.3 Docset

Generate a document-set by submitting a term-set as a query to the Okapi search engine. Information about this set is retained by the server.

A.3.4 Brief

Show brief records from a document-set. For each record, the following information is displayed:

- Document set-number,
- Sequence number of record in set (used by subsequent requests for full-record displays),
- Source (e.g. ZF, AP, FR, etc.),
- Weight ($w^{(1)}$),
- Summary of query terms occurring in the document,
- Both the "official" and the searcher's previous relevance judgements. N.B. These appear only after a full display of the relevant record has been requested.

A.3.5 Show

Show a full record. The text is piped through the unix *less* utility, enabling the user to scroll and search the document. Following this command, the system requires a relevance judgement—in the current context

this should reflect the searcher's estimate as to whether the document contains terms which will be useful at the query expansion phase. A running total is kept of the number of relevant and non-relevant records seen, to assist searchers in deciding when to attempt new term extraction.

A.3.6 Extract

Create a new term-set by extracting terms with a high frequency of occurrence in relevant documents. The top 50 such terms are identified; the top 20 are displayed in weight order. Existing user-defined phrases are submitted to the term extraction process and included in system-generated sets, if their occurrence in relevant documents warrants it.

A.3.7 Remove

Remove terms from a term-set. This operation can be applied to any term-set, but is most likely to be used on one generated by automatic query expansion. Its main purpose is to allow removal of "noise" terms from generated sets, e.g. numbers, typos, and other peculiarities which have a high weight because of their low frequency.

Following term removal, the remaining terms are promoted upwards by one place and the top 20 are again displayed. It is possible to remove a range of terms, including all those not currently displayed, so that a final query formulation is confined to terms actually seen by the searcher.

A.3.8 Results

Produce the final search output, i.e. the term-set which is to serve as the final query formulation.

B Experimental Conditions

B.1 Searcher Characteristics

The five (female) participants comprised two members of academic staff, one member of the research staff, and two postgraduate students. Their ages ranged from mid-20s to early 50s. Each searcher was allocated a batch of ten contiguous queries from the overall list, enabling some comparisons to be made about their search behaviour.

None of the searchers had any prior familiarity with the retrieval topics. They saw themselves as intermediaries, carrying out searches on behalf of end-users who were in the position to deliver relevance judgements. Three had existing experience of Boolean searching; one had a very detailed knowledge of the statistical principles underlying the okapi probabilistic search algorithm.

B.2 Task description / training

All but one of the searchers had participated in TREC–2 and were familiar with the objectives of the experiment—in fact the new interface for TREC–3 was based largely upon their proposals. In preparation for their task, they were given a demonstration of the interface, and undertook some dry runs with previous queries which were not part of the official training set. The system description (see section A) above was treated as a basic user guide, and on-line help was available to give the full syntax of the command language.

C Search process

C.1 Clock time

The figures given below are for on-line clock times only. On average about 5 minutes was spent off-line in thinking about the initial query; most work was done (as it should be in an interactive situation) by examining the effect of using search terms and functions with the database.

Mean	Median	Variance	Range
39.32	39.00	468.47	8–84

C.2 Number of documents viewed

In this context "Viewing" a document means displaying and reading its full text using the *show* command, and making a relevance judgement on it. Since brief record entries were listed 50 at a time, it was not practical to count them individually.

Mean	Median	Variance	Range
27.78	23.00	184.42	10–72

C.3 Number of iterations

At the start of the exercise, two possible forms of search "iteration" were identified:

- A *major* iteration was considered to involve all stages of the search from initial to final query formulation. A straightforward search was expected to require only one such iteration, where the initial query yielded enough relevant records for use by the later processes. A second or third major iteration would be counted when it was necessary to go back and reformulate the query in the light of documents examined.

- A *minor* iteration would involve a sub-series of actions, i.e. create a document-set, make relevance judgements, extract new terms from relevant documents, and create a new document set from the expanded query. A search might include two or three such iterations — repeated until the searcher

was satisfied that the current query was finding a good proportion of relevant documents.

Based on these original criteria, 11 searches consisted of two or more major iterations, in that new definitions were entered *after* term-extraction from relevant documents. In practice, however, new definitions involved addition of a few extra terms to existing queries rather than complete re-starts, so it is probably more accurate to say that there were *no* major iterations.

For reporting purposes, the use of the *extract* function was treated as the boundary between one minor iteration and another. Summary figures for the use of this command are as follows:

Mean	Median	Variance	Range
1.5	1.0	0.62	0–3

Following is a more detailed breakdown of number of queries by number of term extractions. In two cases, (topics 132 and 140), the initial query was considered to produce satisfactory results without the need for any extraction at all.

Queries	Extractions
2	0
28	1
14	2
6	3

C.4 Number of terms used

In this context the "initial query" is considered to be the term-set used by the first *docset* command; the "final query" is the one output following a *results* command. Note that overall 262 "terms" defined by users were in fact *phrases* specified with adjacency operators. System-derived terms were all single words, except for a few ad hoc phrases in the Okapi GSL.

	Mean	Median	Variance	Range
initial	8.06	7	16.47	2–20
final	16.86	20	34.37	3–28

C.5 Use of system features — summary table

Command	Mean	Median	Variance	Range
Define	8.58	7.00	13.84	4–18
Join	3.22	2.00	7.03	1–13
Docset	4.00	3.00	7.47	1–15
Brief	4.74	4.00	14.65	0–16
Show	27.78	23.00	184.42	10–72
Extract	1.50	1.00	0.62	0–3
Remove	3.22	2.00	16.01	0–15

C.6 Number of user errors

No data were collected under this heading.

C.7 Search narrative for query 122

Two attempts were made on this topic because a system failure occurred half way through the first one. On the second occasion the searcher entered fewer terms initially before creating and examining document sets, having found that some of her original candidates (e.g. *evaluation, marketing*) were taking the search in the wrong direction. However it is unlikely that the second attempt was any more successful than the first would have been.

The initial query term consisted of the words: *cancer drug develop test* which were all required to occur in the same sentence. Of the 12 top documents examined from this search, only 3 were officially judged relevant, although the searcher included 2 others as potentially useful for term extraction. As the following quotation illustrates, all the right terms may co-occur in a sentence, without really matching the query:

"The Food and **Drug** Administration has approved a **test** that can detect the sexually transmitted virus believed to be linked to the **development** of cervical **cancer**, a Baltimore newspaper reported Saturday."

The second query term consisted of the words *anti cancer laboratory*, which once again were required to be in the same sentence. This yielded 3 more officially relevant documents, and 2 others deemed useful for term extraction. The search went through two extraction / retrieval cycles, during which one further term *leukemia* was entered.

Extracted terms output for the final query were: *patients, tumors, cells, therapy, immune, agent, chemotherapy, surgery, Kaposi's, transplant, treatment, bacteria, approved, research.* Some extracted terms deleted by the searcher were: *area, New York, food, Kettering, aid, architecture, protein, infected.* One of the difficulties when examining documents was to sift out those mainly concerned with cancer from those mainly concerned with Aids, since these topics were often closely intermixed.

The logged search took just under 24 minutes, although another 10 minutes should probably be added to the overall time to account for the abortive first attempt. Altogether 48 documents were examined, of which only 9 were officially relevant but 23 were selected for term extraction by the searcher. Relevant documents which were found referred to the laboratory stage of anti-cancer drug development: evaluation and marketing were barely touched on, as reflected by the terms output for the final query. This was a disappointing result, which contrasted with others which appeared to be more successful. Looking back over the log, the searcher could see many points at which her strategy could have been improved.

Following is a breakdown of command usage on this search:

Define	Join	Docset	Brief	Show	Extract	Remove
5	2	2	1	48	2	13

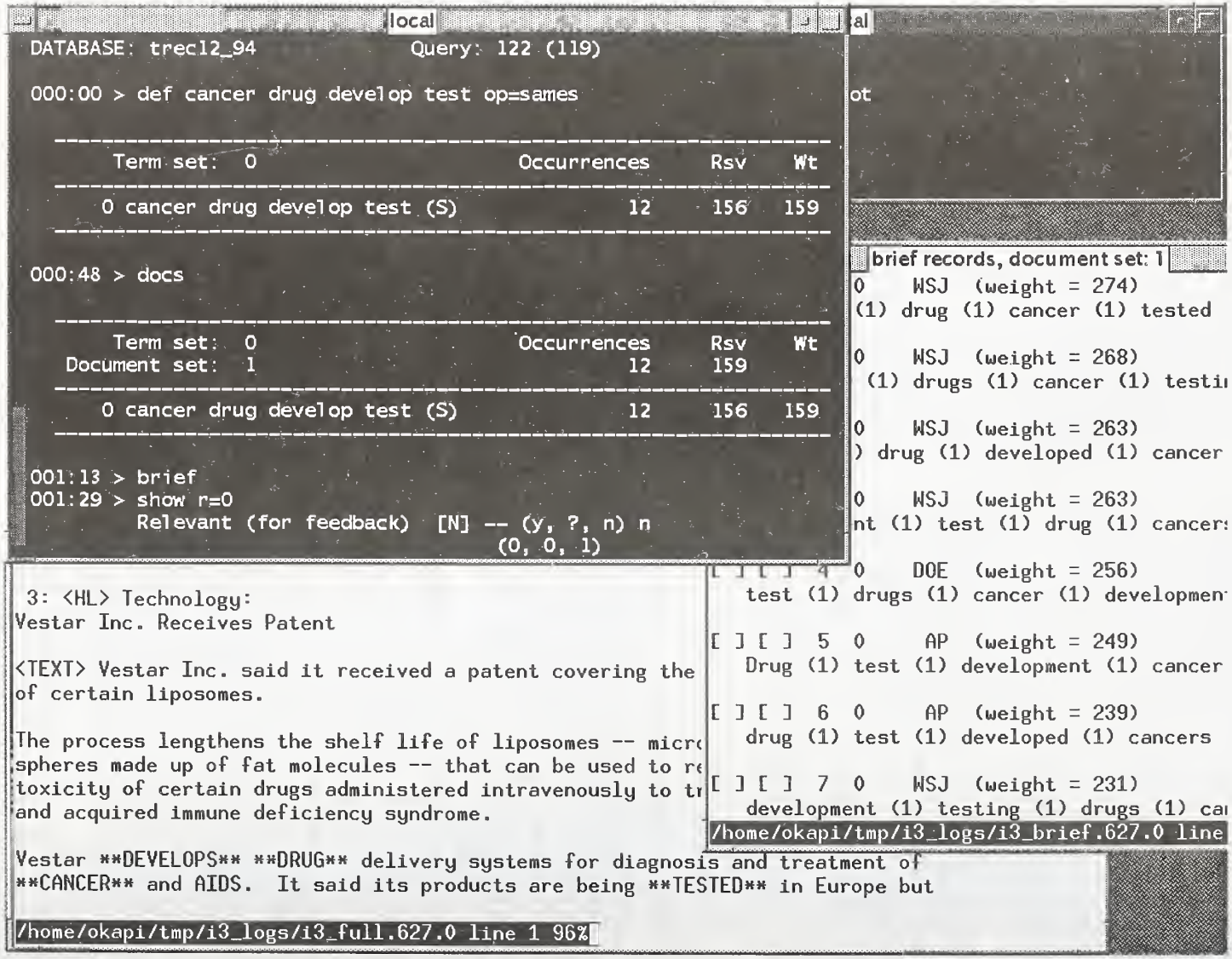


Figure 1: Interactive interface screen

Experiments in the Probabilistic Retrieval of Full Text Documents

William. S. Cooper
Aitao Chen

School of Library and Information Studies

Fredric C. Gey

U.C. Data Archive & Technical Assistance

University of California, Berkeley
wcooper@violet.berkeley.edu

ABSTRACT

The experiments described here constitute a continuation of a research program whose object is to find probabilistically sound, yet simple and powerful, ways of combining search clues in full-text retrieval. The methodology investigated for ad hoc retrieval is that of logistic regression, in which the retrieval rule takes the form of a regression equation fitted to learning data. Most of the variables used in the regression take the form of means rather than the more customary sums, and it is argued that this is logically preferable. Radical manual reformulations of the topics were tried out and found to boost retrieval effectiveness. For routing retrieval, an approach based on the Assumption of Linked Dependence, involving the extraction of relevance-associated stems from feedback documents, is investigated. One characteristic of this approach is that only a very minimal use is made of the original topic formulation.

Introduction

The problem of full text retrieval system design can be thought of as a problem in the combination of statistical clues. The design objective is to achieve as high a level of retrieval effectiveness as possible, consistent with reasonable theoretical and computational simplicity. As its contribution to TREC-3, the Berkeley group has continued its efforts to develop a simple, statistically sound probabilistic basis for text retrieval.

Because the emphasis is on finding a good logic of combination, the statistical clues used are all based on simple conventional frequency counts. This is not because we have objections to thesauri, parsing, phrase discovery, disambiguation, and other natural language processing or AI-like approaches. To the contrary, we feel that it is a virtue of the regression procedures explored here that they are more hospitable than most to the incorporation of additional clues. However, our experience and that of others suggests that an astute use of simple stem and document frequency information immediately lifts one to a high plateau of effectiveness. Further measures to attain slightly higher levels can then be incorporated if they are deemed worth the trouble.

Three experiments were performed for TREC-3 by the Berkeley group. Two of these were experiments in retrospective ('ad hoc') retrieval in which the group continued its earlier explorations of logistic regression as a simple but powerful tool for combining statistical evidence. Specifically, the advantages of separating the 'quantitative' from the 'qualitative' variables in the regression are examined in the experiments to be described. In its single routing retrieval experiment, the group has extended its earlier approach of estimating directly the crucial probabilistic quantities. A possibly novel aspect of the approach taken is that the search topic -- that is, the original natural language question itself -- is essentially ignored in the formulation of the internal search request used by the retrieval algorithm. Instead of the usual 'query expansion', we practiced vir-

tual 'query defenestration', in which only the already-judged documents were used to formulate the internal search specification.

The Ad Hoc Retrieval Rule

Our first experiment in ad hoc retrieval for TREC-3 (coded Brkly6) used fully automatic query formulation, in which the natural language topics were stoplisted and stemmed but not otherwise changed except for doubling of title stems. The computational rule used to assign a probability of relevance to each document in the collection -- and thereby to impose a search order on the collection for the user -- is easily stated. The logodds of relevance of any given document to any given query is estimated by the equation

$$\log O(R \mid Q,D) \approx c_0 + \sum_{i=1}^{6} c_i X_i \qquad (1)$$

where R is the event that the document is in fact relevant to the query, $O(R \mid Q,D)$ is the odds that the event of relevance has occurred, and

$X_1 = \frac{1}{M}\sum_{j=1}^{M} \log QAF_{t_j}$, the logged absolute frequency of occurrence in the topic Q of the match stems, averaged over all match stems. Thus if there are M stems in common between the topic and the document, X_1 is calculated by noting how many times each such stem t_j occurs in the query, taking the log of each of these M quantities, summing the logs, and dividing by M;

$X_2 = \sqrt{QL}$ i.e. the square root of the total number of all (non-stopword) stem occurrences in the topic;

$X_3 = \frac{1}{M}\sum_{j=1}^{M} \log DAF_{t_j}$ the logged absolute frequency of occurrence in the document of the match stems t_j, averaged over all match stems.

$X_4 = \sqrt{DL}$, the square root of the document length (analogous to X_2);

$X_5 = \frac{1}{M}\sum_{j=1}^{M} \log IDF_{t_j}$ the logged inverse document frequency (IDF) of the match stems t_j, averaged over all match stems;

$X_6 = \log M$, i.e. the logarithm of the number of match stems;

and where the values of the coefficients in the equation are $c_0 = 0.679$, $c_4 = -0.0674$, $c_5 = 0.223$, and $c_6 = 2.01$. The logodds-of-relevance figures estimated by this formula and used to rank the output documents are of course easily transformed into probability-of-relevance estimates to be displayed with the output documents for the guidance of the user.

The values of the coefficients $c_0, \ldots, c_6$ in Equation (1) were obtained by fitting the equation to the empirical data by means of a logistic regression analysis (Hosmer & Lemeshow 1989). The empirical data used as the basis of this analysis consisted of the values of X_1 through X_6, and a human judgement of relevance or nonrelevance, for each topic-document pair in the TREC corpus of relevance judgements. Upon supplying this matrix of data to a standard statistical package capable of performing logistic regression (BLSS), the regression equation (1) was obtained.

Actually, so many relevance judgements were available that only a subset of the available data were used. This subsample was constructed by applying, for each topic, a crude preliminary retrieval rule to rank the set of all documents whose relevance to the topic had been judged. The top 433 judged documents were then selected for each topic as the documents whose associated data would be used in the regression analysis. It was felt that by choosing high-ranking documents for the analysis, the resulting regression equation could be made more sensitive to the appropriate ordering of the documents near the top of the output ranking where the output order matters most. The number 433 was chosen arbitrarily as the number that would yield the maximum number of data points ($433 \times 150 \approx 65000$) that the statistical package could handle.

Retrieval based on Equation (1) is readily accomplished by storing stem-specific partial sums from the right side of Eq. (1) as indexing weights on the query stems and document stems. Then, at retrieval time, each match stem's query weight is added to its document weight, the mean of these sums over all match stems is calculated, and $c_6 X_6$ is added in. The procedure is roughly comparable in computational efficiency to the operations required for simple vector processing retrieval models, and the system used for our

experiments was in fact a modified version of the SMART system.

Separating Quality from Quantity

In past regression experiments the Berkeley group, and so far as we know all other IR experimenters attempting a regression approach (e.g. Fuhr & Buckley 1993), have taken as their regression variables various statistics *totaled* over all match stems. For instance, the regression variable used to represent the inverse document frequency information about a topic-document pair might be the *sum* of the (logged) inverse document frequencies of the match terms. Although this seems to have worked well, we have come to believe that it is not the soundest choice of variable form, and that it might be possible to do better.

To see the problem with the use of totals, suppose that a retrieval clue has been scaled (linearly transformed) in such a way that zero and negative values of it can occur. For instance, if it has been transformed into a standardized statistical variable, zero might represent its mean value. Now, how a variable has been scaled should not matter, but under a policy of totaling over match terms it does. Suppose that for a given query, document A has just one match stem and its value for that variable is zero, while document B has several match stems and their values for the variable are also all zero. The total contribution for the variable is zero for both documents, yet B is likelier to be relevant to the topic than A simply because it has more match terms. In other words, information about number of match terms can be lost or distorted when totals are used.

The underlying difficulty is that use of totals confounds match stem *quality* with the number of match stems or match stem *quantity*. The corrective measure that has been taken by the Berkeley group is to calculate the values of the variables, not as totals, but as *means*, of their values for the different match terms. Then, to incorporate the information about the number of match terms, a separate new variable is introduced that deals with nothing but the match term count. This policy overcomes difficulties of the sort just indicated when zero and negative values of clues are allowed, and lessens distortions even when they are not. It can be viewed as a policy of separation of quality variables from quantity variables. For instance, in Eq. (1) the quality variables are X_1 through X_5, while the quantity variable is X_6.

It can also be argued that separating quantity from quality tends to make the resulting regression equation more robust -- that is, less sensitive to systematic errors or changes in the data and more amenable to transfer into different collections or changed conditions. To see this, consider the effect on retrieval of multiplying all IDF values (before logging) by a factor of ten (perhaps because of a systematically misplaced decimal point). With a little study it can be seen that the output ordering of the documents imposed by Eq. (1) will remain unchanged. This is very different from the effect under the conventional totaling procedure, in which the tenfold increase would cause the influence of the IDF variable to overwhelm that of the other variables, and would greatly affect the ordering. Changes by an additive constant (before logging) also cause less disruption of the output ordering when the quality/quantity dichotomy is maintained. Thus there is a lessened sensitivity to any (linear) rescaling of the data from whatever cause.

Manual Query Modification

There has been interest recently in the question of whether it is possible to improve retrieval results significantly by means of expert manual modification of the original natural language query submitted by the user. The Berkeley group's second ad hoc retrieval experiment (Brkly7) attempted to cast some further light on this issue. The second experiment was identical to the first in all respects, including the use of Eq. (1) without modification, with the sole exception that the natural language topics were transformed by human intermediaries into new word lists before being submitted for stoplisting and stemming.

The human intermediaries were instructed to work intuitively to include in the list any words likely to appear in documents relevant to the topic at hand. They were encouraged to consider including, for a start, any content words and phrases that appeared in the original topic, though there was no compulsion to include those that seemed unhelpful. To expand the list further, they were instructed to add to these any synonyms, broader or narrower terms, or indeed any other words they could think of that might enhance the retrieval.

The intermediaries were allowed to exploit any personal knowledge of the topic they might happen to

possess. For instance, for the topic on oil spills, a knowledgeable intermediary might add to the list 'Exxon' and 'Valdez'. As a further source of indicative words and phrases, the intermediaries were encouraged to make crude Boolean searches in other data bases and, when obviously relevant documents were found, to scan them for potentially useful new terms. The Newspaper Articles Database, freely searchable via the University of California's MELVYL catalog system, was used for this purpose. Most of the searches were made in its New York Times database, though a few were made in its Los Angeles Times or Washington Post files. Care was taken not to do any searching in sources, such as the Wall Street Journal, which were also included in the TREC corpus, as this would have contravened the TREC guidelines. The WordNet thesaurus was made available to the intermediaries but for the most part they declined to use it, finding the Boolean searches in the auxiliary collections to be a more fertile source of helpful terms. Finally, the intermediaries were encouraged to write into their lists double or multiple instances of any terms that they considered especially vital to the satisfaction of the information need.

Except for the first two topics, on which the intermediaries spent a long time getting used to the process, about twenty to thirty minutes were spent modifying each topic. This was thought to be an amount of time that a zealous librarian or other intermediary might be willing to spend on occasion to try to improve an important search request. The idea was to make the hypothetical conditions as favorable to modification as possible, to see if manual intervention is of any avail even under especially advantageous circumstances.

To illustrate the results of this process of topic modification, here is the reconstructed form of topic 154, which concerns oil spills:

> *oil spills oil spills oil spills oil spills oil spills oil spills oil spills oil spills oil spills oil spills oil spills major offshore drilling holding tank spills light crude tons gallons slick Exxon Valdez Prince William Hazelwood Unocal Guadalupe Shetlands San Juan Puerto Rico Rubis Lyria Mobil Shell Francisco Ashland Newport Hudson Latouche Delaware Narragansett Rhode Huntington*

In the first ad hoc retrieval experiment, which involved no manual query modification, the average precision over all relevant documents for all fifty routing topics was 27.8%. In the second ad hoc experiment, which differed only in the manual query modifications, it rose to 37.1%. This substantial increase suggests that extensive reworking of natural language requests by intermediaries can indeed, under favorable circumstances at least, considerably enhance a system's retrieval effectiveness.

Topic-Free Routing Retrieval

For its third TREC-3 experiment (Brkly8) the Berkeley group designed a routing retrieval algorithm along the following, we suspect novel, lines. It is usual to base all document retrieval, including routing retrieval, on some sort of search specification derived from the original natural language topic. In many routing systems such a search specification is then expanded in some way by the inclusion of terms taken from documents known from past user feedback to be relevant to the topic. However, under the present Berkeley approach the original topic wording can be ignored completely, provided enough feedback is available.

The unusual character of this procedure is apparent from the fact that if the original natural language topic were written in a language different from the language of the collection, it would not matter, because the method is not dependent upon finding matches between topic stems and document stems. Rather, the internal search request vocabulary is drawn entirely from previously judged documents. It is not essential that the method be applied in this radical, absolutely topic-free form; and in fact the Berkeley experiment did make one slight gesture in the direction of exploiting the original topic wording, as will be explained. However, the essence of the method is that the search specification be constructed independently of the original topic expression to the extent allowed by the amount of available feedback.

As the first step in the construction of the internal search request for a topic, every stem that occurred in at least one document in the set of judged documents for the topic was tested to see whether it was statistically related to the property of relevance to the topic. This was done by constructing a four-cell table for each such stem and conducting a chi-square test on it. The two rows of each stem's table corresponded to the presence or absence of the stem in a document, while the two

columns corresponded to relevance or nonrelevance to the topic of a document. Thus the upper left cell contained the number of documents that (i) contained at least one occurrence of the stem and (ii) had been judged relevant to the topic. The sum of the quantities in all four cells was the total number of available judged documents for the topic.

The chi-square test showed which stems had the property that their presence in a document was statistically associated (either positively or negatively) with the relevance of the document to the topic. All stems for which the hypothesis of independence (i.e. lack of association) could be rejected at the 0.05 level of statistical significance under the test were collected and used in the search specification for the topic; all others were discarded. The stem sets so collected ranged in size from 300 stems for Topic 131 to 4,114 stems for Topic 109. The mean stem set size was 1,357.

Next, for each topic, a weight was calculated for each stem that had survived the chi-square screening for that topic. This weight was an estimate of the surplus logodds (i.e. the logodds over and above the prior logodds) that a document under consideration is relevant to the topic, given that the stem is absent from the document. The motivation for using such estimates as weights will be explained in the next section; for the moment we merely state the formula used to calculate the weight. It is:

$$weight_s = \log\frac{r_s + r\ /\ (n + r)}{n_s + n\ /\ (n + r)} - \log\frac{r}{n} \qquad (2)$$

where r_s is the number of documents in which the stem is absent that have been judged relevant to the topic, n_s is the number of documents in which the stem is absent that have been judged nonrelevant to the topic, r is the total number of documents that have been judged relevant to the topic, and n is the total number of documents that have been judged nonrelevant to the topic.

For stems occurring in the original topic, weights were calculated in accordance with the foregoing formula whether the stems passed the chi-square screening test or not. In cases where these weights turned out to be negative, their magnitudes were arbitrarily increased by a factor of five. This was the only concession in the direction of using the original topic wording. It was made because it was found to improve the retrieval effectiveness of the scheme slightly. However, the tiny size of the gain suggested that this refinement is unimportant when sufficient relevance data are available.

To illustrate the form of the internal search requests, here is a small segment of the list of weighted stems constructed for Topic 102, which is about U.S. strategic defense:

decid	0.236631
declin	0.103601
decoy	-0.162831
defend	-5.052930
deform	-0.014523
defray	-0.014523
degree	0.083949
deliv	0.140666
delt	-0.072431
demand	0.144963

A negative weight on a stem indicates that the stem's absence from the document reduces the chances that the document is relevant. Thus a stem with a negative weight is really a positive clue. For instance, the excellent clue 'defend' has a negative weight -- a large negative weight, in fact, because it occurs in the topic wording and hence had its weight multiplied by five.

The in-principle retrieval rule for estimating the logodds of relevance of a document to a topic is simply to add up the weights of all stems in the stem list for the topic that are absent from the document. In practice, however, it is quicker and produces the same result to add up the weights of all the stems in the list that are present in the document, and subtract this sum from the prestored grand total of all the weights in the stem list for the topic.

While the Brkly8 routing entry produced respectable recall and precision, there remains some question that the size of the expanded queries may be chosen as too large. The Cornell group has conducted experiments (Buckley, Salton, Allan 1994) on massive query expansion and concluded that the point of diminishing returns was an expansion by about 300 terms. Since the Berkeley tests show performance enhancement for terms chosen up to the 0.1 probability level of the Chi square test on the training set, and the 0.05 level was chosen for realistic computational times, we see evidence that evaluation on a training set may lead to the phenomenon of *overtraining* when the expansion is applied to the new test document set.

Theoretical Motivation of the Routing Rule

The foregoing retrieval rule is based on 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. 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 property of the topic-document pair, for example some property of the document that makes it either more or less likely to be relevant to the topic.

From the Assumption of Linked Dependence the following probabilistic modelling equation can be derived:

$$\log O(R \mid A_1, \ldots, A_N) = \log O(R) + \sum_{i=1}^{N} [\log O(R \mid A_i) - \log O(R)] \qquad (3)$$

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).

The properties A_i in the modelling equation can in general be any clues; for instance the occurrence in a document of a particular stem could serve as such a clue, as could its nonoccurrence. In the experiment just described, each property A_i of interest is the absence from the document of some particular stem from the constructed internal stem-list for the topic at hand. The weighting formula (2) estimates, for any given stem-absence, the corresponding quantity in square brackets on the right side of the modelling equation (3). The expressions $r / (n + r)$ and $n / (n + r)$ added into the numerator and denominator of the first fraction prevent either from taking on a value of zero. The retrieval rule stated in the preceding section is simply the computation of the summation indicated in the right side of (3). The initial term $\log O(R)$ appearing after the equal sign in (3) is omitted from the computation because it was found to affect the result only insignificantly in practice.

Either the presence of a stem in a document or its absence may be used as a retrieval clue. For TREC-2, as well as in preliminary TREC-3 experiments, the Berkeley group used both types of clues in tandem. Under that approach, for each stem there were two weights in the search query, one to be added into the retrieval score in case the stem was present in the document, and the other to be added in in case the stem was absent. However, subsequent experimentation showed that the two sets of weights were largely redundant, in the sense that use of just one type produced almost as effective retrieval as use of both. Stem-absence worked somewhat better than stem presence, and was virtually as effective by itself as when used in tandem with stem presence clues. So, to simplify the procedure, the stem-present weights were eliminated.

The Assumption of Linked Dependence, like all simplifying assumptions, introduces errors into the logodds computations. When the number of properties involved is large, as is the case in the present application, this distortion can become severe, taking the form of grossly exaggerated estimates for the logodds of relevance of documents near the top of the output rankings. This exaggeration does not affect the ordering of the output, but it makes the probabilities of relevance displayed for the user unrealistic. To counter this tendency, each sum calculated by the retrieval rule was divided by the square root of the number of stems in the query list for the topic of the search. This crude remedy suggested itself because of its success in compensating for similar distortions in an ad hoc experiment performed by the Berkeley group for TREC-2.

Table I: Brkly8 Routing Entry
Comparison between estimated and actual relevant/non-relevant

recall level	num docs	actual relevant	actual non-rel	actual probability	estimated probability	expected relevant	expected non-rel
0.1	1054	683	371	0.6480	0.7725	814.21	239.78
0.2	1098	683	415	0.6220	0.6547	718.96	379.03
0.3	1351	683	668	0.5056	0.5061	683.81	667.18
0.4	1685	683	1002	0.4053	0.4178	704.14	980.85
0.5	2188	683	1505	0.3121	0.3585	784.57	1403.43
0.6	1983	683	1300	0.3444	0.2610	517.55	1465.45
0.7	1952	683	1269	0.3499	0.2290	447.04	1504.96
0.8	2609	683	1926	0.2618	0.1882	491.08	2117.91
0.9	4330	683	3647	0.1577	0.1276	552.55	3777.45
1.0	6940	681	6259	0.0981	0.0344	238.92	6701.08

Evaluation of Probability Estimates

As in TREC2, the Berkeley text retrieval research group is concerned with the accuracy of probability estimates obtained through its methods. If one can, with confidence, display these estimates to the user, along with the ranking of documents presumed relevant to the topics, the user has the added flexibility of choosing where to stop the search. For example, if the probability estimate is 0.1, and the estimate has been shown to be realistic, the user can realize that, on average, (s)he will have to examine ten documents to find the next relevant one.

One way to test the calibration of probability estimates is to take the entry of 50,000 query-document pairs and sort them in descending order of magnitude of probability estimate. Using the results of the judged set of documents from the TREC3 relevance judges, one can attach relevance judgments for those pairs which have been judged. Of the 50,000 pairs submitted for Brkly8, 25,190 pairs were in the judgment set. This list can then be divided into deciles of recall, and the mean estimated probability of relevance for each level of recall can be computed. Since we know the actual number of relevant for each decile, the actual probability of relevance for that decile can be computed as well. From the estimated probabilities, expected numbers of relevant and non-relevant documents can also be computed. Table I shows the results for the ten levels of recall for Brkly8.

In evaluation of logistic regression models, Hosmer and Lemeshow (1989) have shown that the two sets of 10 pairs of actual relevant/non-relevant and estimated relevant/non-relevant documents in each decile of recall can be used to assess the fit of the model. In particular, the fit is demonstrated to follow a χ^2 distribution with 8 degrees of freedom. Computing this statistic for Table 1, we have

$$\chi^2 = 1327.69$$

and a probability value of less than 0.0001. Similar computations for Brkly6 and Brkly7 result in the following statistics:

$$\chi^2 = 861.99 \quad (Brkly\,6)$$

$$\chi^2 = 758.10 \quad (Brkly\,7)$$

also with probability values of less than 0.0001. For further details on the actual formulae to compute the χ^2 statistics, the reader is referred to Gey (1993).

These results would lead us to reject the null hypothesis that the models correctly fit the data, and perhaps lead us to conclude that further refinement of these models is in order.

Summary

The Berkeley entries to TREC3 have built upon previous research using logistic regression and the principle of linked dependence for combination of evidence. For ad hoc retrieval, we suggest that a separation of quality and quantity variables is appropriate in a regression-based methodology, with the quality variables being combined by taking means rather than sums. The

Brkly7 entry demonstrates that judicious manual intervention to reconstruct queries can, under favorable circumstances, increase retrieval effectiveness considerably. Specifically, the inclusion of proper noun *particularization* of examples of relevant real-world objects (companies, persons, geographic place names) seems to move the query closer to the relevant set.

For routing retrieval, the Brkly8 entry shows that, given a history of relevance judgments, internal requests can be constructed virtually without reference to the original topic wording and that query terms can be combined by a simple retrieval rule based on the Principle of Linked Dependence. A χ^2 test on degree of dependence of the term to relevance is used to rigorously define the query expansion term set. However results seem to suggest that an "overtraining effect" may come in to play when moving from the training judgment set to the test collection.

Finally, preliminary tests to assess the effectiveness of the probability estimates generated by these methods indicate that additional refinement of the models will be necessary before the probabilistic system can, with confidence, display these estimates to the user.

Acknowledgements

We are indebted to Ray Larson for helpful systems advice, as well as to the several colleagues already acknowledged in our TREC-1 and TREC-2 reports. Edwin Cooper, Emily Cooper, and Nate Kurz donated their services as intermediaries in the second ad hoc experiment. 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.

References

Buckley, C.; Salton G.; Allan, J. The Effect of Adding Relevance Information in a Relevance Feedback Environment. *Proceedings Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* Dublin, Ireland 272-281. July 1994.

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

Cooper, W. S.; Dabney, D.; Gey, F. 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; June 1992.

Fuhr, N.; Buckley, C. 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; 1993.

Gey, F. C. *Probabilistic Dependence and Logistic Inference in Information Retrieval.* PhD Dissertation, Graduate Division, University of California, Berkeley, May 1993.

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

Routing and Ad-hoc Retrieval with the TREC-3 Collection in a Distributed Loosely Federated Environment

Nikolaus Walczuch*, Norbert Fuhr,
Michael Pollmann, Birgit Sievers
University of Dortmund, Germany

Abstract

In this paper, we investigate retrieval methods for loosely coupled IR systems. In such an environment, each IR system operates independently on its own document collection. For query processing, an agent takes the query and sends it to the different IR systems. From the answers received from theses servers, it forms a single ranking and sends it back to the user. In the work presented here, we examine different retrieval methods for performing routing and ad-hoc-queries in such an environment. For experiments, we use the TREC-3 collection and the SMART retrieval system.

1 Introduction

In the past, IR systems have been considered as running on a large, fairly static document collection held on a single machine. However, there is a growing need for the development of IR systems that operate in a distributed environment:

- Document collections for similar subject fields are set up at different locations with different systems and retrieval methods. In retrieval, a user would like to query all relevant databases at once, without bothering about the underlying differences.
- With the improvements of local computing power and network access, the coupling of local and non-local IR data bases becomes possible.

*On leave from: Universidad de los Andes, Merida, Venezuela

So far, little research has been performed that focuses on retrieval in a distributed environment. More precisely, we must distinguish between different types of distribution. We use the term "distributed IR systems" in the same way as in the field of databases: On the logical level, such a system behaves in the same way as a non-distributed systems, the differences are only at the physical level (e.g. distributed storage, replication and partitioning of data, special query processing methods and communication protocols). In contrast, loosely coupled systems also behave differently on the logical level. This approach offers the advantage that it is possible to couple existing systems, without the need to change the software. A simple example for loosely-coupled IR systems is the design of the WAIS system (see [Kahle et al. 93]). The underlying protocol Z39.50 allows a client to do parallel searches in IR systems.

In this paper, we only consider the latter type of system, although some of the methods investigated also may be feasible for distributed IR systems. We call the different IR systems "servers". A user submits a query to a client, which in turn connects to an "agent". The agent's task is to communicate with the different servers. It sends the query to the servers, receives the different answers and then forms a single ranking and sends it back to the user. Some of the servers contacted may in fact also be agents, which in turn know further servers or agents that help processing the query. An example structure of this type is depicted in figure 1.

An example for such a structure is the SFgate system described in [Pfeifer et al. 95] which implements a WAIS gateway for WWW clients and acts as an agent in that it can access several servers in parallel. In the original WAIS system, there are only clients and servers, where the client also must perform the agent's task.

For performing retrieval in such an envioronment, a number of problems have to be solved:

- How do agents and servers communicate?
- What information should the agent have about the different servers that it can access?
- How can the agent select servers that provide relevant documents for the current query?
- Which retrieval methods should be provided by the servers in order to give optimum support for retrieval in loosely coupled IR data bases?
- How should the information agent merge incoming ranking results from different sources (with possibly unknown quality)?

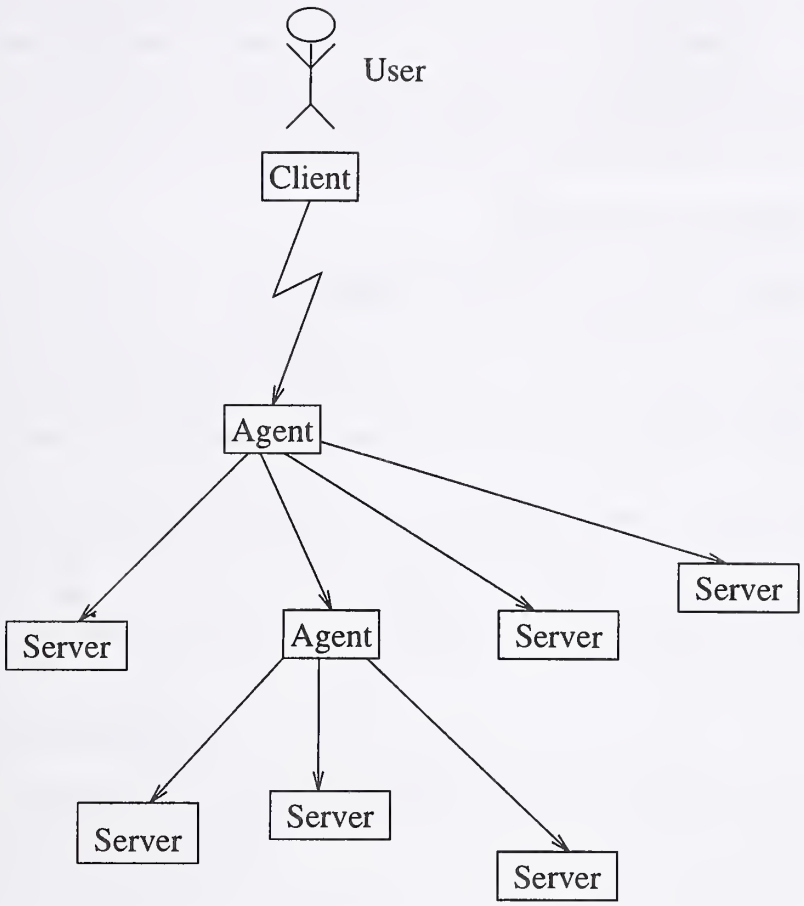


Figure 1: Retrieval with loosely coupled IR systems

- What retrieval quality can be archieved in such an environment, or what is the loss in retrieval quality in contrast to retrieval on a single large data base?

As the TREC collection originally consists of five different document collections, it is quite natural to use this collection for experiments with a distributed environment. In contrast to the very general case, we only consider a single agent contacting five servers, each running on one of the subcollections. As experimental system, the SMART retrieval system was used for simulating retrieval methods in this environment. So we did not really implement a distributed environment, since our focus here is on the investigation of retrieval methods for such an environment. Once good retrieval methods have been devised, an architecture for federated IR systems supporting these methods can be developed.

In the remainder of this paper, we first describe the basic concepts of the retrieval methods investigated, followed by the description of the experiments performed. The results are discussed in Section 4, and finally, we come to the conclusions and give an outlook on further work.

2 Basic concepts

2.1 Distributed routing retrieval

For the routing task, we have queries and a set of documents with relevance judgements. As new documents arrive, the system has to assign relevance status values (RSVs) to these documents, by considering these documents one by one.

Since there is relevance feedback data available, learning methods can be applied in order to improve the representation of queries. As a result, the weights of the query terms are modified. Furthermore, we can apply query expansion methods in order to add terms to the original query.

In a distributed environment, the agent collects relevance feedback data and performs the learning process, possibly in cooperation with the servers. Having improved the query representation this way, it sends it to the servers. Each server processes this query against its incoming documents, and sends the requested number of documents (or all documents with a RSV exceeding a predefined threshold) back to the agent. The agent combines the streams from the different servers into one single output stream and sends it to the client.

In our experiments, we assume that we can compute overall document frequencies, thus being able to compute query term weights in the same way as in a single collection environment. This can be achieved by additional communication overhead, where the agent collects the collection frequencies of the query terms from all servers in the learning phase. On the other hand, document indexing weights are independent of the overall document frequencies of terms.

2.2 A Concept for distributed ad-hoc retrieval

For ad-hoc queries, only the query formulation, but no relevance feedback data is given for the current query.

After receiving a new query from a client, the agent forwards the query to the servers (possibly after preselecting the servers relevant for this query). Each server processes this query on its database and sends the prespecified number of answer documents to the agent. Now the agent merges these

results into one single ranked output list. For performing this task, there are at least three different possibilities:

1. The agent takes the RSVs computed by the servers as absolute values and forms the output by merging the input lists according to decreasing RSVs.

2. The agent collects all the documents received for a query and creates a new ranking for this set of documents, e.g. by treating this set as a small document collection.

3. If the server receives additional collection information from the servers, a better ranking can be produced: Assume that each server also sends its document frequencies for all query terms. Then the agent can compute the overall document frequencies and use them as improved query term weights for ranking the set of documents received.

In our experiments, we investigated the last two possibilities.

3 Experiments

For the description of the document and query indexing methods given below, we use the following notations:

q_k	query
d_m	document
t_i	term
tf_{im}	within-document-frequency (wdf) of t_i in d_m
$\max tf_m$	maximum wdf tf_{mi} of all terms in d_m.
idf_i	inverse document frequency of t_i

Based on these parameters, standard SMART indexing routines were applied ([Salton & Buckley 88]): The augmented term frequency atf_{im} is computed as

$$atf_{im} = 0.5\left(1 + \frac{tf_{im}}{\max tf_m}\right),$$

and the logarithmic term frequency ltf_{im} is defined as

$$ltf_{im} = 1 + \ln tf_{im}.$$

Furthermore, we always use cosine normalization for document indexing, i.e. the weights are normalized such that the sum of squares of the indexing weights in a document is 1.

3.1 Distributed Routing

As pointed out in 2.1, we can use the same method of query term weighting as in a single collection environment. On the other hand, document indexing cannot be based on overall document frequencies. For this reason, we choose to use no collection frequency information at all for document indexing. This method is also suited very well for highly dynamic collections.

For experimental setting, the following steps were performed:

1. The training data set D1 first was indexed with the "anc" method using only augmented term frequency and cosine normalization. Alternatively, we also applied the "ltc" method based on logarithmic term frequency in combination with cosine normalization.

2. The query set Q3 (queries 101 to 150) was weighted with the "atc" method using the product of augmented term frequency and inverted document frequency .

3. For internal verification of our methods, data set D2 was used by applying the same document indexing method as for D1.

4. Since the document indexing weights are independent of any specific collection, we were able to use a single database containing the whole TREC collection for our simulation experiments.

5. For learning from feedback data, we produced statistics of terms and phrases only from the relevant documents of D1 for the queries of Q3.

6. Using this statistics, we ran query expansion of Q3 using the standard Rocchio method (see e.g. [Salton & Buckley 90]). Let $\vec{q}$ the original query vector, $\vec{r}$ the centroid of the relevant document vectors and $\vec{n}$ the centroid of the nonrelevant ones, then we computed a improved document vector according to the formula

$$\vec{q'} = 8 \cdot \vec{q} + 16 \cdot \vec{r} - 4 \cdot \vec{n}.$$

   For query expansion, only a certain number of terms finally was considered in the computation of $\vec{q'}$. For this purpose, we used "percent expansion" with different parameters for words and phrases. That is, from all terms occurring at least once within a relevant document, only a certain percentage was considered in the final query; for this purpose, terms were ranked according to the number of relevant documents in which they occurred.

7. By using only relevant documents of D1 for expansion statistics we tried to avoid influence of terms from non-retrieved but nonrelevant documents in the occurrence statistic and in the expansion process.

Table 1 shows the experimental results for query set Q3 and document set D2. Obviously, document indexing method "lnc" gives better results than "anc". We also varied the weighting factors for phrases in order to optimize retrieval quality. For query expansion, percent expansion with different parameters was tested and compared with the case of a fixed number of expansion terms.

The parameter combination of the last line in table 1 was used for the official run. For computing the query term weights, the combination of the document sets D1 and D2 was used. Running these queries on the test data set D3 produced the official run `dortR1`.

3.2 Distributed ad-hoc retrieval

For simulating distributed retrieval, we tested the cases 2 and 3 from above. For this purpose, document sets D1 and D2 were split according to the 5 different sources, thus forming 5 separate databases. The documents in each database were indexed with the method "ltc", i.e. the product of logistic term frequency and inverse document frequency, followed by cosine normalization. The idf weight was computed from the local collection frequency only. Then, for each query from the set Q4, the following steps were performed:

1. For each database, the query was indexed with the "ltc" method (using local collection frequency only). With this query, the top ranking 1000 documents were selected from each database.

2. With the 5 results, a new temporary document collection was formed. In a distributed environment, this document base would be constructed by the agent.

3. For testing the case 2 from above, the temporary collection was reindexed with the "lnc" method, and the query with Q4 with "ltc" (based on frequency information from the temporary collection only). Retrieval with this combination produced the results for the official run `dortD2`.

4. For case 3, we assumed that we had collection frequency information for each query term from all servers. Thus, we could apply the "ltc" indexing method for both documents and queries based on frequency information from the whole collection. This way, the official run `dortD1` was produced.

index. method	word exp.	phrase exp.	word wt.	phrase wt.	11p-average
anc	20%	8%	1.0	0.5	0.4051
anc	16%	8%	1.0	0.5	0.4098
anc	14%	7%	1.0	0.5	0.4121
anc	13%	6%	1.0	0.5	0.4125
anc	14%	6%	1.0	0.5	0.4127
anc	300	50	1.0	0.5	0.4130
anc	350	50	1.0	0.5	0.4141
anc	14%	7%	1.0	0.7	0.4147
anc	14%	6%	1.0	0.7	0.4149
anc	14%	5%	1.0	0.7	0.4150
anc	14%	4%	1.0	0.7	0.4155
anc	14%	3%	1.0	0.7	0.4155
anc	13%	6%	1.0	0.7	0.4157
anc	14%	6%	1.0	0.8	0.4146
anc	14%	7%	1.0	1.0	0.4120
lnc	14%	4%	1.0	0.5	0.4275
lnc	350	50	1.0	0.5	0.4275
lnc	13%	4%	1.0	0.5	0.4278
lnc	12%	4%	1.0	0.5	0.4279
lnc	11%	4%	1.0	0.5	0.4283
lnc	14%	7%	1.0	0.7	0.4259
lnc	14%	6%	1.0	0.7	0.4268
lnc	16%	3%	1.0	0.7	0.4274
lnc	15%	4%	1.0	0.7	0.4274
lnc	13%	6%	1.0	0.7	0.4274
lnc	350	50	1.0	0.7	0.4277
lnc	15%	3%	1.0	0.7	0.4278
lnc	14%	5%	1.0	0.7	0.4278
lnc	300	50	1.0	0.7	0.4280
lnc	14%	4%	1.0	0.7	0.4282
lnc	14%	3%	1.0	0.7	0.4284
lnc	13%	4%	1.0	0.7	0.4284
lnc	12%	4%	1.0	0.7	0.4291

Table 1: Results of learning runs for routing (learning with D1, testing with D2)

4 Results

4.1 Routing results

The results of the official runs show that our routing run is clearly above the average of all runs. Thus, we can conclude that our approach — although it is very simple — works and yields good results.

4.2 Ad-hoc results

Both ad-hoc runs produced results of average quality. Considering the constraints underlying our approach, and that no tuning or learning was performed, this is a positive result. Comparing the two approaches, there seems to be no significant difference. Thus, we can conclude that it does not matter whether or not the ranking performed in the agent can use global or local term frequency information only. So the overhead in transmitting this frequency information to the agent can be saved. It seems that the retrieval quality is affected mainly by the indexing functions used in the servers.

5 Conclusions and Outlook

The results of our experiments show that retrieval in distributed environments can be performed without loosing too much in terms of retrieval quality.

Here we only have considered very simple weighting schemes. By using more sophisticated schemes, and especially by applying appropriate learning methods, much better results can be expected.

With the increase of network connectivity and the growing number of document bases accessible, the development of retrieval methods for even large numbers of loosely coupled IR systems is becoming a major research issue.

References

Kahle, B.; Morris, H.; Goldman, J.; Erickson, T.; Curran, J. (1993). Interfaces for Distributed Systems of Information Servers. *Journal of the American Society for Information Science 44(8)*, pages 453–467.

Pfeifer, U.; Fuhr, N.; Huynh, T.T. (1995). Searching Structured Documents with the Enhanced Retrieval Functionality of freeWAIS-sf and SFgate. To appear in: *Proc. of the 3rd World Wide Web*

Conference '95. Also available from http://ls6-www.informatik.uni-dortmund.de/~pfeifer/fwsf.html.

Salton, G.; Buckley, C. (1988). Term Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management 24(5)*, pages 513–523.

Salton, G.; Buckley, C. (1990). Improving Retrieval Performance by Relevance Feedback. *Journal of the American Society for Information Science 41(4)*, pages 288–297.

New Tools and Old Habits: The Interactive Searching Behavior of Expert Online Searchers using INQUERY

Jürgen Koenemann* Richard Quatrain Colleen Cool

Nicholas J. Belkin†
School of Communication, Information, and Library Studies
Rutgers University
New Brunswick, New Jersey 08903
{koeneman,ccool,belkin}@cs.rutgers.edu

Abstract

We present data that describe the interactive searching behavior of ten searchers using the INQUERY retrieval engine in the context of the TREC-3 routing task. We discuss how these searchers with a strong background in the use of traditional online retrieval mechanisms adapted, after very limited training, to the use of a best-match, ranked-output, full-text retrieval mechanism.

1 Introduction

The development of IR system technology from slow, batch-mode, query submission to online query submission with rapid response generation by the IR system now allows for a highly interactive, iterative query development process. From experimental studies, it has been known for some time that best-match, ranked-output retrieval techniques are in general superior to exact-match systems in terms of recall and precision performance measures (e.g. [Belkin & Croft1987]). Furthermore, it seems that systems which allow queries to be put in unstructured form allow easier query formulation than those which require boolean structure, and are, at least for end-users, more effective [Frei & Qiu1993, Turtle1994]. Finally, it is quite clear that automatic relevance feedback, using term weighting and query expansion, either singly or in combination, significantly improves retrieval performance [Salton & Buckley1990].

The first users of these new retrieval mechanisms are likely to be those that have had access to and experience with traditional, boolean, set-based systems. We are interested in how these users will adapt to the new retrieval mechanisms. On the one hand, these users, typically professional online searchers, will have acquired an extensive repertoire of searching strategies. For example, searchers may be able to transfer their knowledge about the effects of adding terms to boolean queries in the form of conjunctions or disjunctions to a situation where they add terms within or outside the scope of proximity operators. On the other hand, these searchers are novices with respect to the retrieval mechanism and some of its interactive features. Furthermore, whereas some experiences with information retrieval mechanisms may be helpful because they can be directly transferred to the new situation, others may be not applicable or not effective in the new situation. We were interested to see the degree to which searchers attempted to transfer their existing search strategies such as a component-approach or use of boolean operators to the new environment and to what degree they would experiment with new tools such as relevance feedback made available to them.

We used the TREC-3 routing task context as a testbed to investigate these questions. We provided

*Department of Psychology
†to whom all correspondance should be addressed

ten experienced online searchers with a training manual that guided them through a sequence of hands-on exercises that familiarized them with the various features of the interface. After searchers completed the first part of the manual they performed one routing task (that is not included in our analysis). During the training phase, we provided some additional help when serious misconceptions persisted or when searchers experienced breakdowns from which they could not recover on their own. Each searcher then performed interactive searches for five TREC-3 topics (each lasting up to 20 minutes) without further help.

This paper focuses on the analysis of these expert online searchers with no prior INQUERY searching experience (the results reported as rutir1 in this volume). The comparative analysis of their performance and behavior with the experimenters' performance on the same task (reported as rutir2 in this volume) will be the focus of a forthcoming report.

The remainder of the paper is organized as follows. We first provide details on the searchers, experimental materials, the interface and retrieval mechanism, and the experimental procedure. Next we discuss summary performance and interaction data that demonstrate common trends among searchers. We continue with a discussion of some particular interactive searches that highlight some of the successes and failures users experienced. We conclude with some observations on the ways in which our searchers used the interactive facilities which were made available to them, and with some discussion of problems of evaluation of highly interactive IR systems.

2 A Case Study

2.1 Searchers

We recruited 10 professional searchers to participate in our study. The searchers had between 1 and 15 years experience with online search with traditional, boolean, set-based information retrieval mechanisms ($\overline{X} = 6.1$ years) and their current position required the large majority to perform online searches at least a couple of times each week. All searchers had a Master's degree in Library Science and used computers and graphical user interfaces on a regular basis. Searchers had at best some experience with full-text search but almost no experience with ranked-output retrieval mechanisms (8 out 10 had no experience at all). Two out of ten searchers had previously seen a cursory, ten-minute demonstration of the INQUERY system with an interface similar to the one used in this study in the context of a classroom demonstration. None of the searchers had any prior hands-on experience with the INQUERY retrieval engine nor with the specific interface employed. Searchers volunteered their time and were not compensated for their efforts.

2.2 Materials

The Retrieval Mechanism We used the INQUERY retrieval engine (version 1.6.3), developed at the University of Massachusetts [Callan et al.1992]. The underlying mechanism of INQUERY is a Bayesian probabilistic inference network [Pearl1988] which provides strict rules for the computation of probabilistic belief values for each document in which the degree of belief that a document is relevant is based on the terms that are shared by the query and the document and the operators in the query formulation used to combine these terms. For a detailed description of the INQUERY system see [Callan et al.1992]. The standard stop-word and stemming algorithms were used. We restricted the query language to the use of the two proximity operators #uw and #n and the synonym operator #syn, i.e. none of the soft or strict boolean operators of INQUERY were available to the searchers. See the System Description and Timing of the system for a detailed description.

The Interface The interface to INQUERY was a modified version of the INQUERY TK user interface developed at the University of Massachusetts [Callan et al.1992] implemented in Tcl/Tk [Ousterhout1994] a widget-style interface development language for the X windowing system running under the UNIX operating system. INQUERY and the interface ran on a remote SUN workstation; we

used a networked SPARC 4 with greyscale monitor, mouse, and keyboard for the searcher's interaction. See the appendix for a detailed description of the interface and all its features.

2.3 Procedure

The 50 TREC-3 routing topics (101–150) were categorized by hardness (median R-precision from TREC-2 results) into five groups. We created ten topic sets consisting of five topics each with about equal overall hardness by randomly selecting one topic from each hardness pool for each of the ten sets. Given these ten topic sets, we randomly assigned each searcher to one of the ten sets. The final queries generated by our ten searchers together constitute our **rutir1** submission for TREC-3.

Searchers were given a short general description of the TREC project and were given a consent form that outlined the experiment and the data collection. Upon signing of the form they were led to a screened-off area that was set up with the computer equipment and a video recording device. The experimenter gave a 2-minute introduction to INQUERY by listing its major features, namely ranked output (rather than set based retrieval), full text search (rather than title, abstract, or keyword search), the availability of automatic relevance feedback functionality, and the ability to process free form, unstructured queries. Searchers were next given a 14 page, ring-bound tutorial, used for a hands-on training session of about 90 minutes. The tutorial concluded with an example of their searching task for the experiment (see appendix for a detailed description of the tutorial and the searching task).

After a short break each of the ten searchers performed a series of five searches. The topics were determined by the random selection procedure described above and within one topic set topics were sorted by increasing hardness. That is, participants searched easier topics first and topics became harder with each search. Even though this fixed order for all searchers conflated hardness with learning, we nevertheless felt that searchers had to perform easier routing tasks first to enable at least some initial retrieval successes that motivated participants to continue with the experiment.

At the beginning of each search participants were handed the TREC topic description and a description of the routing task (see appendix). They were free to start interaction with the system at any time. During searches participants were allowed to refer back to the tutorial if they desired to do so but experimenters provided no further help during the trials except for restarting the system if a system failure had occurred. A search was completed when the searcher was satisfied with the result and saved a query as "final". If searchers had not completed their search after about 16 minutes they were reminded that each search was restricted to 20 minutes. After twenty minutes searchers were asked to complete the current iteration and to save what they felt was their best query as the final query.

Each search was followed by a short break during which the searcher filled out a short questionnaire regarding the particular search that had been completed (their familiarity with the topic, the ease of query construction, their confidence in the quality of the final query, and their rating of the severity of the 20 minute time constraint for this search). Searchers did not receive any feedback from the experimenter regarding the quality of the queries or regarding the strategies they employed except for the general encouragement that they were doing fine (independent of their actual performance).

After the last search and search questionnaire searchers filled out a background questionnaire and were debriefed about the goals of the study.

3 Results and Discussion

This section contains the major results from our study. We provide a quantitative and qualitative analysis of the data and present some explanations for the observed behavior. We first discuss the performance of searchers with respect to the TREC-3 routing task. The next section captures the structure and content of all queries issued throughout the interaction. The third section focuses on the interactive process itself and describes how searchers interacted with the various features of INQUERY and the in-

terface and how they iteratively developed their final queries.

3.1 Performance

This section describes the performance of our searchers using the standard performance measures of precision at 100 retrieved documents and R-precision. We also looked at other performance measures like precision at 20 documents, precision at 1000 documents, and average 11-point precision. All measures are highly correlated and using any of them would not change our findings. We first present data on the performance of searchers on the **training** collection, followed by data on the performance on the TREC-3 **test** collection. We compare these performances to each other and compare the test results to the best, median, and worst results of TREC-3.

Training Collection Searchers performed their 5 searches each on the TREC disks 1&2 (the TREC-2 "ad hoc" database). During the interaction, searchers were given the relevance rating for each displayed document title (obtained from TREC-2). Although searchers were instructed to perform a routing task, most searchers focused on developing a good adhoc query for the training collection under the (implicit) assumption that good performance of the query on the training collection was not a sufficient but a necessary condition for a successful routing query. Thus, it makes sense to analyze the performance of searchers on the training collection.

The mean R-precision was just under 30% ($\overline{X} = 0.2905, s_D = .1661$). The overall precision at 100 documents retrieved was $\overline{X} = .3828$ with a standard deviation of $s_D = .2719$. The single worst performance was on topic 101 for which the final query retrieved no documents at all because of an error in the query formulation and topic 124 for which the R-precision was .0114 (prec.@100=.02). The best R-precision results were obtained for topics 134 (R-prec.=.6596, prec.@100=.8500) and 148 (R-prec.=.6443, prec.@100=.7900), the best precision at 100 retrieved documents was obtained for topics 130 (R-prec.=.6216, prec.@100=.90) and 135 (R-prec.=.5478, prec.@100=.93).

Precision on Training Collection				
	avg. R-prec		avg. prec. @100	
n=5	$\overline{X}$	s_D	$\overline{X}$	s_D
S01	0.3328	0.1582	0.4520	0.2899
S02	0.1664	0.1573	0.2360	0.2454
S03	0.1365	0.1159	0.2180	0.1934
S04	0.3070	0.1137	0.3800	0.2411
S05	0.3815	0.2702	0.4500	0.3522
S06	0.2396	0.1532	0.3540	0.2753
S07	0.3393	0.1485	0.4600	0.2941
S08	0.3848	0.1560	0.4720	0.2958
S09	0.3067	0.1354	0.3260	0.2280
S10	0.3100	0.1434	0.4800	0.3648
All	0.2905	0.1661	0.3828	0.2719

Table 1: Searcher differences for precision of final queries on training collection. Given are the average R-precision and average precision at 100 documents for each searcher.

There was considerable variation among searchers and topics. The average R-precision for searchers (see table 1) ranged from .1365 for searcher S03 to .3848 for searcher S08. R-precision correlated highly with precision at 100 retrieved documents, $r = .8926, df = 48, p < .001$.

There was a moderate but not significant negative correlation between years of online search experience and average R-precision, $r = -.44, df = 8, n.s.$, that is, searchers with more experience in traditional online searching tended to perform worse.

As described, topics were grouped by hardness (based on the median TREC-2 performance) and searchers searched for topics in increasing order of hardness. There was a significant trial effect for R-precision, $F(4, 36) = 15.52, p < .001$, as well as for precision at 100 documents retrieved $F(4, 36) = 32.47, p < .001$. Performance decreased with increasing hardness (see table 2).

This trend is difficult to interpret since the order of topic hardness was not systematically varied. Thus, possibly learning effects and other factors such as fatigue may have contributed to the result. However, learning seems not to be occurring since it should re-

Precision on Training Collection				
	avg. R-prec		avg. prec. @100	
n=10 Trial	$\overline{X}$	s_D	$\overline{X}$	s_D
1	0.4293	0.1601	0.7120	0.2044
2	0.4105	0.1232	0.5880	0.1561
3	0.2758	0.1519	0.3250	0.1613
4	0.1966	0.0711	0.1730	0.0672
5	0.1401	0.0919	0.1160	0.0829
All	0.2915	0.1661	0.3828	0.2719

Table 2: Trial differences for precision of final queries on training collection. Given are the average R-precision and average precision at 100 documents for each trial (Hardness TREC-2 category)

sult in better not in worse performance. One may argue that without learning results would have been even worse for harder topics. The data for other TREC-3 participants (see below) and our own rutir2 results suggest that indeed hardness is the major factor since there was no inter-trial learning involved in these systems and orders presumably varied from the one we employed; nevertheless the same pattern of decreasing performance can be found.

Test Collection The performance of final queries on the TREC-3 test collection (TREC disk 3) is the official performance evaluation criterion for TREC-3. It evaluates whether the queries developed by searchers in the context of the training collection were indeed suitable routing queries that produced relevant documents when applied to a different albeit similarly structured collection. In summary, the results from the training collection were replicated for test collection with somewhat lower overall results as one would expect.

The mean R-precision was $\overline{X} = 0.2549$ with a standard deviation of $s_D = 0.1880$. The mean precision at 100 documents retrieved was $\overline{X} = .2898$ with a standard deviation of $s_D = .2597$. Again, the single worst performance was on topic 101 for which the final query retrieved no documents at all because of a query error. Topics 104, 105, 106, and 116 also retrieved zero relevant documents. Similar results were

Precision on Test Collection				
	avg. R-prec		avg. prec. @100	
n=5	$\overline{X}$	s_D	$\overline{X}$	s_D
S01	0.3052	0.1812	0.3340	0.3217
S02	0.1304	0.0979	0.1160	0.1124
S03	0.1211	0.1560	0.1740	0.2825
S04	0.1549	0.1525	0.1860	0.2066
S05	0.4007	0.2609	0.3340	0.2723
S06	0.1723	0.1802	0.2300	0.2230
S07	0.3213	0.1835	0.4080	0.2491
S08	0.3825	0.1827	0.4420	0.3072
S09	0.2904	0.1876	0.3120	0.2419
S10	0.2699	0.1410	0.3620	0.3430
All	0.2549	0.1880	0.2898	0.2597

Table 3: Subject differences for precision of final queries on the test collection (TREC disk 3). Given are the average R-precision and average precision at 100 documents for each searcher

obtained using precision at 100 retrieved documents as a measure of performance: in addition to the above topics, queries performed very poorly on topics 102, 103, 116, 127, 131, 144, and 149 (all with prec.@100 <.04. With the exception of topics 127 and 149 all these topics were characterized by a very small number of relevant documents. In evaluating these low numbers one needs to take into account that the theoretical maximum performance on these topics was very low as well since 5 topics had under 10 relevant documents in the collection and 23 topics had under 100 relevant documents.

The best R-precision results were obtained for topics 148 (R-prec.=.6952, prec.@100=.7700) and topic 135 (R-prec.=.5961, prec.@100=.84), the best precision at 100 retrieved documents was obtained for topics 150 (R-prec.=.4119, prec.@100=.88) and topic 135.

There was considerable variation among searchers and topics. The average R-precision for searchers (see table 3) ranged from .1211 for searcher S03 to .4007 for searcher S05. R-precision correlated highly with precision at 100 retrieved documents, $r = .826, df = 48, p < .0001$.

Precision on Test Collection				
	avg. R-prec		avg. prec. @100	
n=10 Trial	$\overline{X}$	s_D	$\overline{X}$	s_D
1	0.4503	0.1784	0.6460	0.2504
2	0.3579	0.1211	0.3530	0.1870
3	0.2015	0.1855	0.2410	0.1727
4	0.1409	0.1124	0.0940	0.0885
5	0.1238	0.0895	0.1150	0.0848
All	0.2549	0.1880	0.2898	0.2597

Table 4: Trial differences for precision of final queries on test collection (TREC disk 3). Given are the average R-precision and average precision at 100 documents for each trial (Hardness TREC-2 category)

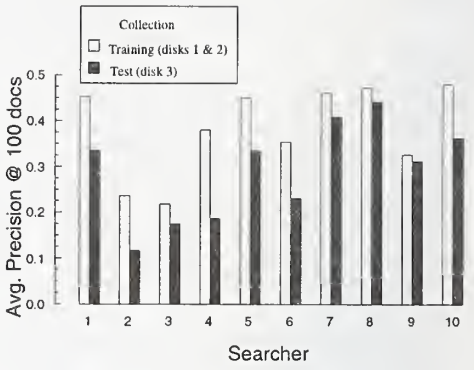


Figure 1: Searcher Averages

There was a significant effect of trial (hardness based on Trec-2 median R-Precision) on R-precision, $F(4,36) = 17.008, p < .001$, as well as for average precision at 100 retrieved documents, $F(4,36) = 23.975, p < .001$.

Differences between Training and Test Collection The routing task required searchers to develop a routing query in the retrieval context of a training collection (TREC disks 1& 2) but for an unknown test collection (TREC disk 3). It is, thus, interesting to compare the performance on the test collection to the performance on the training collection. The average R-precision dropped from .2905 to .2549 (a difference of .0356, $s_D = .0996$). This drop was significant, $F(1,49) = 6.379, p < .05$. The same holds for average precision at 100 retrieved documents: performance dropped significantly from .3828 to .2898 (diff= .093, $s_D = .171$), $F(1,49) = 14.75, p < .001$.

The correlation between precision at 100 retrieved documents for the test and the training collection was significant, $r = .794, p < .001$. The correlation for the R-precision measure was $r = .849, p < .001$.

The by far worst drop for R-precision (-.468) occurred for topic 104 (to 0.0), a topic with only 3 relevant documents in the test collection, whereas the best case was an improvement in R-precision by 0.1457 for topic 110 (from .4442 to .5899) probably due to the large number of relevant documents in the test collection (534). Overall, the performance dropped for 32 topics, remained the same for 1 topic, and increased for 17 topics. For 40 out of the 50 topics the difference was less than .1.

The worst precision drop at 100 retrieved documents (-.71) was for topic 134 (from .85 to .14) due to a small number (19) of relevant documents in the test collection, whereas the best case was an improvement in precision at 100 retrieved documents by 0.3 for topic 142 (from .37 to .67) probably due to the large number of relevant documents in the test collection (638). Overall, the performance dropped for 33 topics, remained the same for 1 topic, and increased for 16 topics. For 21 out of the 50 topics the difference was less than .1.

Figure 1 shows the average precision at 100 retrieved documents for each of the ten searchers on the training as well as the test collection. One searcher's queries (S09) did only minimally worse (.014 precision difference) on the test collection, and three searchers (S03, s07, and S08) had on average only slightly less good performance ($\leq$ .05 precision difference) on the test collection. The remaining 5 searchers had larger drops in performance ($>$ .1 precision difference). The worst drop occurred for searcher S04 whose queries showed an average drop in precision of .194.

We also looked at differences between the train-

ing and the test collection by trial (hardness category). Average precision at 100 retrieved documents dropped for all trials (hardness categories) with essentially no drop for the hardest topics (-.001) and the biggest drop occurring in trial 2 (-0.235). These differences were barely significant, $F(4, 36) = 2.667, p = .048 < .05$. However differences by trial were not significant for the R-precision measure, $F(4, 36) = 1.575, n.s.$ This is not surprising since the hardness categories were formed based on the TREC-2 performance on the training collection TREC disks 1&2 and thus are less applicable to the test collection TREC disk 3.

Comparing Rutir1 to General TREC-3 Performance The variation in indexing and retrieval mechanisms makes it difficult interpret the differences between our human searchers and the general performance of all systems that participated in TREC-3. Nevertheless, such a comparison allows us to judge the performance of our information system, comprised of INQUERY, the XINQUERY interface, and the human searcher as a unit and to detect general similarities and dissimilarities. Since R-precision data were not available at this point for all systems, we restrict our discussion to a comparison of precision as measured by the best, median, and worst precision at 100 retrieved documents for each topic. One should note as a proviso that the comparison against the best and the worst performance is somewhat problematic if one averages over a (sub)set of topics since one would not expect a single system to perform as well (or as badly) as the level which one obtains by combining the best (worst) performances of a large group of systems. A fairer comparison is probably the comparison with the median performance of all TREC-3 systems.

The average precision at 100 retrieved documents for our human searchers was .2898. The combination of best systems reached an average precision of .5342, a significant difference of .2444, $t(48) = 9.31, p < .001$. The median TREC-3 average precision was .4190, a significant difference of .1292, $t(48) = 12.22, p < .001$, and the combination of worst system performances yielded an average precision of 0.1782, a significant difference of .1116 in favor of our searchers, $t(48) = 10.25, P < .001$.

We correlated the performance of the human searchers (rutir1) with the best, median, and worst results for all TREC-3 systems. A high positive correlation would suggest that the human searcher would perform similarly on each topic (albeit on a different level) whereas a negative correlation would suggest that, e.g. topics that were most difficult for machine-based systems were much easier for human searchers and vice versa. The correlation of precision at 100 retrieved documents between rutir1 and the best TREC-3 results was $r = .802$ ($n = 50$ topics), the correlation of rutir1 precision at 100 retrieved documents with the median TREC-3 result was $r = 0.870$, and the correlation of rutir1 precision at 100 retrieved documents with the worst precision level in TREC-3 was $r = .829$; all significant at $p < .001$. This suggests that the difficulty of topics relative to each other was similar for human searchers and the machine approaches. One should note that the independence assumption for the above statistical test is violated, but given the size of the effects these violations will not change the results.

Given that relative searcher performance was influenced by topic difficulty based on the training collection, we may want to confirm our results for the test collection by comparing searchers not just among themselves (see above), but also against the general TREC-3 performance on the matching topic sets. Table 5 indicates the average difference in precision at 100 documents between the human searchers and the best, median, and worst TREC-3 results.

The searchers are sorted in order of decreasing average precision. Searcher performance is correlated with the best and median TREC-3 performance on the same topic set. Compared to the median performance, searcher S08 performed best (an average of -0.022 under the median), whereas searcher S02 performed worst (an average of -.234 under the median TREC-3 precision). We will return to these differences when we discuss the search processes and query characteristics associated with these searchers in more detail.

We discussed earlier that trial effects could be attributed either to learning or to the hardness of the

Comparison to TREC-3 Precision @100 on Test Collection				
n=5	$\overline{X}_{rutir1}$	$\overline{X}_{Best}$	$\overline{X}_{Median}$	$\overline{X}_{Worst}$
s08	0.4420	0.5840	0.4640	0.2660
s07	0.4080	0.6720	0.5380	0.2140
s10	0.3620	0.6260	0.4760	0.1800
s01	0.3340	0.6060	0.4420	0.1000
s05	0.3340	0.6180	0.4980	0.2480
s09	0.3120	0.5680	0.4400	0.2280
s06	0.2300	0.4360	0.3600	0.1800
s04	0.1860	0.3920	0.3040	0.1420
s03	0.1740	0.3780	0.3180	0.1600
s02	0.1160	0.4620	0.3500	0.0640
All	0.2898	0.5342	0.4190	0.1782

Table 5: Subject differences for precision of final queries on test collection. Given are the average precision at 100 documents for each searcher and the average performance for the same topics with the best, median, and worst TREC-3 results

topic. It is, thus, interesting to compare the average precision of our human searchers for the various hardness categories to the best, median, and worst precision values for the same topic sets achieved by the TREC-3 systems as a group. Figure 2 shows the average precision at 100 retrieved documents for the human searchers on the test and on the training collection and compares it to the best, median, and worst TREC-3 results on the test collection. One can see that there is a steady linear decline of precision with increasing hardness for all systems. Thus, it seems appropriate to attribute the drop in performance among the human searchers to the hardness of the topic and not to fatigue, negative transfer, or other problems. A by-hardness analysis of differences between the various systems shows no significant effect, except for the somewhat larger drop in performance in hardness category 4 for the human searchers. There is no trend that the difference between human searchers and automatic retrieval systems is larger or smaller based on topic difficulty.

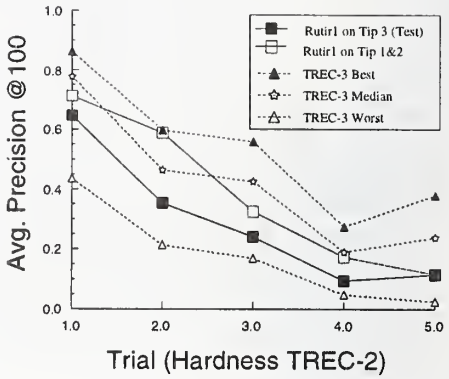


Figure 2: Performance Comparison

3.2 Query Characteristics

In our study we were interested in the nature of interactive query construction. This section describes the queries searchers entered throughout the interaction as well as the final queries saved as the routing queries that constituted our official TREC-3 submission. Sections 3.3.1 and 3.3.2 discuss characteristics of the interactions in which query construction took place.

Searchers entered a total of 338 queries, i.e. an average of $\overline{X} = 6.76$ queries per topic ($s_D = 4.00$). There were large individual differences: the average number of queries per topics ranged from 10 for searchers S02 and S04 to 3.8 queries per topic for searchers S06 and S10 but within searcher variation was also substantial.

There was a significant increase in the number of queries with increasing trials, $F(4, 36) = 4.67, p < .01$. Table 6 gives the averages for each trial (hardness category). Due to the confounding of trial and hardness alternative interpretations are possible and verbal protocols and interaction data suggest that both are at play: on the one hand, easier topics often led to satisfactory performance of early queries and searchers simply ended the interaction without having used up the allotted time. Secondly, searchers became more confident in the use of system features

Average Number of Queries		
n=10 Trial	$\overline{X}$	s_D
1	4.40	3.1340
2	5.90	2.9981
3	6.30	4.4485
4	7.20	3.5528
5	10.00	4.0825
all	6.7600	3.9978

Table 6: Trial (Hardness Trec-2) differences for average number of queries.

Query Characteristics				
Querytype	First	Interm.	Last	All
no rf				
no operators	11	30	0	41
w/ operators	39	152	22	213
Sum no rf	50	182	22	254
with rf				
no operators	n/a	10	4	14
w/ operators	n/a	46	24	70
Sum rf	0	56	28	84
Sum	50	238	50	338

Table 7: Query types for first, intermediate, and last (final) queries. Relevance feedback (rf) not possible in first query.

and were able to execute more queries in the constrained time span allowed for each topic.

We can characterize the queries by three factors: the use of terms in the query, the use of operators (synonym operator and proximity operators), and the use of relevance feedback to expand the user's formulated query and to reweight the terms. Searchers had been trained on the use of all three query formulation mechanisms. Users entered terms for all queries. Table 7 provides a breakdown of the use of operators and relevance feedback in the first, last, and intermediate queries.

The average number of terms in a query that was entered by the searcher was $\overline{X} = 7.19$. The majority of these words was taken from the topic descriptions. The minimum number of query terms was one and

Avg. Number of Query Terms			
n=5	$\overline{X}$ All	$\overline{X}$ First	$\overline{X}$ Final
S01	6.20	3.2	10.2
S02	5.66	3.4	7.0
S03	9.06	5.2	13.2
S04	6.36	3.8	7.2
S05	4.26	2.8	5.2
S06	6.00	4.6	6.2
S07	4.93	3.2	6.6
S08	16.15	11.2	17.0
S09	8.00	8.4	8.6
S10	10.47	6.4	11.4
All	7.19	5.22	9.26

Table 8: Subject differences for the number of terms entered per query. Given are the average number of user terms entered for all queries, the first query, and the last query.

the maximum was 36 terms. For 42 of the searches users never entered more than 12 terms.

Table 8 shows the average number of user terms entered for all 338 queries, for the first query for each topic, and for the final query for each topic. The 50 first queries contained an average of $\overline{X} = 5.22$ terms whereas the the 50 final queries had an average of $\overline{X} = 9.26$ words entered by the user. One should note that the number of processed terms in the final queries is even higher in cases (28) where relevance feedback led to query expansion. The average number of processed terms in the final queries was $\overline{X} = 23.82$ terms.

The average number of user terms per query also varied with trial (topic hardness). The largest number of terms was found for trial 1 ($\overline{X} = 9.75$) with a steady decrease to trial 4 ($\overline{X} = 5.64$). Trial 5 had an average of $\overline{X} = 7.46$ user terms per query. Further analysis needs to be done to find an explanation for these data. It may be the case that an increased reliance on relevance feedback combined with increasing fatigue led to the decline in user-supplied terms. There was a weak but significant positive correlation between the number of user words and the precision at 100 retrieved documents, $r = .2855, p < .05$, that

is, more user words led to better performance. However, the correlation between the actual number of processed terms used for retrieval (i.e. after relevance feedback led to query expansion) was not significant, $r = .209, p > .1$.

Searchers were introduced during the training process to the use of three non-boolean operators: fixed-order proximity (#n), unordered proximity (#uw), and synonym (#syn). The average number of operators per query was $\overline{X} = 2.54$. The average number of operators used in final queries was $\overline{X} = 3.34$, and the average number of operators in first queries was $\overline{X} = 1.94$. Searchers made almost equal use of all three operators (27.9% #n, 35.5% #uw, 36.6% #syn). There was no significant correlation between operator use in the final query and performance. The structure of queries was relatively simple; the vast majority of queries had at most one operator nested inside of another operator (typically a synonym operator inside of a proximity operator).

Table 9 details the use of operators by each searcher. Given are the average number of operators used in each query. Most searchers used on average between one and three operators in each query. Only searchers S03 and S08 had significantly higher use of operators. Searcher S08, the best searcher in our group, used four times as many unordered proximity operators compared to the other searchers and had a higher frequency of synonym use as well. Searcher S03 used many more phrase (fixed proximity) operators than others, and slightly more synonym operators as well.

Table 10 provides a breakdown of operator use by trial (hardness). Searchers used the most operators during their first search, in particular, #n operators. The differences between trials were not significant due to the variation within groups.

Finally, we looked more closely at the use of relevance feedback. We noted already above (see table 7) that over half of the final queries used relevance feedback and that a total of 84 queries were relevance feedback queries. Tables 11 and 12 show the use of relevance feedback on a by-trial (hardness) and by-searcher basis.

There was no systematic increase or decrease in relevance feedback usages over trials. Thus, neither

Operator Use in Query Construction				
n=5	$\overline{X}$all	$\overline{X}$ #n	$\overline{X}$#uw	$\overline{X}$#syn
s01	2.22	0.68	0.93	0.61
s02	1.28	0.02	0.88	0.38
s03	4.89	2.40	0.86	1.63
s04	1.18	0.38	0.00	0.80
s05	1.71	0.90	0.65	0.16
s06	2.53	1.37	0.58	0.58
s07	2.04	0.07	0.98	1.00
s08	6.81	0.27	4.19	2.35
s09	2.24	0.57	0.29	1.38
s10	2.95	1.68	0.11	1.16
All	2.54	0.71	0.90	0.93

Table 9: Searcher differences for the use of operators. Given are the average number of operators per query. The columns contain the averages for the number of all operators used, the use of the proximity operators #n and #uw, and for the use of the synonym operator

Operator Use in Query Construction				
n=10 Trial	$\overline{X}$all	$\overline{X}$ #n	$\overline{X}$#uw	$\overline{X}$#syn
1	3.43	1.39	1.07	0.98
2	3.14	0.51	1.39	1.24
3	1.86	0.35	0.81	0.70
4	2.63	0.81	0.79	1.03
5	2.18	0.69	0.68	0.81
All	2.54	0.71	0.90	0.93

Table 10: Trial (hardness) differences for the use of operators. Given are the average number of operators per query. The columns contain the averages for the number of all operators used and averages for each operator type

Use of Relevance Feedback			
n=10 Trial	rf possible	rf used	%
1	34	7	0.21
2	49	8	0.16
3	53	16	0.30
4	62	27	0.44
5	90	26	0.29
all	288	84	0.29

Table 11: Trial (Hardness) differences for the use of relevance feedback. Given are the number of queries in which rf use was possible (non-first queries), the number of queries in which rf was used, and the percentage of use for each trial.

Use of Relevance Feedback			
n=5	rf possible	rf used	%
S01	36	12	0.33
s02	45	0	0.00
s03	29	9	0.31
s04	45	25	0.56
s05	26	6	0.23
s06	14	6	0.43
s07	41	0	0.00
s08	21	8	0.38
s09	19	8	0.42
s10	14	10	0.71
all	288	84	0.29

Table 12: Subject differences for the use of relevance feedback. Given are the number of queries in which rf use was possible (non-first queries), the number of queries in which rf was used, and the percentage of use for each subject.

experience with the system nor the increased hardness of topics led to a significant increase in the use of relevance feedback. Searcher differences were pronounced: two searchers (S02, S07) did not use relevance feedback for any of their searches, whereas searchers S04 and S10 used relevance feedback for 56% and 71% respectively of the queries where it was possible to do so. A detailed analysis of the protocols will be needed to determine situations that trigger the use of relevance feedback by searchers. There was no correlation between the use of relevance feedback and performance.

3.3 The Interaction Process

The searchers in this project were presented with the task of developing a single best query for each of the topics given to them. Therefore, our discussion of interactive searching behavior is in fact a discussion of query formulation procedures. An interesting question which we address in this study concerns the ways in which experienced online searchers interact with a new retrieval system, INQUERY. The searchers in our study were experienced users of interactive systems, and they came to this experiment with a stock of routine query formulation procedures. However, some of these procedures, such as boolean set construction, do not easily map onto INQUERY. At the same time, INQUERY offered several interactive features, such as automatic relevance feedback, which were completely new to the searchers. The searchers in our study were faced with conceptual problems of understanding the topic itself, and in addition, the conceptual and procedural problems of learning to use the INQUERY system. An interesting research question concerns the extent to which searchers in this situation try to make the new system work to support their typical searching behaviors; or whether the searchers develop new searching behaviors, and new query formulation procedures as they interact with the new retrieval system.

3.3.1 Characteristics of the Interaction

We begin our discussion with some general characterizations of the interactive query formulation process,

relating them, where relevant, to performance measures. The extent to which query reformulation actually took place in our searches, and the means and character of the changes, can be indicated by several parameters, including:

- changes between the initial queries and the final queries;
- number of iterations in a search;
- use of relevance feedback;
- use of features supporting manual reformulation;
- kinds of interaction with displayed output

Changes between initial and final queries can be measured relatively easily in terms of differences in numbers of query features. Tables 7 and 8 and the related discussion in section 3.2 show that there were substantial changes in these characteristics, with mean number of searcher-supplied words increasing from 5.22 to 9.26 (to 23.82 when terms supplied by relevance feedback are considered), and number of operators increasing from 1.94 to 3.34.

The number of iterations per search is a strong indicator of extent of interaction. Our definition of iteration, "that part of a search bounded by uses of the "Run Query" button" (first iteration beginning with starting the search, last ending with saving the final query), is equivalent to the concept of "cycle" as used in discussing interaction in traditional boolean retrieval systems (cf. [Harter1986]). The mean number of iterations per search, for all searchers, was 7.760. There was no significant difference in number of iterations by searchers, but the number of iterations increased significantly by trial (that is, by hardness), $F(4, 36) = 4.607, p < .01$ (see table 13).

There was a medium but significant negative correlation between the number of iterations and precision at 100 retrieved documents, $r = -.435, p < .05$ and a medium correlation between the number of iterations and trial (hardness), $r = .438, p < .01$. One possible interpretation is that topic difficulty led to increased number of iterations but this increased effort did not pay off. To account for training and hardness, we computed the correlation between the number of iterations and the precision at 100 documents retrieved separately for each trial, i.e. for topics with similar difficulty. None of the correlations was significant (they ranged from .001 to -.498, but the trend was clearly in the same direction as the overall result: the more iterations searchers performed, the poorer was the result. One possible partial explanation is the fact that the submission of a syntactically wrong query ended one iteration. Thus, searchers who made syntax errors and maybe had more difficulties in general had more iterations.

Average Number of Iterations		
n=10 Trial	$\overline{X}$	s_D
1	5.50	3.064
2	6.90	2.998
3	7.30	4.449
4	8.10	3.604
5	11.00	4.083
all	7.7600	3.982

Table 13: Trial (Hardness Trec-2) differences for average number of iteration, per topic.

The methods by which searchers reformulate their queries can be characterized, in our setting, as automatic (using relevance feedback) or manual (adding/deleting terms and/or operators). Automatic relevance feedback was used in 29.2% of all of the queries in which it was possible. Interestingly, it was used in 58.3% of the final queries (28 out of the 48 searches which actually had some reformulation). In fifteen searches relevance feedback was never used. Two searchers accounted for ten of those searches, the remaining five distributed amongst four searchers. Two of the latter instances were searches with no reformulation of any kind. See tables 11 and 12 in section 3.2 for detailed data on use of relevance feedback, and its effect on performance.

Manual query reformulation, that is, addition, deletion or replacement of terms or operators by the user directly, was not used in seven searches. Five of those searches used automatic relevance feedback, and for two there was no reformulation at all.

From our data, it is clear that manual and automatic query reformulation are not mutually exclusive. Two searches involved no reformulation at all, five searches involved only automatic query reformulation, and 13 searches used only manual query reformulation. Thus 28 of the 50 searches used both automatic and manual query reformulation techniques.

The use of relevance feedback in our interface required the user to turn relevance feedback on, and to select at least one document as relevant (negative relevance feedback is not supported in INQUERY). Over all 50 searches, relevance feedback was turned on 54 times, and turned off 19 times (relevance feedback remains on in our interface to INQUERY, unless explicitly turned off, once it is invoked). The mean number of documents selected per iteration, over all searches in which relevance feedback was used, was 3.969, and the mean sum of documents marked as relevant, per search, was 16.120, with a range of 0 - 116. Over all searches, there were only 25 instances of deselection of documents.

Support for manual query reformulation was minor in our interface. The major such tool was the ability to save query statements, and subsequently combine them. We can get some indication of the general strategy of successive formulation of parts of a query, with subsequent combination of those parts (the most common query formulation pattern in traditional IR systems) through analysis of the use of the save and load query facilities. The mean use of the "save query" tool was 2.140, but since all searchers were required to use this feature at least once, to save the final query, the use of this tool for purposes other than saving the final query was 1.140. The total number of such uses was 57. The "load query" is perhaps the most direct indicator of this type of query reformulation strategy. 41 searches did not use this tool at all, and there were only 36 total occurrences, of which one searcher accounted for 22.

Finally, we can ask where the searchers found the evidence on which to base their query reformulation strategies, whether manual or automatic. In the automatic relevance feedback strategy, the most obvious source of evidence was the relevance judgments that were displayed with each document title. But for many retrieved documents, there were no relevance judgments, so the presumption is that decisions would be made on the basis of the other information in the summary window (belief value, document id, including the acronym indicating the database, and title), or on the basis of the texts of documents. For support of manual query reformulation, the highlighted terms in the texts of displayed documents could serve as an additional source for query reformulation. Accordingly, we consider here the number of titles, and the number of documents, that were viewed by the searchers.

In our interface, the default display as a result of a search was the summary window with five titles. In order to view a document, the user either had to double-click on the desired title, to double-click on the desired title as indicated in the bar graph (this never happened), to select a document through the scroll-bar next to the document display window, or to invoke the "next document" or "previous document" buttons below the document display window (this technique was effective only if a document was already displayed). The mean number of instances of viewing the full-text of a document, per search, was 6.620. Since our definition of iteration leads to one less chance for documents to be displayed than there are iterations in a search, there was an overall average of about one full-text display per iteration. That is, the total number of displayed documents, for all iterations, was 331, and the total number of iterations in which it was possible to see a document was 338.

There were clear differences between searchers in the display of the full-text of documents, as indicated in table 14. The searcher with the highest average (S09) looked at between 13 and 36 documents during a search, whereas the searcher with the lowest average never looked at more than one full text of a document during a single search. Absolute numbers for the other searchers ranged from 0 (searcher S10) to a maximum of 14 full texts viewed.

We identified three different conditions under which our searchers could view titles: the total number of titles displayed during a search or interaction; the number of unique titles displayed during an interaction; and, the number of unique titles displayed during an entire search. The first gives some indication of the overall extent of interaction in a search;

Viewing of Full Text of Documents		
n=5	$\overline{X}$	s_D
S01	5.6	3.21
S02	0.6	0.55
S03	6.6	5.55
S04	8.0	2.45
S05	7.0	2.00
S06	6.4	4.34
S07	8.8	3.03
S08	2.0	1.73
S09	18.4	9.91
S10	2.8	4.38
All	6.62	6.22

Table 14: Searcher differences for the number of times a full text of a document was displayed during one search (topic) and presumable viewed by a searcher.

the second, in comparison to the first, is an indication of searchers' returning to earlier displays; the second and third show how many titles searchers had available to them as sources for query reformulation; and, the third is an indication of differences in outcome of queries. We make the assumption, for general analysis, that the display of a title is equivalent to the searcher having attended to (viewed or seen) that title, and the further simplifying assumption that the number of documents without titles is relatively small (or, alternatively, that the searchers gain at least some information from the other information associated with the summary display, even when there is no title).

Table 15 shows the means per search, by searcher, of total titles seen for each of these viewing conditions.

Overall means for each condition, respectively, are 136.34 titles seen per search, 110.08 iteration-unique titles seen per search, and 67.2 unique titles seen per search. The first column (Table 15) indicates that there was a fairly high degree of interaction overall. The second column suggests that there was not too much "thrashing", or repetitive viewing within an iteration. The mean percentage of duplicate titles seen in an iteration was 19.26%, but eight out of the ten searchers had percentages below this mean, with one accounting for an average of 95 duplicate titles per search. The third column indicates that the queries were in general reasonably effective in finding new documents. Although the overall percentage of unique documents viewed is about 49%, seven of the searchers performed at between 58% and 64% "effectiveness", with one, with a large number of viewed documents, finding only about 27% new titles. There are clear differences in all three characteristics by searcher (table 15), but no significant differences by trial (hardness). It is of some interest to note that of the 6817 titles that were displayed, to all searchers, in all iterations, only 331 (somewhat less than 5%) were selected to be viewed in full.

Overall, these data indicate that the searchers engaged in substantial interaction in the formulation of their queries, using a wide variety of tools and facilities. Judging the effectiveness of the various forms of interaction, in terms of performance, is not easy, however. The problem is that it is not clear how the initial user queries should be interpreted. If one views them as initial starting points for subsequent interaction, then it makes no sense to compare the effectiveness of the first query with that of the final query.

3.3.2 Interactive Query Formulation

In this section we look at some of the query formulation procedures used by our searchers. We describe three sample searches, in order to illustrate three different query formulation strategies; one in which the searcher makes minimal or no use of new system features; one in which the searcher attempts to learn how to use system features over the course of a search, and one in which the searcher uses new system features right from the start.

Topic 122 - Minimal use of new features We chose to describe the query formulation procedures used by S07 in topic 122 as an example of minimal use of new system features. Topic 122 was chosen because it is familiar to all of the interactive systems participating in TREC-3; that is, all interactive sys-

Viewing of Document Titles						
	All Instances of viewing		Unique in Iteration		Unique in Search	
n=5	$\overline{X}$	s_D	$\overline{X}$	s_D	$\overline{X}$	s_D
S01	70.2	17.0	55.0	13.9	42.2	15.3
S02	53.2	24.7	45.2	18.5	31.0	11.4
S03	82.0	55.8	69.6	45.2	37.4	23.5
S04	319.0	94.4	224.0	47.6	85.4	32.9
S05	120.0	30.2	98.0	24.9	69.2	24.2
S06	58.0	30.5	48.0	23.6	33.8	14.5
S07	59.0	27.0	50.0	26.9	36.2	15.8
S08	319.2	129.6	267.8	127.9	157.0	91.5
S09	78.8	42.6	68.8	33.1	49.8	22.3
S10	204.0	102.4	174.4	86.0	130.0	63.6
All	136.3	118.2	110.08	93.4	67.2	55.6

Table 15: Searcher differences for the number of titles displayed in the summary window throughout the course of a single search (topic) and presumably seen by the searcher.

tems have been asked to provide a detailed description of the entire search process for this topic.

S07 began t122 with the following query formulation:

```
#uw50(cancer #syn(drug chemotherapy
treatment))
```

After 8 iterations, the final query was the following:

```
#uw100 (cancer #syn( research testing
evaluation))
#syn(cancer leukemia)
#uw50 (cancer #syn (drug chemotherapy
treatment))
```

In this example, the searcher constructed three separate queries, evaluated them individually and then together as one set. Table 16 gives the steps Searcher 07 went through in developing a final query for topic 122.

This searcher made no use of automatic relevance feedback, and minimal use of full text. For the most part, searcher 07 looked through the full text of documents to make sure that she hadn't missed anything, rather than using them to search for new concepts. Throughout the search, query construction centered around changes in scope rather than on changes in conceptual content of the queries. A total of seven words was used to construct the combined query set.

Searcher 07 used a building blocks approach to query formulation, in which three components were identified, terms selected to represent the components, and finally, the three components merged into one search statement. This is one of the most commonly used approaches to online searching, and it seems reasonable to think that searcher 07 was applying routine query formulation procedures to the INQUERY environment. We characterize the strategy used by searcher 7 as one of "fitting new tools to old habits."

Topic 116 - Partial use of new system features

The strategy employed by searcher 03 for topic 116 is characterized by a combination of manual and automatic query reformulation. S03 began with the following initial query: `#syn(drug generic)`

After 13 iterations, the final query entered by the user in the query window was the following:

```
#uw10( #syn(brand generic) #syn(drug
pharmaceutical))
```

The searcher decided to use relevance feedback, so the final query as produced through automatic rele-

Iteration	Step	Description
1	1	Formulate query 1 (see text)
	2	Run query
2	3	Display full text of the one relevant document in ranked set
	4	Save query 1 without modification
	5	Reset system and construct query #2:
		#uw100(cancer#syn(treatment research testing evaluation)
	6	Run query
3	7	Fixes syntax error (missing closing parenthesis)
	8	Run corrected query
4	9	Look at titles and relevance judgements of top five documents
		(One relevant document in list)
	10	Save query #2 without modification
	11	Reset system and construct query #3:
		#syn(cancer leukemia)
	12	Run query
5	13	Look at titles and relevance judgements of top five documents
		(One relevant document in list)
	14	Save query #3 without modification
	15	Load query #3; load query 1; load query 2
	16	Run combined query set
6	17	Look at titles and relevance judgements of top five documents
		(All have "?"as relevance judgement)
	18	Display full text of document ranked #2 in set
	19	Reset system
	20	Load query #2; delete term "treatment"
	21	Run modified query #2
7	22	Display full text of document 1 in ranked set (relevant)
	23	Display full text of document 4 in ranked set (?)
	24	Display full text of document 5 in ranked set (?)
	25	Save new query #2:
		#uw100(cancer#syn(research testing evaluation))
	26	Load query #3; load query #1
	27	Run combined query set
8	28	Scroll through retrieved documents,
		and look at 4 full-text documents
	29	Save combined query set as final

Table 16: Query Formulation Process for Topic 122 (Searcher 7)

vance feedback was the following:

```
#WSUM(1.000000 20.072113 #UW10( #SYN(
brand gener) #SYN( drug pharmaceut))
9.267074 vitarin 0.763216 bolar 5.181479
fda 0.489698 maxzid 0.420534 hemant
2.163789 dyazid 0.317706 hypertens 0.306859
shah 0.458595 pharmac)
```

In contrast to searcher 07, S03 made some use of new system features. Relevance feedback was used in three iterations; 14 full text documents were looked at in seven of the 13 iterations. A total of 12 search terms were selected by the searcher, and an additional four were added by the system, through relevance feedback. Searcher 03 went through the query formulation moves described in table 17

This example search illustrates some of the ways in which searcher 3 combined manual query expansion techniques (steps 4,8,12) with query expansion through automatic relevance feedback. In these steps, S03 repeatedly used a strategy of running a query, looking at retrieved texts, and then modifying the query. In step 21 the searcher uses relevance feedback to run a new query, and then manually adds a new term. The pattern of interaction in this search suggests that Searcher 03 did not have a complete understanding of how to use the new system, and in particular, how to use relevance feedback.

The example of topic 116 illustrates a search strategy which begins with routine manual query formulation procedures, and then attempts to incorporate some of the automatic query expansion techniques found in INQUERY. We characterize this strategy as an attempt to combine old and new models of systems and search strategies.

Topic 107 – Effective Use of New Features Topic 107 has been selected as an example of a one searcher's ability to effectively use features of INQUERY. Searcher 08 began this search with the following query formulation:

```
#uw300(#uw3(insider trading)japan)
```

After five iterations, the final query as entered in the query window was the following:

```
#uw300(#uw(insider trading) japan#syn(law
regulations guidelines legislation
#uw3(self regulation)))
```

The searcher decided to use relevance feedback, so the final query as produced through automatic relevance feedback was the following 120 term query:

```
#WSUM(1.000000 2.658688 #UW300( #UW3(
insid trade) japan #SYN( law regul guidelin
legisl #UW3( self regul))) 2.621457 profit
2.841300 corpor 2.116411 osaka 2.873363
loss 9.742378 #foreigncountry 1.359818
unlist 9.114288 exchang 3.300622 disclosur
6.330032 compan 2.571121 client 1.268526
diet 1.321397 shimakura 2.059871 broker
  [...91 terms left out...]
  1.226791 know 11.650723 ministr 3.683496
foreign 4.168829 aide 11.580430 tokyo
1.515035 kiich 1.579114 question)
```

Searcher 8 used the scroll bar to move through the document list, and looked at a total of 268 unique titles over the course of four iterations. He consulted only one full text document. Relevance feedback was turned on during two iterations, and 85 documents were selected by S08 as relevant. The query formulation process for topic 107 (Searcher 8) is depicted in table 18.

Summary These example interactions illustrate three different ways in which experienced online searchers in our study adapted to a new retrieval system, INQUERY. Two of the searchers were able to adapt to the new retrieval mechanism, one by using new system features in ways which supported her routine searching strategies, and the other by developing new searching behaviors which matched the capabilities of the system. A third searcher tried to use both traditional query formulation strategies and automatic features of the new system, with less success.

The data in our study do not enable us to say what predicts successful adaptation to a new retrieval system environment. From the descriptive evidence we have presented, it seems as if Searcher 7 understood how to use specific features of INQUERY, such as proximity and synonym operators, in ways which supported her routine searching strategies in boolean systems. This searcher made the new system work for her; or we might say that she applied new tools to

Iteration	Step	Description
1	1	Formulate query #1 (above)
	2	Run query
2	3	Look at top 25 titles and their relevance judgements
	4	Delete query 1; replace with: #uw200 (#syn(nomenclature #3(chemical formula) #3(brand name) #3(generic name)))
	5	Run query 2
3	6	Look at top five titles and relevance judgements
	7	Display full text of one relevant document
	8	Add one term to query 2, to form new query 3
	9	Run query 3
4	10	Correct spelling "drugnomenclatura" to "drug nomenclatura"
	11	Run corrected query
5	12	Look at top five titles and their relevance judgements
	13	Display full text of one relevant document
	14	Delete query 3 and replace with: #uw10(#syn(brand generic)#syn(drug or pharmaceutic))
	15	Run query
6	16	Look at top fifteen documents
	17	Display four full text documents
	18	Automatic relevance turned on; one document selected
	19	Run query with relevance feedback
7	20	Look at top five titles and their relevance judgements
	21	Display 2 full text documents
	22	Reformulate query: #uw10(#syn(brand generic chem)#syn(drug or pharmaceutic)
	23	Run Query
8	24	Relevance feedback on, add one document
	25	Run query
9	26	Relevance feedback on, delete 1 document, add 1 document
	27	Relevance feedback off
	28	Fifteen titles and relevance judgements looked at
	29	One full text document displayed
	30	Run Query
10	31	Relevance feedback off, 1 document selected
	32	Run query
11	33	Relevance feedback off, 1 additional document selected
12	34	Relevance feedback turned on, query run
13	35	Save last query as final query

Table 17: Query Formulation Process for Topic 116 (Searcher 3)

Iteration	Step	Description
1	1	Formulate query (above)
	2	Run Query
2	3	Relevance feedback turned on
	4	Scroll through the document list look at 90 titles (50 unique)
	5	Mark 31 documents relevant for relevance feedback
	6	Retrieve 1 full text document
	7	Run query
3	8	Relevance feedback turned off; save query
	9	Look at 65 titles; 41 of which are unique
	10	Modify query by adding 6 new terms. New query: #uw300(#uw(insider trading) japan#syn(law regulations guidelines legislation #uw3(self regulation)))
	11	Run query
4	12	Relevance feedback turned on; 23 documents added
	13	Run query (step 10) with relevance feedback on
5	13	save query as final query

Table 18: Query Formulation process for topic 107 (Searcher 8

old habits. Searcher 8, on the other hand, appears to have learned new searching behaviors through his interaction with INQUERY. Rather than adapting the new system to his old behaviors, Searcher 8 changed behaviors to fit the new system environment. S08 appears to have understood the concept of automatic relevance feedback, and the procedures for using it in INQUERY. Our last searcher, S03, did not change routine searching behaviors to fit the new system environment, and she did not use features of INQUERY to support her old habits. It appears as if searcher 3 did not have a well formed model of the new system, and her attempts to use it were based upon trial and error experiences.

Based upon this descriptive evidence, we suggest that a future research goal might be the investigation of factors which constrain or facilitate the adoption of effective new system models.

4 Conclusions

One conclusion that we can draw immediately from our data is that searchers in this study have been able to use the interactive features, including relevance feedback, offered by the INQUERY IR model and the interface which we provided. Furthermore, it appears that the combination of automatic and manual query reformulation can be quite effective, if used appropriately. From our few extended examples, we have some reason to believe that appropriate use of the reformulation and interaction facilities or capabilities might depend upon the ability of the searcher to develop an appropriate mental model of the IR system.

One somewhat problematic result was the seemingly negative effect of extent of interaction, as measured by number of iterations, on retrieval performance. Although this effect is conflated, in our study, with the hardness of the topic being searched, there still seems to be some possibly significant effect. This remains for us an open question, but one which might possibly be answered by analysis of the thinking aloud protocols associated with the searches, which we are now beginning.

Perhaps the most immediately striking result of our study is the consistently poor performance of the Rutir1 final queries, with respect to the median performance of all TREC-3 systems. Since this is also a result for all of the other interactive systems in TREC-3, it seems worthy of some comment.

Our searchers basically took words from the topic descriptions as the source for their queries. This is also precisely what the automatic systems did. Perhaps because of a lack of familiarity with the INQUERY system, or with the relationships between language in full-text documents and query effectiveness, or because of lack of good tools to support reformulation, our searchers tended not to exhibit much of what one might expect to be the potential advantage of human searchers in the interactive environment, namely, exploration leading to new ways to formulate the information problem.

The automatic systems, on the other hand, started with very rich queries (that is, basically, all of the terms in the long topic descriptions), and used them precisely in ways that were tailored to their various retrieval techniques. Since the TREC topics are unusually long, in terms of what one might ordinarily expect as input to a system from a user, and in terms of our searchers' starting queries, this alone might explain some of the difference in performance between automatic and interactive systems. It might indeed be possible to interpret the original TREC topics as the result of some substantial user interaction in an IR system (indeed, at least some topics were produced in precisely this way), and thus the automatic systems are working on top of a previous interaction. But this reasoning is all highly speculative, and the issue clearly needs to be addressed explicitly in further studies.

Another issue that needs to be more directly addressed is the relationship of the searcher to the information problem for which the search is being conducted. In an automatic system, this may not have a terribly strong effect on performance, but it seems reasonable to suppose that searching proceeds differently for persons who are doing searching for their own problems, than for persons who are doing delegated searches (e.g. see [Shenouda1990] and [Spink1994]). Thus, the entire experimental framework of TREC may be working against appropriate evaluation of end-user interactive searching.

This brings us to our final comment. That is, the experience of doing a study of an interactive system with human searchers in the TREC context leads us ever more strongly to the conclusion that we need to develop new evaluation measures and methods for interactive IR for end users. One such change might be to have searchers doing ad hoc style searches for their own information problems, or problems which are salient to them in other ways. Another might be to address in some way the idea that queries in such systems are being progressively formulated, rather than progressively better specified. And something that in our case can be done immediately, is to make use of the searchers' comments in the thinking aloud task in order to understand better their problems and their behaviors. This last point is one that we are now taking up, in order to address many of the open issues that have been raised in these concluding comments.

Whatever changes might eventually result in methods for studying interactive IR, it seems clear to us that the experience of our interactive systems studies in TREC-3 has been highly beneficial in several ways. It has given us an early, and important look into how human searchers interact in best-match IR systems with relevance feedback, the context in which we can expect most IR to take place in the future. It has in particular suggested several issues that will need to be addressed in order to make these systems more relevant to this context, and in order to make them more supportive of users. And, it gives us some hope that we will be able to design IR systems which directly support effective human interaction.

Acknowledgements

We owe special thanks to the searchers who volunteered to participate in this study, and to Jamie Callan, Bruce Croft and Steve Harding of the Center for Intelligent Information Retrieval at the University of Massachusetts for their unstinting support of our use of INQUERY, and for running the final queries on the test collection.

References

[Belkin & Croft1987] Belkin, N. J. & Croft, W. B. (1987). Retrieval techniques. In Williams, M. E., editor, *ARIST*, chapter 4, pages 109–145. Elsevier.

[Callan et al.1992] Callan, J. P., Croft, W. B., & Harding, S. M. (1992). The inquery retrieval system. In *DEXA 3: Proceedings of the Third International Conference on Database and Expert Systems Applications*, pages 83–87. Berlin, Springer Verlag.

[Frei & Qiu1993] Frei, H. & Qiu, Y. (1993). Effectiveness of weighted searching in an operational ir environment. In *Information retrieval 93: von der Modellierung zur Anwendung; Proceedings der 1. Tagung Information Retrieval '93*, pages 41–54. Konstanz, Universitaetsverlag Konstanz.

[Harter1986] Harter, S. P. (1986). *Online Information Retrieval.* San Diego, CA, Academic Press.

[Ousterhout1994] Ousterhout, J. K. (1994). *Tcl and the Tk Toolkit.* Reading, MA, Addison-Wesley.

[Pearl1988] Pearl, J. (1988). *Probabilistic reasoning in Intelligent Systems: Networks of plausible Inference.* San Mateo, CA, Morgan Kaufmann.

[Salton & Buckley1990] Salton, G. & Buckley, C. (1990). Improving retrieval performance by relevance feedback. *JASIS*, 41(4):288–297.

[Shenouda1990] Shenouda, W. (1990). Online bibliographic searching: how end-users modify their search strategies. In *Information in the year 2000: from research to applications. Proceedings of the 53rd Annual Meeting of the American Society for Information Science*, pages 117–128. Learned Information, Inc.

[Spink1994] Spink, A. (1994). Term relevance feedback and query expansion: relation to design. In *SIGIR '94. Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 81–90. London, Springer Verlag.

[Turtle1994] Turtle, H. (1994). Natural language vs. boolean query evaluation: A comparison of retrieval performance. In *SIGIR'94. Proc. of the Seventeenth Annual ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 212–220. London, Springer Verlag.

APPENDIX: rutir1 INTERATIVE SYSTEM DESCRIPTION

I. System description

I.1 Screen dump of "typical" screen

Figure 3 shows a typical state of the Rutgers XINQUERY interface during a search. Individual features are discussed below.

I.2 Usable features of the interface

I.2.a General

In our experiments we used the INQUERY retrieval engine (version 1.6.3), developed at the University of Massachusetts (Turtle Croft, 1991). We used the INQUERY TK user interface, XINQUERY, developed at the University of Massachusetts with a number of modifications. Most actions are carried out through keyboard-based and mouse-based interaction with the main Tkinq window (I.1) which is comprised of four subwindows and a number of interaction devices. Below we describe the various funcionalities of the interface, and specify the particular feature whose use we have monitored, in **UPPERCASE**.

I.2.b Query formulation features:

On the top right is a query window with limited text editor functionality. Searchers use this window to enter and edit their queries. Queries can extend over multiple lines and a scrollbar allows scrolling through complex queries that extend beyond the size of the query window.

SAVE QUERY, LOAD QUERY Buttons above the query window can be used to save a query to file or to load a previously entered query into the query window. In either case, a dialog window with a list of current filenames and default names will appear. The user can select an existing name or enter a new name. The text of a loaded file will be appended to the current contents of the query window, allowing for the combination of previously stored query segments. Clicking on the "Reset ALL" button will clear the query window as well as all other information from previously executed queries.

Figure 3: The Rutgers XINQUERY user interface

To execute a query the searcher enters (or loads) a query and clicks on the "Run Query" button above the query window. If the query is syntactically incorrect a warning message and a separate error window will pop-up. The error window will give extremely limited feedback on the type and location of a syntax error (if at all). These windows remain on the screen until the users dismisses them explicitly. If the query is syntactically correct it will be processed further by the INQUERY system. Status messages will appear that indicate that a query is processed, the number of terms that are evaluated, and that the resulting list of retrieved documents is sorted. Upon completion, the summary window (see below) will be updated with the results from the current query.

I.2.c. Search results

The window in the middle is a summary window that gives summary information on five documents retrieved by the system in response to a query. It displays the rank of a document, the belief value assigned by the system, a relevance rating of the document as assigned by the TREC raters (if available), its external document ID, and the title of a document.

SCROLLBAR IN SUMMARY Users can scroll through the ranked list of retrieved documents by means of the scrollbar to the right of the summary window.

SCROLLBAR IN DOCUMENT WINDOW Searchers can also use the scrollbar on the lower right to change the summary window. As the user is dragging the scroller, feedback about the rank of the about-to-be-selected document (that will appear on the top of the summary window) is provided. Both scrollbars are very sensitive if large numbers of documents have been retrieved and direct scrolling to a particular rank in the list is not (easily) possible; triangular buttons at the respective ends of the summary window scrollbar need to be used to change the ranks displayed in the summary window by increments/decrements of five. An indicator under the summary window gives the total number of documents retrieved by the current query.

DOCUMENT SELECTED/UNSELECTED (AS RELEVANT) To the left of the titles are toggle buttons that allow a searcher to mark individual (or groups of) documents.

RELEVANCE FEEDBACK ON/OFF Directly above the summary window is the automatic relevance feedback toggle. If the toggle is "off" (the default), markers only serve the orientation of the user. If automatic relevance feedback is turned "on", every marked document is used as a source in the computation of a relevance feedback query.

DOUBLE-CLICK ON TITLE The bottom half of the window consists of the document window. Users can select a document by double-clicking on its title. As a result, the full text of the document is displayed in the window. A scrollbar can be used to scroll through the document if the document exceeds the given height of the window. Each word in the full text of the selected document who's stem matches a stem of a term in the query is highlighted in the document.

KEYWORD The "scroll to Keyword" button under the document allows a searcher to scroll to the next highlighted term in the document (if any).

NEXT DOCUMENT, PREVIOUS DOCUMENT A searcher can directly display the full text of the next or previous document by clicking on the respective buttons under the document window.

BARGRAPH On the top left is a bar graph window that depicts the belief values assigned to the retrieved documents in graphical form. Each graph depicts a range of 100 documents (by default the top 100 documents). Searchers can use the scrollbar under the bargraph window to bring other portions of these 100 documents into view. The bargraph will be updated to depict a different range of the ranked list if a document outside the current range is selected. The two numeric values represent the belief values assigned to the highest and the lowest ranked documents in the current graph respectively. Each

bar represents one document and the height of the bar corresponds to its belief value; heights are scaled relative to the belief value assigned to the document ranked lowest in the current graph, i.e. the rightmost document in the bar graph has a bar height of 0. The bar associated with the currently selected document is hatched. If a document is marked by the searcher (with automatic relevance feedback being either on or off) will appear grayed. A searcher can bring a document into full text view by double-clicking on the associated bar in the bar graph.

A user quits the inquery system by clicking on the top left "Quit" button.

I.3 Style of Interface:

Graphical User Interface (GUI) (see also section I.2)

II Experimental conditions

II.1 Searcher Characteristics

a. **Number of searchers in experiment**: 10

b. **Number of searchers per topic**: 1

c. **Age/age group of searchers.**

Age	n (searchers)
Under 21	0
21-30	3
31-40	3
41-50	4
Over 50	0

d. **IR searching experience of searchers**

We measured four dimensions of searching experience in this study:

- **Years of searching experience:** Mean: 6.1 years, Range: 1–15

- **Frequency of online searching**

Searching Frequency	n (searchers)
More than twice a day	3
Once or twice a day	1
More than twice a week	0
One or two times a week	4
More than twice a month	0
One or two times a month	1
Less than monthly	1

- **Experience searching full-text databases**

Experience searching full-text databases was measured by responses to a five point Likert scale, in which 1= None at all, 3= Some and 5= A great deal.

Full-text Experience	n(searchers)
1 (None at all)	2
2	2
3 (Some)	5
4	0
5 (A great deal)	1

- **Experience searching in ranked-output information retrieval systems**

The same five point Likert scale was used to measure searchers' experience using ranked-output retrieval systems, as full-text systems.

Ranked-output Experience	n (searchers)
1 (None at all)	8
2	1
3 (Some)	1
4	0
5 (A great deal)	0

e. **Educational level of searchers**

Searchers were asked to list all of their college or university degrees.

All of the searchers had, or expected, the Masters degree in Library Science. Four of our

searchers had graduate degrees in addition to the MLS. Two searchers had a JD; two had, or expected to obtain, a PhD; and two searchers had an additional masters degree.

f. **Undergraduate major of searchers**

Major	n
English	1
History	1
Psychology	2
Political Sci	1
Zoology	1
Math	1
Chemistry	1

Two of the searchers did not list their undergraduate major. One of these searchers had a JD; the other searcher had a PhD in Oceanography.

g. **Experience/familiarity with subject of topic**

After completing each search topic, subjects were asked about their familiarity or experience with the subject matter covered in that topic. A five point Likert scale was used to record responses to the question, "Is this topic related to things you normally search on?" where 1=Not at all; 3= Somewhat; and 5=Very much. Overall, the Trec topics were not routine subject matter for the subjects who searched them. The mean familiarity value for all 50 topics, across all searchers is 2.0. The distribution of topics across the scale is the following:

Familiarity	n (topics)
1 (Not at all)	20
2	13
3 (Somewhat)	12
4	3
5 (Very much)	2

Work affiliation:	n (searchers)
Academic Library	6
Industrial/Research Lib	2
Research Lab	1
Full time MLS student	1

II.2 Task description

Below are the verbatim instructions for the task set our searchers. The cut-off level for each topic was written on the topic description.

THE DOCUMENT ROUTING TASK

Task and context

Imagine a situation in which you are an intermediary using the INQUERY system to satisfy the information needs of a user. Rather than being a one-time affair, the user wants you to develop a query he or she can use repeatedly in the future to retrieve documents of interest. For example, the user may want to reissue the same query every month to stay up-do-date on developments in his or her field. Thus, your goal is to develop a query that not only retrieves relevant documents now (and ranks them reasonably high) but that will work equally well on future document collections with different documents.

Since ranking systems like INQUERY may retrieve large numbers of documents, the user will not be able to look at all documents the system retrieves. The number of documents that a user might be willing to look at will vary by each information problem. The user will tell you for each topic how many documents the user is willing to read. The user will not consider any document ranked below this number.

Topic Description

The user has prepared a statement that describes in detail the topic he or she is interested in. This statement also contains a description of the kinds of documents the user considers to be relevant and those s/he considers not to be relevant. Assume that the user has left you with this written statement, i.e. you cannot get clarifications and you cannot negotiate changes.

For the purposes of training we are using a somewhat outdated document collection. Because of this, some topics will appear outdated as well. Please act as if you are doing this search at a time in which the topic would have been relevant.

User Judgments

To help you with your task, the user has looked through parts of the currently available document collection and has noted which of the documents are relevant and which are not relevant. These ratings have been made available to you in the INQUERY system. The summary for each document has a notation after the belief value that states the judgement of the user: documents that the user found to be relevant are marked "REL" and documents the user

judged not- relevant are marked "NOT". If the user has not judged a document it is marked as "?". The judgements are also given above the full display of a document. Again, since the user is not currently present, you should accept ratings as they are.

- Reset the system
- Develop a query for the topic description handed to you.

  Your query should retrieve as many documents as possible among the top number of documents that the user is willing to look at which satisfy the specified information needs. The query should also be likely to do will on future document collections.

  NB: Each topic had a routing cut-off written on the topic description.

  You may use any of the system features (including automatic relevance feedback) you have learned about, but you are not required to do so.

  You have up to 20 minutes to complete this task.

  We are interested in how people interact with the INQUERY system. Please think-aloud throughout this session, i.e. we ask you to verbalize any thoughts, comments, plans, or goals related to what you are doing and why you are doing it as they come to mind. Act as if you were talking to yourself (rather than talking to us). Occasionally we may remind you to keep talking.
- Once you are satisfied with your result, save your final query under the name "final".

II.3 Training

II.3.a Description of the training process

Subjects were given a short general description of the TREC project and were given a consent form that outlined the experiment and the data collection. Upon signing of the form they were led to a screened-off area that was set up with the computer equipment and a video recording device. The experimenter gave a 2-minute introduction to INQUERY by listing its major features, namely ranked output (rather than set based retrieval), full text search (rather than title, abstract, or keyword search), the availability of automatic relevance feedback functionality, and the ability to process free form, unstructured queries. Subjects were next given a 14 page, ring-bound tutorial. The tutorial contained a sequence of interaction exercises that demonstrated the various interface features, the use of operators, and the use of automatic relevance feedback. Subjects set their own pace working through the tutorial. One experimenter was seated next to the subject and provided additional help and guidance if the subject got stuck or had questions that were not answered by the tutorial. The experimenter did not carry out any interactions with the system except for occasional restarts of the system necessitated by system failures; most interventions by the experimenter were short interjections that made sure that the intended learning experience took place. The document collection used in the tutorial was disk 3 of the TREC test collection and all examples were taken from the implicit context of topic 77 (poaching of wildlife). Subjects completed the tutorial's main portion (the first eleven pages) in 50-70 minutes.

The last part of the tutorial (after a short break, if desired) consisted of a test trial of an actual search. Subjects received the same task description used later in the actual trials (see II.2) and practised on a topic that was not part of the TREC-3 topic set, namely topic 84 (Alternative/Renewable Energy Plant Equipment Installation). The document collection was again the TREC test collection disk 3. Subjects were also asked to "think-aloud" during this phase as this was a requirement during the later search trials. An experimenter operating under the same model described above was available during this trial search as well. No feedback was given by the experimenter regarding the quality of individual queries nor regarding the quality of the overall search session. Subjects were asked after 15 minutes to wrap up their search and to save a final query formulation.

II.3.b Time for training

Time for training in minutes (times are approximate whole minute): Mean:80, Range: 65 - 90

III Search Process

1. Clock time (i.e. real, elapsed time) per search, from the time the searcher was given the topic, until the final query was saved, in seconds. Note that users were restricted to 20 minutes (1200 seconds).

Mean: 928.6, Median: 981.0, SD: 245.24, Range: 421 - 1328

2. Number of documents "viewed" in during the search.

a. Definitions of viewing:

We report on three categories of viewing, as defined below:

1. "All titles" defined as the total number of document titles displayed during the course of the entire search (the default display is five titles at a time). This can be thought of as title "tokens".

2. "Unique titles", defined as the number of unique titles seen by the user during the course of the entire search. This can be thought of as title "types".

3. "Text", defined as the number of documents for which the user had the text of a document displayed.

b. Number of items viewed per search (repeat for each viewing category).

Viewing Behavior				
Viewing Type	Mean	Median	SD	Range
All titles	136.3	94.5	118.2	15-515
Unique titles	67.2	51.0	55.6	8-310
Text	6.6	5.5	6.2	0 - 36

3. Number of iterations per search.

a. Definition of iteration:

An iteration is all that occurs between one invocation of the "run query" button and the next invocation of that button. The first iteration begins when the user is given the topic for searching, and the last iteration ends when the user invokes the "save query" button, saving the query as "final". The query run by an invocation of the "run query" button is part of the iteration defined by that invocation as the end point.

b. Number of iterations per search.

Mean: 7.76, Median: 7.0 SD:3.98, Range: 2 - 21

4. Number of terms used in queries.

a. Number of terms in the first query of a search, per search.

Mean: 5.38, Median: 4.0, SD: 3.90, Range: 1 - 21

b. Number of terms in the final query of a search, per search.

Mean: 23.82, Median:18.0, SD: 22.50, Range: 3 - 120

5. Use of system features.

The following table provides summary statistics for the use of system features, $N = 50$ topics (10 searchers).

Use of System Features				
Feature	$\overline{X}$	M_d	SD	Range
bargraph double-click (to view fulltext)	0	0	0	0
scrollbar in summary	19.34	10.5	21.9	0–95
scrollbar in document window	2.72	2.0	3.6	0–20
double-click on title (to view fulltext)	4.82	4.0	4.0	0–16
next document (to view fulltext)	1.14	0.0	3.1	0–20
previous document (to view fulltext) 5 uses by 4 searchers	0.1	0	0	0–2
keyword	6.48	0.5	11.4	0–41
save query	2.14	1.0	1.8	1–7
load query	0.72	0.0	1.8	0–8
relevance feedback on	1.08	1.0	1.0	0–5
relevance feedback off	0.38	0.0	0.8	0–4
relevance feedback used (number of iterations/search)	1.68	1.0	1.9	0–7
documents selected (as relevant)	8.26	5.0	9.9	0–54
documents unselected	0.50	0.0	1.2	0–7

6. Number of user errors made per search

a. Definition of an error

Error is defined as the use of syntax in a query which caused the query to be uninterpretable by the system query parser (the most common syntax error was to not include a required closed parenthesis).

b. Number of user errors per search

Mean: 0.7, Median: 0, SD:1.18, Range: 0 - 5

7. Transcript for Topic 122 (Searcher 7)

What follows is the log of the interaction for topic 122 (searcher S07, trial 4. Given are the elapsed time, the real time, a description of the searcher's actions, system responses and experimenter comments, Statements enclosed in "" constitute the complete verbal protocol of the searcher. The full text of viewed documents is reproduced for short documents, for long documents only the approximate amount that fit on the screen and was visible without scrolling is given.

```
Iteration 1:
00:00:00 11:46:34  START topic from VIDEO
00:00:15 11:46:49  starts entering query, enters "uw50" (misses # in uw)
00:00:26 11:47:00  backspaces over uw50 and replaces it with #uw50(
00:00:54 11:47:28  adds term #syn and term - pauses: #uw50(cancer #syn (drug

                   "That is something I don't understand about this system..."
                   "chemotherapy ... treatment ..."
                   enters complete query:
                   #uw50 (cancer #syn (drug chemotherapy treatment))
00:01:28 11:48:02  hit run query button

Iteration 2:
                   *** parsed query ***
                   #UW50( cancer #SYN( drug chemotherap treatment))
                   *** Query processing: 4 words
00:01:34 11:48:08  ***  new summary  ***
                   1. 0.653723 NOT AP880524-0052 Ex-President's Brother Enters Cancer Institute for Treatment
                   2. 0.653723  ?  AP880322-0156 Experimental Treatment Avoids Surgery for Bladder Cancer
                   3. 0.653723 REL WSJ870415-0071 Cetus Venture Wins Approval to Market Anti-Cancer Drug -
                   4. 0.653723  ?  DOE2-42-0592
                   5. 0.653723  ?  DOE2-38-0875
                   *** 2443 documents retrieved ***
00:01:50 11:48:24  *** Double_Click_Title ***
                   Reads document, voice not audible
                   3. 0.653723 REL WSJ870415-0071

                   Cetus Venture Wins Approval to Market Anti-Cancer Drug ---
                   Partnership With Ben Venue Gets FDA's Clearance To Sell
                   Generic Product --- By Marilyn Chase Sta

                     Cetus Corp. said its joint venture with Ben Venue
                   Laboratories Inc. received
                   approval from the U.S. Food and Drug Administration to begin
                   marketing its second generic anti-cancer product.

                     Cetus, based in Emeryville, Calif., and Ben Venue, Bedford,
                   Ohio, will soon begin marketing vinblastine sulfate, a
                   standard chemotherapy agent extracted from periwinkle
                   flowers and used in the treatment of Hodgkin's disease,
                   lymphomas, testicular cancer, unresponsive breast cancer and
                   other malignant tumors.
                   [...]

00:02:12 11:48:45  hits save query button
                   enters file name "cancerdrug''; queries read:
                       #uw50 (cancer #syn (drug chemotherapy treatment))
                   RF: #UW50( cancer #SYN( drug chemotherap treatment))
00:02:27 11:49:00  clicks on "OK"
00:02:32 11:49:05  chooses "Reset All"
00:02:35 11:49:08  enters "uw" backspaces over it
00:02:37 11:49:10  enters "#uw"  pauses
00:02:48 11:49:21  adds to query to read "#uw100 (cancer"  and pauses
00:03:08 11:49:41  adds #syn( for query to read " #uw100 (cancer #syn(" pauses
00:03:10 11:49:43  completes query to read:
                   #uw100 (cancer #syn(treatment research testing evaluation)
00:03:27 11:50:00  clicks on "Run Query" button
```

```
Iteration 3:
00:03:29 11:50:02  Gets syntax error msg.(small window)
00:03:30 11:50:03  Gets error parsed query window [missed closing parenthesis]
00:03:31 11:50:04  Dismisses  error message
00:03:32 11:50:05  Dismisses large window error msg
00:03:33 11:50:06  "I have an about 50% hitrate on putting the parenthesis"
00:03:36 11:50:09  "I'm also trying to run a demo search system..."
                   adds the closing parenthesis for query to read:
                   #uw100 (cancer #syn(treatment research testing evaluation))
00:03:39 11:50:12  clicks on "Run Query" button

Iteration 4:
                   *** parsed query ***
                   #UW100( cancer #SYN( treatment research test evalu))
                   *** Query processing: 5 words
00:03:46 11:50:19  gets ***  new summary  ***
                   1. 0.632350  ?  AP880322-0156 Experimental Treatment Avoids Surgery for Bladder Cancer
                   2. 0.632350 REL WSJ870519-0014 Tests to Begin on a Toxin That Attacks Cancer Cells
                   3. 0.632350  ?  FR891011-0077
                   4. 0.632350  ?  FR89322-0189
                   5. 0.632350  ?  DOE2-42-0592
                   *** 3954 documents retrieved ***
00:03:48 11:50:21  "A lot of question marks, not a lot of no's (?)"
00:03:53 11:50:26  clicks on "Save Query" button
                   enters file name "cancertreat"; queries read:
                       #uw100 (cancer #syn(treatment research testing evaluation))
                   RF: #UW100( cancer #SYN( treatment research test evalu))
00:04:04 11:50:37  clicks on "OK"
00:04:14 11:50:47  clicks on "Reset All"
00:04:17 11:50:50  starts entering query  "#syn( cancer leukemia )"
00:04:31 11:51:04  "Hmm actually"  picks up topic sheet reads description
                   "these [inaudible]  clients actually...
                   he or she is what I mean is actually the client
                   is actually interested only in the specifics of it [??]...
                   but there are a lot of different forms of cancer ...
                   [inaudible] ... it doesn't matter what ... [sighs]
                   it doesn't matter what kind of cancer it is... Hmm...
                   I don't know what kind of cancer ... [???inaudible]
                   #syn (cancer leukemia )
00:05:20 11:51:53  clicks on "run Query" button

Iteration 5:
                   *** parsed query ***
                   #SYN( cancer leukemia)
                   *** Query processing: 2 words
00:05:24 11:51:57  gets ***  new summary  ***
                   1. 0.593854 REL WSJ920318-0054 Medicine:    Scientists Say Progress Is Made
                   2. 0.593854 NOT WSJ920305-0102 Technology &amp;amp; Health: Colon Cancer Test Cuts Chance
                   3. 0.593854 NOT WSJ920302-0204 Technology &amp;amp; Medicine:    ICI Cancer Drug's
                   4. 0.593854 NOT WSJ920226-0129 Technology:    Medarex Drug Trials Approved
                   5. 0.593854 NOT WSJ920219-0135 Technology &amp;amp; Medicine: Combination of Acne Drug...
                   *** 9412 documents retrieved ***
00:05:26 11:51:59  click on "Save Query" button
                   enter filename "cancerany"; queries saved:
                       #syn (cancer leukemia )
                   RF: #SYN( cancer leukemia)
00:05:32 11:52:05  click on "OK"
00:05:35 11:52:08  click on "Reset All"
00:05:35 11:52:08  click on "Reset All"
00:05:36 11:52:09  "I'm gonna see what I'm getting (?)
00:05:39 11:52:12  click on "Load Query" button
                   selects file "cancerany" which reads:
                   #syn (cancer leukemia )
00:05:43 11:52:16  clicks on "OK"
00:05:49 11:52:22  clicks on "Load Query" button
                   selects file "cancerdrug" which reads:
                   #uw50 (cancer #syn (drug chemotherapy treatment))
00:05:53 11:52:26  clicks "OK"
```

```
00:06:02 11:52:35  clicks on "Load Query" button
00:06:03 11:52:36  move cursor over files, seems unsure which one to load next
00:06:11 11:52:44  selects file "cancertreat" which reads:
                   #uw100 (cancer #syn(treatment research testing evaluation))
00:06:14 11:52:47  clicks on "OK"
                   current content of query window now:
                   #syn (cancer leukemia )
                   #uw50 (cancer #syn (drug chemotherapy treatment))
                   #uw100 (cancer #syn(treatment research testing evaluation))
00:06:19 11:52:52  clicks on "Run Query" button

Iteration 6:
                   *** parsed query ***
                   #SUM( #SYN( cancer leukemia) #UW50( cancer #SYN( drug
                   chemotherap treatment)) #UW100( cancer #SYN( treatment
                   research test evalu)))
                   *** Query processing: 11 words
00:06:23 11:52:56  "Let's see what happens .. comes back ... up [??]
                   ..'cause the ...ah.. "
00:06:29 11:53:02  gets ***  new summary  ***
                   1. 0.626642  ?  AP880322-0156 Experimental Treatment Avoids Surgery for Bladder Cancer
                   2. 0.626642  ?  DOE2-42-0592
                   3. 0.626642  ?  DOE2-38-0875
                   4. 0.626642  ?  DOE2-32-1236
                   5. 0.626642  ?  DOE2-10-1097
                   *** 9412 documents retrieved ***
00:06:31 11:53:04  "[laughs] well comes up with a lot of questions...oh well"
00:06:35 11:53:08  double-click on title 2. full text for doc 2. displayed
                   2. 0.626642  ?  DOE2-42-0592

                   Electron beam therapy has become more widely available in the radiation
                   oncology community. It is a unique modality, offering important contributions
                   to the management of cancer. It is, however, necessary to optimize dose
                   distribution, patent selection, and treatment techniques.

00:06:40 11:53:13  ''at least it didn't come up with instant no's'' [??]
00:06:46 11:53:19  looks at document (evaluates its relevance)
                   "well  [inaudible]  research and development"
00:06:58 11:53:31  clicks on *** Next_Doc *** button, new full text displayed
                   3. 0.626642  ?  DOE2-38-0875

                   Indications and contraindications for radiation treatment of esophagus
                   cancer are presented. The role of chemoradiation among esophagus cancer
                   treatment methods is determined. Thechnical,dosimetric and clinical data
                   are sequently delivered. Preparation of a patient for chemoradiation
                   is described. Recommendations on their most efficient use are given.
                   95 refs.; 15 figs.; 19 tabs.

00:07:11 11:53:44  looks at document full text
                   "treatment is getting some ...
00:07:20 11:53:53  clicks on "Reset All" button
00:07:23 11:53:56  clicks on "Load Query" button
                   selects file "cancertreat" which reads:
                   #uw100 (cancer #syn(treatment research testing evaluation))
00:07:27 11:54:00  clicks on "OK"
00:07:28 11:54:01  "I better modify this"  (deletes the word "treatment"), now:
                   #UW100( cancer #SYN( research test evalu))
00:07:37 11:54:10  clicks on "Run Query" button

Iteration 7:
                   *** parsed query ***
                   #uw100 (cancer #syn( research testing evaluation))
                   *** Query processing: 4 words
00:07:40 11:54:14  "I'm leaving out treatment because it was... hmmm"
00:07:43 11:54:17  gets ***  new summary  ***
                   1. 0.644194 REL WSJ870519-0014 Tests to Begin on a Toxin That Attacks Cancer Cells
                   2. 0.644194  ?  FR891011-0077
                   3. 0.644194  ?  FR89322-0189
```

4. 0.644194 ? DOE2-20-0076
5. 0.644194 ? DOE1-87-0742
*** 3028 documents retrieved ***

00:07:47 11:54:21 double-clicks on title of document 1. Full text gets displayed:

1. 0.644194 REL WSJ870519-0014

Tests to Begin on a Toxin That Attacks Cancer Cells

Cetus Corp. said the first clinical tests on humans will soon begin involving a breast cancer treatment that " targets" diseased cells and attacks them with a toxintipped monoclonal antibody.

Under the treatment, a toxin will be bonded to antibodies that recently were proven capable of locating and attaching themselves to cancer cells in humans. Scientists hope the toxin will destroy the targeted cells without the damage to healthy tissue that accompanies conventional chemotherapy cancer treatments.
[...]

An official of Cetus, a biotechnology concern that makes both the toxin and the antibody, stressed that the initial test will involve only a few patients. If it is successful, he said, the tests will be expanded, but he added it will probably be " several years" before the company would be prepared to seek Food and Drug Administration approval to market the product.
The treatment has proven successful in tests on animals, he added.

Cetus, which said it expects to begin similar tests on ovarian cancer in humans later this year, noted that other companies are testing the process on other cancers,including melanoma and colon cancer.

00:07:49 11:54:23 reads through document, moves cursor over full text.
00:08:04 11:54:38 "I'm looking for any added term I could conceivably use to
00:08:07 11:54:41 clicks on "Scroll to Keyword" button
00:08:10 11:54:44 clicks on "scroll to Keyword" button (no change in display (?))
00:08:15 11:54:49 moves cursor into bargraph, onto scroll in bar graph
but reconsiders and moves cursor back into summary window
00:08:18 11:54:52 double-clicks on title for doc 4. Full text displayed:

4. 0.644194 ? DOE2-20-0076

In this paper, the emphases are put on the description of design principle and calculation method for radiation protection in the new radiotherapy department of Cancer Research Institute and Hospital, Chinese Academy of Medical Science, as well as the evaluation of meusaring results. In addition, the problem of photo-neutron contamination amcng 16 MV X-ray from SL 75-20 Philips Linear Accelerator and its relavent shielding measures have been discussed.

00:08:23 11:54:57 "I'm looking at question mark documents to see...
"radiation protection [from doc text] that's not relevant"
00:08:36 11:55:10 double-clicks on title for doc 5. Full text displayed:
00:08:37 11:55:11 reads doc full text (cursor in full text window)

5. 0.644194 ? DOE1-87-0742

High Performance Gel Permeation Chromatography (GPC) was evaluated as an alternative to the more expensive Nuclear Magnetic Resonance (NMR) spectroscopy technique for cancer detection using human plasma. These two techniques show a biphasic relationship which can be explained on the basis of the relative amounts of the lipoprotein levels present in the plasma and a good correlation with total triglyceride concentrations obtained from standard blood tests. The major difference in the GPC elution nprofiles (254 nm) of plasma from normal individuals and that from cancer patients occurred in the peak eluting at the void volume. This peak has a retention time consistent with very low density lipoprotein (VLDL) and is elevated in most cancer patients and in normal patients with triglyceride levels greater than 200 mg/ml. The use of these techniques as a screening test for cancer in an asymptomatic population needs further evaluation.

00:08:54 11:55:28 clicks on "Save Query" button
edits default file name (add 2) to "cancertreat2 ", which reads:

```
                    #uw100 (cancer #syn( research testing evaluation))
                  RF: #UW100( cancer #SYN( research test evalu))
00:09:01 11:55:35 clicks on "OK"
00:09:06 11:55:40 "So I'm gonna to try again having left out treatment"
00:09:07 11:55:41 clicks on "Load Query" button
                  selects file "cancerany" which reads:
                  #syn (cancer leukemia )
00:09:12 11:55:46 clicks on "OK"
00:09:15 11:55:49 "Lets see what I get"
00:09:16 11:55:50 clicks on "Load Query" button
                  selects file "cancerdrug" which reads:
                  #uw50 (cancer #syn (drug chemotherapy treatment))
00:09:21 11:55:55 clicks on "OK", contents of query window now:
                  #uw100 (cancer #syn( research testing evaluation))
                  #syn (cancer leukemia )#uw50 (cancer #syn (drug chemotherapy treatment))
00:09:25 11:55:59 "The more I use this the more confused I get"
00:09:30 11:56:04 "Put a return "
                  adds a return that puts last loaded query on newline:
                  #uw100 (cancer #syn( research testing evaluation))
                  #syn (cancer leukemia )
                  #uw50 (cancer #syn (drug chemotherapy treatment))
00:09:31 11:56:06 *** Run_Query ***

Iteration 8:
                  *** parsed query ***
                  #SUM( #UW100( cancer #SYN( research test evalu)) #SYN( cancer
                  leukemia) #UW50( cancer #SYN( drug chemotherap treatment)))
                  *** Query processing: 10 words
00:09:40 11:56:15 gets ***  new summary  ***
                  1. 0.625523 REL WSJ870519-0014 Tests to Begin on a Toxin That Attacks Cancer Cells
                  2. 0.614311 REL WSJ910412-0102 Technology &amp;amp; Medicine: Cancer Drug Taxol May Wor
                  3. 0.614101 NOT AP881023-0047 New Cancer Treatment Tested On Two Patient
                  4. 0.613794 REL WSJ900419-0016 Technology &amp;amp; Medicine:    New Drug Shows Promise Tr
                  5. 0.612908 NOT WSJ870424-0155 Whose Life Is It Anyway? --- By Robert K. Oldham
                  *** 9412 documents retrieved ***
00:09:45 11:56:20 "Hmm... a little better"
00:09:46 11:56:21 Double-click on title for document 4. Full text displayed:
                  4. 0.613794 REL WSJ900419-0016

                  Technology &amp;amp; Medicine:    New Drug Shows    Promise
                  Treating    Type of Leukemia    ----    By Ron Winslow
                  Staff Reporter of The Wall Street Jour

                  A new cancer drug proved strikingly successful with very
                  few side effects in treating a relatively rare form of
                  leukemia, researchers said.
                     The drug, called 2-CdA, was nearly 100% effective treating
                  12 patients with hairy-cell leukemia. The patients underwent
                  just one seven-day treatment and experienced almost none of
                  the debilitating side effects such as vomiting, hair-loss and
                  kidney and liver problems associated with other chemotherapy.

                     The results need to be replicated by other researchers
                  before definitive conclusions can be made, and the long-term
                  effect of the treatment remains unknown. But the initial
                  experience with the compound is especially promising, and an
                  apparent improvement over other therapy for the disease,
                  scientists said.
                  [...]
00:09:56 11:56:31 "Okay It gets as good as I'm going to be able to get"
                  "It won't   [inaudible]''  [laughs]
00:10:02 11:56:37 scrolls outside doc counter to top, new full text display:
                  1. 0.625523 REL WSJ870519-0014

                  Tests to Begin on a Toxin That Attacks Cancer Cells

                      Cetus Corp. said the first clinical tests on humans will soon begin involving
                  a breast cancer treatment that " targets"  diseased cells and attacks them with
```

a toxintipped monoclonal antibody.

Under the treatment, a toxin will be bonded to antibodies that recently were proven capable of locating and attaching themselves to cancer cells in humans. Scientists hope the toxin will destroy the targeted cells without the damage to healthy tissue that accompanies conventional chemotherapy cancer treatments.
[...]

00:10:10 11:56:45 scroll outside doc counter: new summary and full text displayed:
00:10:11 11:56:46 gets *** new summary ***

79. 0.587702 NOT DOE1-29-0593
80. 0.587600 NOT WSJ911226-0005 Technology & Health: Many Women With Breast Cancer
81. 0.587433 NOT WSJ910405-0005 Technology: Genentech Inc. Cancer Therapy
82. 0.586518 NOT AP880525-0051 Signs That Chemotherapy May Help Control Lung Cancer
83. 0.586406 NOT WSJ911107-0058 Business Brief -- Schering-Plough Corp.: Company Reite

79. 0.587702 NOT DOE1-29-0593

The combination of radiotherapy (RT) and chemotherapy (CT) has markedly improved the therapeutic results for those tumors which are both chemosensitive and radiosensitive, such as lymphomas, embryonal tumors, small cell lung carcinomas, breast cancers, etc. Despite some spectacular results reported following non-controlled studies, a significant increase in the total survival or the relapse free survival has never been documented in controlled trials in head and neck, anal, ovarian carcinomas. However, in these tumors, a combination of RT and CT may reduce the mutilations and sequellae caused by the treatment and may induce an increase in the survival in some subsets of patients. Further clinical research is needed along these lines. Cross resistance between ionizing radiation and
[...]

00:10:13 11:56:48 "[still laughing] oh you know .. there are some there .. there are not a 150 there"
00:10:19 11:56:54 scrolls outside doc counter to top: new summary and full text:
00:10:20 11:56:55 gets *** new summary ***

1. 0.625523 REL WSJ870519-0014 Tests to Begin on a Toxin That Attacks Cancer Cells
2. 0.614311 REL WSJ910412-0102 Technology & Medicine: Cancer Drug Taxol
3. 0.614101 NOT AP881023-0047 New Cancer Treatment Tested On Two Patients
4. 0.613794 REL WSJ900419-0016 Technology & Medicine: New Drug Shows
5. 0.612908 NOT WSJ870424-0155 Whose Life Is It Anyway? --- By Robert K. Oldham

1. 0.625523 REL WSJ870519-0014
Tests to Begin on a Toxin That Attacks Cancer Cells

Cetus Corp. said the first clinical tests on humans will soon begin involving a breast cancer treatment that " targets" diseased cells and attacks them with a toxintipped monoclonal antibody.

Under the treatment, a toxin will be bonded to antibodies that recently were proven capable of locating and attaching themselves to cancer cells in humans. Scientists hope the toxin will destroy the targeted cells without the damage to healthy tissue that accompanies conventional chemotherapy cancer treatments.
[...]

00:10:21 11:56:56 "I used chemotherapy"
00:10:26 11:57:01 "I really, I really think this is, I think this is the final one.''
00:10:29 11:57:04 clicks on "Save Query" button
enters name "final", query reads:

#uw100 (cancer #syn(research testing evaluation))
#syn (cancer leukemia)
#uw50 (cancer #syn (drug chemotherapy treatment))

RF: #SUM(#UW100(cancer #SYN(research test evalu)) #SYN(cancer leukemia) #UW50(cancer #SYN(drug chemotherap treatment)))

00:10:36 11:57:11 Clicks on "OK"
00:10:40 11:57:15 "[inaudible] in my particular performance"[laughs]

Interactive Exploration as a Formal Text Retrieval Method: How Well can Interactivity Compensate for Unsophisticated Retrieval Algorithms

Nipon Charoenkitkarn, Mark Chignell, and Gene Golovchinsky
Department of Industrial Engineering
University of Toronto

Introduction

Our goal in participating in TREC-3 was to see if it is possible to achieve adequate performance using relatively simple retrieval techniques, coupled with the type of interactivity at the user interface that is typical of a browsing system.

As the results reported in this paper show, our system was able to perform large scale text retrieval on routing queries, albeit not at a level to challenge well established text retrieval systems. However, since ST-PatTREC can handle both browsing and querying styles of interaction in a single environment, and since ST-PatTREC did reasonably well in comparison with other systems in category B of the competition, the present findings demonstrate that it should be possible to develop general-purpose information exploration systems that allow for both browsing and querying styles of interaction. The results that we obtained provide a baseline for what can be done when a browsing style of query formulation is combined with simple text retrieval methods. In addition to discussing the methods used and results in TREC-3, future plans are outlined for using visualizing and other styles of browsing in information exploration.

Information Exploration (Browsing AND Querying)

Current methods of accessing information tend to fall into two separate categories, depending on whether a browsing or querying style of interaction is used. In browsing, the goal is to navigate through a collection of documents or information nodes, looking for interesting or relevant information.

In contrast to the point and click selection that typically occurs in browsing, querying requires the user or system to specify a search goal that can be matched in some way against the database, in order to identify a set of relevant documents. One of the characteristics of querying situations is that it is much easier to define criteria of success. For instance, if one can measure the relevance of documents in some way, then standard measures such as precision and recall can be assessed.

The approach that we used in TREC-3 grew out of our earlier research on information exploration. Waterworth and Chignell (1991) identified three dimensions of information exploration. The first dimension of information exploration (structural responsibility) concerns who is responsible for searching and who must consequently be concerned with structure.

The second dimension of information seeking behaviour (target orientation) arose from the distinction between browsing and querying. According to Waterworth and Chignell, "Browsing is distinguished from querying by the absence of a definite target in the mind of the user." Thus the distinction between browsing and querying is not determined by the actions of the user, or by the configuration of the system, but by the cognitive state of the

user. One hypothesis stated in this early conceptualization of the information exploration task is that there is a continuum of user behaviours varying between querying and browsing that is characterised by the level of specificity of the user's information seeking goals. In our recent research we have been studying the degree to which users can readily mix browsing and searching styles of information exploration when given appropriate tools. The ST-PatTREC system used in TREC-3 allowed users to intermix browsing and querying styles of interaction.

A third dimension of information seeking behaviour can be identified based on the method of interaction used in the interface to the information system. Descriptive interfaces have generally been associated with querying behaviour, and referential (point and click) interfaces have generally been associated with hypertext browsing, but there is no intrinsic correlation between interaction method, target orientation, and structural responsibility. Thus the three-dimensional description of information seeking activity can be represented as in Figure 1.

The Interactive Querying Approach

One of the points made by Waterworth and Chignell (1991) is that the space of possible information exploration systems has not been fully explored. Most existing systems tend to fall into two general categories, text retrieval (i.e., systems that focus on command-based querying) and hypertext (systems that emphasize referential, or point and click, interaction within a navigational browsing framework).

Information retrieval and hypertext systems as currently envisioned are only two extreme cases within a family of possible information exploration systems. Our research is looking at the properties of intermediate members of that family that combine the functionalities of browsing and querying. One way of making querying more like browsing is to increase the interactivity of querying. If sufficiently rapid and incremental feedback is provided, there is very little cost to trying out a large number of different queries. The result is a more exploratory style of interaction that starts to approximate the interactivity of hypertext. The analogy with hypertext can then be reinforced by allowing queries to be marked up directly on text. Hits then appear as menus of hypertext links from the marked up concept specified by the graphical query to corresponding concepts in other documents.

User centeredness of hypertext systems can also be enhanced. Hypertext can be envisioned as a special kind of retrieval process where the goal is to provide the user with a link to the article or node that is most relevant given the context and status of the current transaction. A fundamental problem in manually authored (static) hypertext is that the links that are likely to be relevant in a particular context must be predicted in advance. The user's range of action is often limited to choosing among the available links at each step in the browsing process.

One way to increase user centeredness of hypertext is to give users more control over selection, as well as creation of links. The result would then be a form of dynamic hypertext in which the user expresses a concept of interest and the system identifies the nodes that address or relate to that concept. In this approach, link authoring is done at run-time, on the basis of selected concepts. Clearly, this type of user centeredness places considerable responsibility on the user to formulate and express concepts that are used as the basis for creating links. However, with appropriate support tools and user interfaces, such reader-based (dynamic) linking has considerable potential for enriching and customizing the information exploration task.

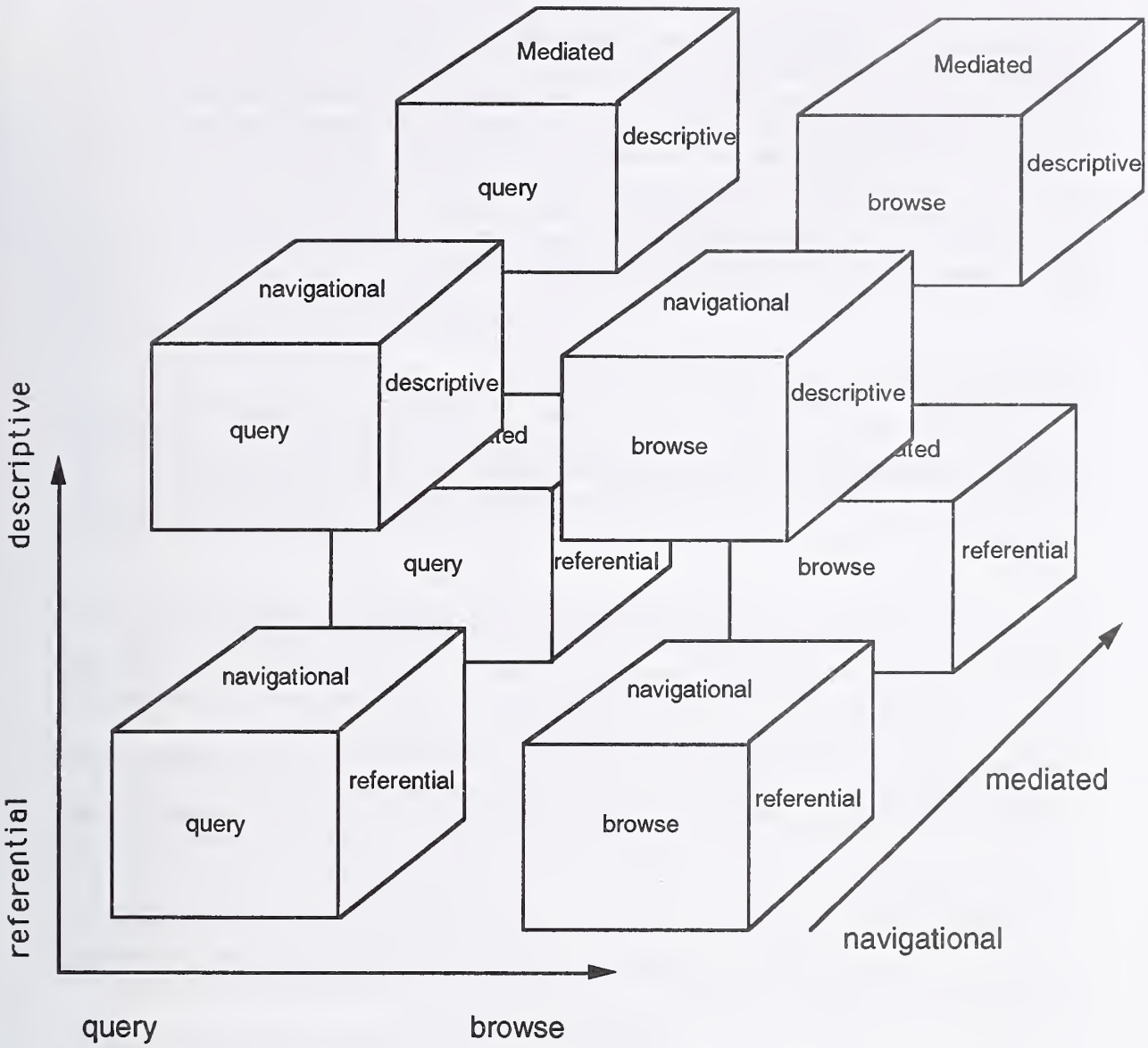


Figure 1. A Model of Information Exploration

Different types of feedback may be relevant at different stages of search or browsing. In the early stages of exploring a topic, users may not fully understand what they are interested in, or what terminology is used in the database. Thus the concept of a search lifecycle may be appropriate, with statistical information about co-occurrence of terms being more useful input from the user early in the search, and relevance feedback (based on accumulated evidence about topics of interest to the user) being more useful for topic reformulation later on in the search.

Some forms of dynamic feedback may approximate forms of authored information that have proved to be useful in books and other information retrieval structures and systems. For instance, feedback concerning term co-occurrence information provides some of the functionality of a hierarchical index. The first term acts to specify an entry in the index. The coordinate terms for that first term serve as sub-topics listed under the entry. One can then

explore further levels of the index hierarchy by looking for the coordinate terms of the first and second terms joined together. For instance one might have "transport" as the first term. A coordinate term might then be "roads." This would be equivalent to a hierarchical index where *road transport* was a sub-entry of *transport.* A coordinate term of "roads" and "transport" jointly might then be "trucking," corresponding to a sub-sub-entry under *transport.* Not many books include more than two levels of indexing because of the effort required to produce highly nested indices. The use of coordinate terms or more sophisticated techniques for assessing relationship and cohesion can simulate some of the functionality of a hierarchical index without authoring (manual indexing) effort. This process capitalizes on hierarchical indexing implicit in the text.

Support Tools for the User: ST-PatTREC

ST-PatTREC[1] was developed to implement a text browsing system capable of handling moderately large (hundreds of megabytes) databases and providing enhanced feedback to the user. ST-PatTREC uses a graphical notation developed by the second author (Golovchinsky, 1993) for specifying queries against databases capable of supporting them. It maintains a history of user requests and provides a workspace for storing components of queries.

The number of nodes returned can be controlled in ST-PatTREC by reducing the scope of the proximity operator (represented by the AND operator) until the desired number of nodes is displayed. For example, assume a user set a target of 10-15 nodes. A query with proximity of 500 characters may retrieve 100 nodes, but when the proximity criterion is reduced to 150 characters, a more manageable set of about 10 nodes might be retrieved. Furthermore, this set would likely be more relevant, since the terms would tend to be more closely related in the text. Giving users dynamic control over the volume of data should encourage exploration by reducing some of the uncertainty about the effects of commands.

IR Algorithms

A detailed description of the ST-PatTREC user interface and functionality is provided in the appendix to this paper. In this section we will briefly review the IR algorithms that ST-PatTREC uses, focusing on query execution and ranking.

ST-PatTREC uses exact matching of non-negated Boolean queries expressed in disjunctive normal form. Queries in ST-PatTREC can be entered by directly marking up text (drawing lines between selected words to indicate the AND operator). Non-negated disjunctive normal form Boolean command strings, e.g., (cancer AND drug) OR (cancer AND therapy) can be created by building AND clusters of terms, where each AND cluster contains a collection of words linked by lines (as explained by Golovchinsky and Chignell, 1993). The AND operator was actually executed as a proximity operator in ST-PatTREC. In the default condition, the AND condition was true if the words covered by the AND appeared within 500 characters of each other. In addition though, the user was provided with a proximity slider so that the scope of the proximity could be varied directly by the user.

---

[1]ST represents the Smalltalk interface, Pat is the full-text search engine from OpenText Foundation, and TREC is the Text REtrieval Conference sponsored by NIST for which this software was developed.

A very simple form of document ranking was used for each final query. The first challenge was to obtain a set of 1000 documents, given that many of our queries would extract considerably less than a 1000 documents (sometimes less than 100 documents). Once the set of 1000 documents (or as close as we could get to 1000 documents) was obtained, the documents were ranked according to the number of times that the query terms appeared in their text, normalized by the length of each document. This ranking was based on exact matching of terms (i.e., no truncation was used).

The selection of the set of 1000 documents was done manually using the following algorithm that could have been implemented in software had more time been available. This algorithm was generally adhered to, but other methods were also tried if necessary to produce the right number of documents.

> **Step 1.** Use the exact final query with the exact proximity that was specified. This provided the initial set of documents.
>
> **Step 2.** Use the exact final query with the proximity expanded so that it equalled the length of the document. That is, the AND became an actual Boolean AND rather than a proximity operator.
>
> **Step 3.** Use truncation AND/OR loosening of the query. For instance, loosening might be done by removing a term from an AND cluster.

The procedure for expanding the hit set to the necessary 1000 documents can be illustrated with an example. The original routing query that we identified for one topic was:

> (Japan AND market AND access) OR (Unfair AND Talk) with proximity = 200

The first expansion of this query involved removing the proximity restriction so that the scope of the AND operator was within entire documents. The query was then loosened by removing the word "market" from the first AND cluster so that it became (Japan AND Access). At this stage we still did not have the required number of documents. The final expansion in this case involved conflating two terms across the AND clusters to obtain (Japan AND unfair), which yielded the desired number of documents.

TREC-3 Evaluation Strategy for ST-PatTREC

Queries were constructed based on the work of two searchers. The searchers carried out query formulation using the ST-PatTREC system. They worked independently and then reviewed their results to select a routing query based on the precision and recall measures obtained on the training data. In our study, the Wall Street Journal data was the only data used in training.

Given that we used relatively simple text retrieval algorithms[2] , how could we expect to obtain reasonable results in comparison with much more sophisticated retrieval algorithms? Our hope was that our searchers would be able to come up with better search terms based on extensive browsing of the training data. To the extent that the TREC routing topics already included rich statements of the search concept, along with key terms, the task of compensating for rudimentary IR algorithms through better selection of terms based on extensive browsing was made more difficult.

---

[2] The version of ST-PatTREC used in TREC-3 did not even have a truncation operator, which has since been added to the system.

We tried to increase the likelihood of including meaningful terms in the query by cross-validating queries and terms across two separate halves of the training database. Each half of the database was approximately 250 MB in size. We carried out browsing and query formulation on one half of the database and then ran the resulting query on the other half of the database to see how much "shrinkage" of the precision and recall scores occurred. We then carried out query formulation on the other half of the database. The routing query was chosen that seemed to maximize precision and recall across both halves of the database. An example of the variability that can occur in the split half comparisons is shown in Table 1. Numbers in parentheses show the corresponding values of precision and recall that occurred in the second half of the database.

In general, Table 1 shows a fair amount of consistency between the two halves of the database in terms of recall and precision associated with the exact final query. The large differences that occurred (e.g., the increase in recall from .67 to .87 for the first query) are generally due to changing the proximity, or seem to be associated with a tradeoff between recall and precision (e.g., the query in the second row of the table has a higher recall in the second half of the database, but a lower precision). Based on generally strong correlations on recall and precision between the two halves of the training database, we expected that better recall and precision on the training database would be predictive of better recall and precision on the test database.

Query	Recall	Precision	Proximity
(drug AND cancer)	.67 (.87)	.16 (.13)	128 (275)
(drug AND cancer OR cancer AND research)	.69 (.75)	.15 (.12)	85 (85)
(drug AND cancer OR cancer AND research OR cancer AND development)	.72 (.78)	.14 (.10)	85 (110)
(drug AND cancer OR cancer AND therapy)	.69 (.78)	.14 (.14)	128 (110)

Table 1. Performance of Query for Topic 122 across Two Halves of the Training Database.

Results

ST-PatTREC did reasonably well in comparison with the other two systems in category B of the competition. Table 2 summarizes the results that we obtained across the 50 topics. In addition to the topic number (top), the table shows the elapsed time for each query (time), the number of iterations (iter), the precision of the query on the training data (tpre), the recall of the query on the training data (trec), the corresponding recall (erec) and precision (epre) scores for the test data, the average precision for the test data (avgpre), the number of relevant documents (numr) that existed in the test database for that topic (as assessed by the TREC organizers), the number of terms in the final query (numt), and the number of OR operators in the final query (numor). The ranking (rank) of our system on a three point scale (best, median, worst) based on feedback provided to us about how our system compared with the other systems in our section of the competition, is also shown, as are the ratings that the searchers made concerning their familiarity with the general search topics (exp). All precision and recall scores shown in Table 2 except average precision scores (avgpre) are based on exact final queries with exact proximities. The ranks are based on those of relevance retrieved at top 100 articles. If our rank was tied with another system, we used the average rank (e.g. tied for first = 1.5).

top	time	iter	tpre	trec	epre	erec	numr	numt	numor	rank	avgpre	exp
101	33	14	.13	.67	.15	.22	9	8	2	1.50	.2973	8
102	22	10	.22	1.00	.17	.20	5	7	2	1.00	.2246	8
103	17	8	.00	.00	.02	.05	20	7	2	3.00	.1025	5
104	15	6	.00	NA	.00	NA	0	6	2	2.00	.0000	5
105	36	10	.14	1.00	.00	NA	0	8	2	2.00	.0000	5
106	60	10	.07	.45	.24	.42	12	5	1	2.00	.2063	5
107	40	6	.42	.25	.43	.12	24	4	1	2.00	.1629	5
108	12	18	.26	.23	.28	.07	67	5	1	3.00	.1377	5
109	39	7	.49	.92	.97	.82	132	5	4	1.00	.8952	8
110	29	10	.38	.71	.46	.52	102	4	1	3.00	.4595	7
111	36	15	.39	.49	.43	.30	231	11	5	2.00	.4488	7
112	18	4	.27	.36	.33	.22	72	6	2	2.00	.3161	5
113	51	7	.10	.71	.09	.57	21	8	3	1.50	.1386	8
114	42	12	.07	.29	.10	.03	40	6	1	3.00	.1563	7
115	23	7	.09	.73	.18	.64	25	7	4	2.00	.1548	5
116	33	11	.05	.44	.00	NA	0	5	2	2.00	.0000	5
117	20	9	.13	.39	.10	.20	15	4	1	2.00	.1762	6
118	57	10	.12	.34	.23	.16	183	6	2	3.00	.1652	7
119	29	10	.16	.41	.23	.19	139	6	2	2.50	.2159	7
120	20	12	.00	.00	.25	.01	89	4	1	1.50	.1016	5
121	45	6	.00	.00	.08	.03	118	7	2	2.00	.1157	5
122	20	10	.15	.80	.17	.80	30	4	1	1.00	.2204	5
123	18	7	.16	.49	.40	.54	125	6	2	2.00	.3976	5
124	22	6	.14	.78	.05	.67	21	8	3	2.00	.1037	5
125	17	5	.21	.64	.46	.61	101	7	3	1.00	.5791	5
126	59	13	.37	.45	.30	.27	67	6	2	2.00	.2621	5
127	20	7	.04	.56	.08	.62	24	6	2	2.00	.1857	7
128	39	12	.23	.60	.25	.22	27	8	3	1.00	.2913	5
129	14	7	.06	.86	.25	.01	36	6	2	2.00	.1891	7
130	28	8	.70	.36	.38	.17	59	3	1	2.00	.2295	7
131	54	13	.12	.42	.00	.00	1	3	0	2.00	.0038	4
132	21	6	.28	.72	.18	.71	28	8	3	2.00	.3737	4
133	21	1	.47	1.00	.20	.91	11	1	0	1.50	.5123	8
134	20	1	.89	.94	.50	.83	6	1	0	1.50	.7553	5
135	27	7	.63	.84	.53	.66	76	8	3	2.00	.4793	5
136	60	3	.59	.80	.71	.45	11	8	3	2.00	.5451	6
137	22	12	.39	.52	.22	.23	43	5	2	3.00	.1972	6
138	49	10	.06	.75	.08	1.00	16	4	1	1.50	.1559	7
139	16	6	.NA	.00	.67	.20	10	5	1	2.00	.2198	7
140	18	5	.33	.50	.00	.00	7	2	0	1.00	.0618	7
141	60	18	.00	.00	NA	.00	1	3	0	1.00	.0106	6
142	37	7	.42	.44	.86	.29	123	7	2	1.00	.4965	7
143	40	9	.12	.47	.14	.16	19	6	2	2.00	.2143	6
144	12	8	.00	.00	.00	NA	0	4	1	2.00	.0000	7
145	37	9	.46	.51	.41	.52	21	4	2	1.00	.7523	6
146	16	5	.00	NA	.11	.67	3	4	1	1.50	.3269	7
147	55	10	.21	.43	.40	.21	98	11	5	3.00	.2087	6
148	20	7	.56	.64	.54	.94	78	4	3	2.00	.7951	4
149	55	7	.46	.28	.33	.05	58	4	1	1.00	.1511	5
150	48	10	.38	.64	.35	.55	155	4	2	1.00	.4903	6

Table 2. Evaluative and Descriptive Data for the ST-PatTREC Searches

Table 3 shows the results obtained by our system (marked with asterisks) compared with other systems in our section of category B, as provided to us by the TREC organizers. While it is difficult to make meaningful comparisons across different databases and topics, our average precision aggregated across all the topics was .27, within the range obtained by the Okapi system in TREC-2 (Robertson et al. 1993). The Okapi average precisions varied between a high of .36 and a low of .142 (aggregating across all the different topics with different weightings and algorithms). Our average precision results also appeared to be competitive with results obtained in TREC-2 using combination of evidence (Belkin et al. 1993).

We also looked at how well recall and precision on the training set predicted recall and precision on the test data. We found evidence of predictive relationships on both recall and precision scores (calculated from documents retrieved by exact final queries with exact proximities). That is, if a query achieves a high recall score or high precision score on the training data it seems to do well on the test data as well. Table 4 shows the matrix of intercorrelations for the data shown in Table 2. The correlation between the precision measures on the training data versus the corresponding measures on the test data, and the correlation between recall measures for training and test data were both .70

We then looked to see which other factors would predict when our system would perform well. We looked at the relationship between the ranking of our system (rounded up to 1, 2, or 3) across the 50 topics, and a variety of predictive measures, including the training precision and recall, the elapsed time for the query formulation process, the number of iterations used, the number of terms in the final routing query, etc. Separate analyses of variance were carried out for each of these predictors as the dependent variable, with the independent variable being the ranking of the topic. None of these ANOVAs were significant at the .05 level. However, the number of terms in the final query ($F[2, 47]=2.42$) and the number of iterations ($F[2, 47]=2.70$) were both borderline significant ($p <.10$). Figures 2 and 3 show the corresponding error bar charts. It can be seen in Figure 3 that the possible significant difference for iterations is due to the fact that more iterations tend to be associated with relatively poor performance of the routing query. This seems reasonable to us, since a large number of iterations indicated that the searchers were having trouble in identifying a suitable query. Similarly a relatively low (less than 5) terms in the query tended to be associated with better performance. This suggests that our system worked best when a fairly simple query could be expressed with a few diagnostic terms.

Relatively few of the correlations shown in Table 4 are significant. Not surprisingly, perhaps, there was a high ($r=.83$) correlation between the number of terms in the final routing query and the number of OR operators used. However, the correlation between number of iterations and time taken in performing the query was low (about .3). The number of iterations was inversely related to recall on both the training and test data, confirming the impression that a large number of iterations is generally a sign of problems in query formulation.

There was also a significant ($r=.45$) correlation between the number of relevant documents in the test database and the number of ORs in the final routing query. This seems reasonable, since a broad topic might be expected to cover a number of disjunctive concepts.

We also looked at what sort of topics our method did better or worse in. The two searchers each independently rated their familiarity with the general topics used in the searches (e.g., science and technology, finance, international politics, etc.) on a 5-point scale. We added the two sets of ratings to obtain a 10-point rating that represented the joint familiarity of both searchers with the general topics. We then correlated the expertise ratings of

Topic	Rel.	Relevant Retr. @100 Best	Median	Worst	Relevant Retr. @1000 Best	Median	Worst	Average Precision Best	Median	Worst
101	9	7*	7	4	9*	9	4	0.3120	0.2973*	0.0980
102	5	5*	4	4	5*	5	4	0.5520	0.5187	0.2246*
103	20	8	7	3*	18*	18	11	0.1194	0.1172	0.1025*
104	0	0*	0	0	0*	0	0	0.0000*	0.0000	0.0000
105	0	0*	0	0	0*	0	0	0.0000*	0.0000	0.0000
106	12	9	8*	1	11*	10	2	0.2453	0.2063*	0.0066
107	24	15	14*	10	21	17*	17	0.3117	0.1629*	0.1615
108	67	29	27	18*	61	59*	43	0.3138	0.2091	0.1377*
109	132	94*	89	13	129	127*	41	0.8952*	0.8804	0.0668
110	102	53	52	50*	102*	102	69	0.5480	0.4595*	0.4175
111	231	78	59*	40	222*	221	123	0.4488*	0.4140	0.4130
112	72	37	36*	6	66*	65	21	0.3639	0.3161*	0.0368
113	21	14*	14	8	21*	21	9	0.2068	0.1444	0.1386*
114	40	24	24	14*	40*	38	27	0.4086	0.3713	0.1563*
115	25	18	16*	8	24*	23	9	0.2662	0.1548*	0.0998
116	0	0*	0	0	0*	0	0	0.0000*	0.0000	0.0000
117	15	14	10*	8	15	14*	8	0.5962	0.2636	0.1762*
118	183	67	59	32*	154	115*	106	0.5081	0.4247	0.1652*
119	139	36	26*	26	124*	108	44	0.2827	0.2159*	0.1366
120	89	13*	13	9	84*	72	13	0.1016*	0.0926	0.0276
121	118	52	14*	2	114	107*	39	0.4721	0.1157*	0.0190
122	30	20*	13	12	26*	18	12	0.2690	0.2316	0.2204*
123	125	58	49*	24	121*	105	49	0.4521	0.3976*	0.1002
124	21	12	11*	4	20*	14	8	0.1532	0.1037*	0.0387
125	101	61*	46	42	101	100*	44	0.5791*	0.4037	0.3042
126	67	43	32*	25	55	49*	31	0.4598	0.3337	0.2621*
127	24	22	15*	12	24	24*	12	0.3712	0.2350	0.1857*
128	27	22*	10	5	25*	18	10	0.2913*	0.1724	0.0247
129	36	20	18*	13	33	31*	16	0.2750	0.1891*	0.1387
130	59	30	23*	19	59	52*	23	0.3379	0.2360	0.2295*
131	1	0*	0	0	1*	1	1	0.0072	0.0057	0.0038*
132	28	23	22*	15	28	27*	15	0.5139	0.4022	0.3737*
133	11	10*	10	5	10*	10	5	0.7622	0.5123*	0.3690
134	6	6*	6	1	6*	6	1	0.7553*	0.4451	0.0833
135	76	53	50*	30	76	63*	37	0.6848	0.4793*	0.2947
136	11	10	9*	4	11	10*	6	0.5451*	0.4997	0.0651
137	43	30	20	18*	41	38*	22	0.6102	0.2496	0.1972*
138	16	12*	12	11	16*	16	12	0.2786	0.2146	0.1559*
139	10	8	5*	4	8	5*	4	0.2198*	0.1640	0.1224
140	7	5*	4	4	7*	5	4	0.3813	0.3805	0.0618*
141	1	1*	0	0	1*	1	0	0.0106*	0.0064	0.0000
142	123	69*	42	31	99*	89	62	0.4965*	0.3113	0.1784
143	19	12	10*	5	18*	17	8	0.2573	0.2143*	0.0471
144	0	0*	0	0	0*	0	0	0.0000*	0.0000	0.0000
145	21	20*	12	2	21*	15	6	0.7523*	0.2259	0.0116
146	3	3*	3	1	3*	3	1	0.4019	0.3333	0.3269*
147	98	38	37	32*	77	66	59*	0.2889	0.2661	0.2087*
148	78	71	70*	44	77*	77	45	0.7951*	0.7870	0.5345
149	58	17*	16	10	40*	39	12	0.1511*	0.1141	0.0563
150	155	56*	53	19	140*	111	56	0.4903*	0.3862	0.1160

Table 3. TREC-3's Routing Results for Category B.

Correlation Coefficients (Coefficient/(Cases) / 2-tailed Significance) "." is printed if a coefficient cannot be computed

	TIME	ITER	TPRE	TREC	EPRE	EREC	NUMR	NUMT	NUMOR	RANK	AVGPRE	EXP
TIME	1.0000	.3023	.0355	-.0760	.1266	-.1770	.1564	.1567	.0934	-.0524	-.0512	.0152
	(50)	(50)	(49)	(48)	(49)	(46)	(50)	(50)	(50)	(50)	(50)	(50)
	P= .	P= .033	P= .809	P= .607	P= .386	P= .239	P= .278	P= .277	P= .519	P= .718	P= .724	P= .917
ITER	.3023	1.0000	-.3251	-.3225	-.2044	-.4384	.1612	.1641	.0035	.1856	-.3509	-.0135
	(50)	(50)	(49)	(48)	(49)	(46)	(50)	(50)	(50)	(50)	(50)	(50)
	P= .033	P= .	P= .023	P= .025	P= .159	P= .002	P= .263	P= .255	P= .981	P= .197	P= .012	P= .926
TPRE	.0355	-.3251	1.0000	.3858	.6963	.3029	.1933	-.2065	.0498	-.1504	.6984	-.0153
	(49)	(49)	(49)	(47)	(48)	(45)	(49)	(49)	(49)	(49)	(49)	(49)
	P= .809	P= .023	P= .	P= .007	P= .000	P= .043	P= .183	P= .155	P= .734	P= .302	P= .000	P= .917
TREC	-.0760	-.3225	.3858	1.0000	.1333	.6995	-.1021	.1008	.2609	-.2690	.4551	.1607
	(48)	(48)	(47)	(48)	(47)	(45)	(48)	(48)	(48)	(48)	(48)	(48)
	P= .607	P= .025	P= .007	P= .	P= .372	P= .000	P= .490	P= .495	P= .073	P= .065	P= .001	P= .275
EPRE	.1266	-.2044	.6963	.1333	1.0000	.2250	.4520	.0540	.2957	-.1805	.7620	.1059
	(49)	(49)	(48)	(47)	(49)	(45)	(49)	(49)	(49)	(49)	(49)	(49)
	P= .386	P= .159	P= .000	P= .372	P= .	P= .137	P= .001	P= .713	P= .039	P= .215	P= .000	P= .469
EREC	-.1770	-.4384	.3029	.6995	.2250	1.0000	-.0353	-.0994	.2087	-.2539	.6165	.0057
	(46)	(46)	(45)	(45)	(45)	(46)	(46)	(46)	(46)	(46)	(46)	(46)
	P= .239	P= .002	P= .043	P= .000	P= .137	P= .	P= .816	P= .511	P= .164	P= .089	P= .000	P= .970
NUMR	.1564	.1612	.1933	-.1021	.4520	-.0353	1.0000	.3071	.4471	.1467	.3504	.0905
	(50)	(50)	(49)	(48)	(49)	(46)	(50)	(50)	(50)	(50)	(50)	(50)
	P= .278	P= .263	P= .183	P= .490	P= .001	P= .816	P= .	P= .030	P= .001	P= .309	P= .013	P= .532
NUMT	.1567	.1641	-.2065	.1008	.0540	-.0994	.3071	1.0000	.8307	.2449	-.0603	-.0233
	(50)	(50)	(49)	(48)	(49)	(46)	(50)	(50)	(50)	(50)	(50)	(50)
	P= .277	P= .255	P= .155	P= .495	P= .713	P= .511	P= .030	P= .	P= .000	P= .087	P= .677	P= .873
NUMOR	.0934	.0035	.0498	.2609	.2957	.2087	.4471	.8307	1.0000	.1163	.2885	-.0336
	(50)	(50)	(49)	(48)	(49)	(46)	(50)	(50)	(50)	(50)	(50)	(50)
	P= .519	P= .981	P= .734	P= .073	P= .039	P= .164	P= .001	P= .000	P= .	P= .421	P= .042	P= .817
RANK	-.0524	.1856	-.1504	-.2690	-.1805	-.2539	.1467	.2449	.1163	1.0000	-.2791	-.1068
	(50)	(50)	(49)	(48)	(49)	(46)	(50)	(50)	(50)	(50)	(50)	(50)
	P= .718	P= .197	P= .302	P= .065	P= .215	P= .089	P= .309	P= .087	P= .421	P= .	P= .050	P= .461
AVGPRE	-.0512	-.3509	.6984	.4551	.7620	.6165	.3504	-.0603	.2885	-.2791	1.0000	.0893
	(50)	(50)	(49)	(48)	(49)	(46)	(50)	(50)	(50)	(50)	(50)	(50)
	P= .724	P= .012	P= .000	P= .001	P= .000	P= .000	P= .013	P= .677	P= .042	P= .050	P= .	P= .537
EXP	.0152	-.0135	-.0153	.1607	.1059	.0057	.0905	-.0233	-.0336	-.1068	.0893	1.0000
	(50)	(50)	(49)	(48)	(49)	(46)	(50)	(50)	(50)	(50)	(50)	(50)
	P= .917	P= .926	P= .917	P= .275	P= .469	P= .970	P= .532	P= .873	P= .817	P= .461	P= .537	P= .

Table 4. Matrix of Intercorrelations of Data in Table 2.

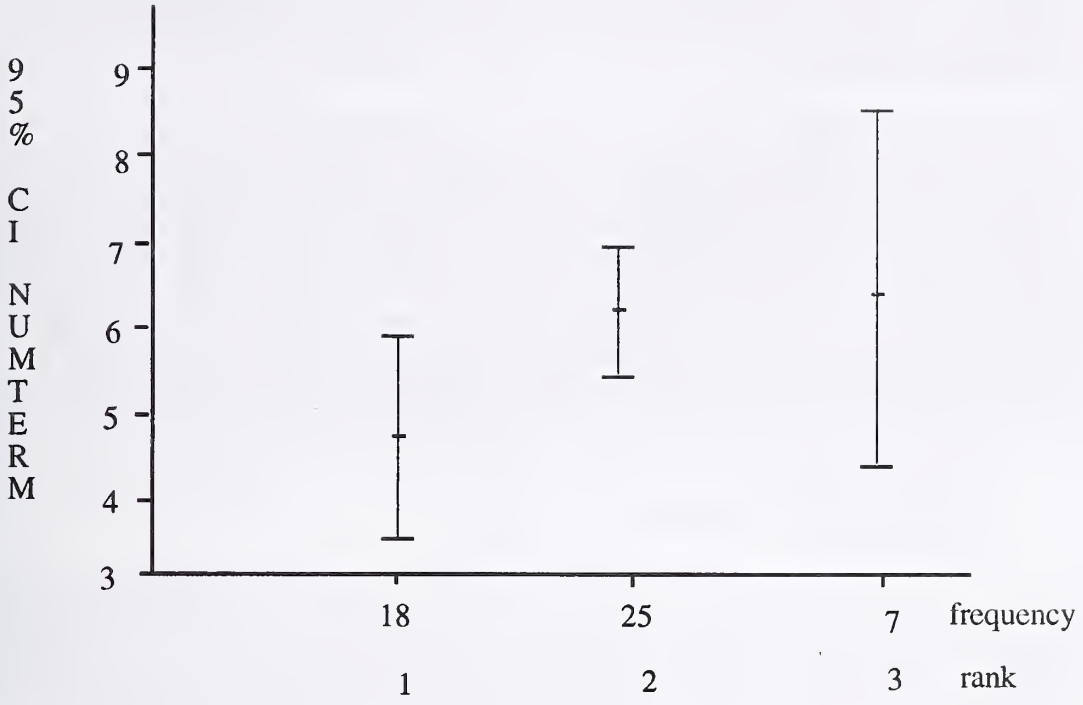


Figure 2. Effect of Number of Terms in Final Query on Performance Rankings.

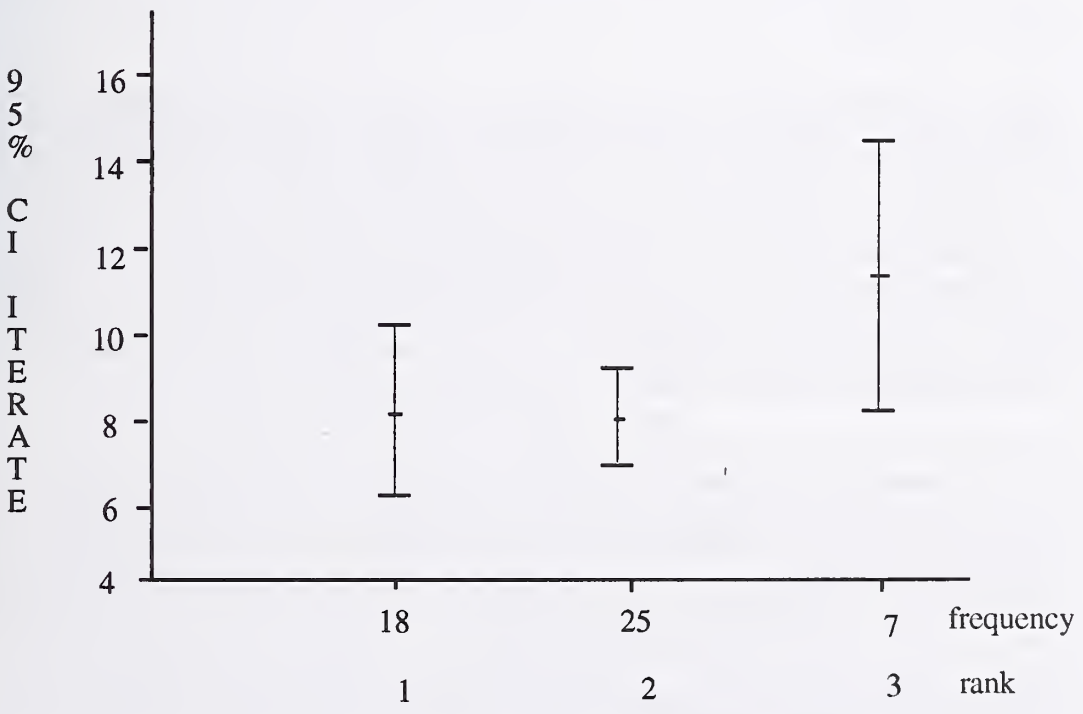


Figure 3. Effect of Number of Iterations on Performance Rankings.

familiarity with each topic with the corresponding measures of precision and recall on the training and test data, respectively. None of these correlations were significantly greater than zero ($r<.10$ in every case). Thus we have no evidence that domain expertise was a factor in the searches.

The final analysis that we carried out looked at what predicted performance of the routing queries in terms of average precision. A stepwise multiple regression was carried out using query formulation time, number of iterations, precision of the query for the training data, recall of the query for the training data, the number of relevant documents for the topic actually in the test database, and the number of terms used in the final query as predictors. Training precision was the strongest predictor of average precision of the routing query ($r=.54$). The other two predictors that entered the final regression equation were the number of relevant documents and training recall (multiple R for the three predictor equation was .79). None of the other predictors significant improved the three term prediction equation. The beta weights in the three term equation were as follows:

training precision	.54
number of relevant documents	.29
training recall	.29

The precision and recall of the exact final query on our training data were strongly related to precision on the test data. The other predictor of performance on the test data appeared to be number of iterations. A number of different attempts to form a search query is generally indicative of problems in query formulation that tend to show up in the eventual performance of the routing query. In the following section we will look at some of the lessons learned concerning the usability of the system.

Lessons Learned

The two searchers spent many hours working on the 50 topics to form the final routing queries. Much of this time was wasted due to deficiencies in the ST-PatTREC system. One problem was that the graphical queries for complex disjunctions were sometimes parsed incorrectly, requiring the users to try out different input strategies in order to get the query they wanted. This problem has since been fixed by allowing users to type in Boolean queries directly. We also found that the demands of the TREC evaluation on our system were greater than expected. As a research prototype, the system seemed to work quite well, but in the face of many hours of focused searches, the system would often crash, adding to the time and effort required. Thus there has been some effort to improve the reliability of the system and make it more suitable for large scale searches in the future.

One of the features that we missed the most was truncation during the query formulation. We used truncation during the query expansion phase, but in the interactive querying that preceded the generation of the final routing query truncation was not available. Truncation was one of the features that did not seem to be a "natural" part of the browsing interface, but turned out to be important for querying styles of search. Although our original philosophy had been to construct queries by directly marking up the text in a point and click style of interaction, we soon found that direct input of queries as text was more convenient in constructing complex queries. It was also easier to combine older queries into new queries using text entry in a workspace than to try and modify the markup once a certain number of terms were involved in the query.

The online dictionary didn't seem to add any value during the query generation process and was not used. Similarly, the online thesaurus (WordNet) was not used. We found that the

material in the online dictionary and thesaurus was too general, so that they did not add value for specific queries (at least queries on topics relevant to the Wall Street Journal database). In contrast, the coordinate terms turned out to be very useful. Knowing the coordinate terms made it much easier to identify terms that could be ANDed to the current term.

The searchers found that interactive feedback of recall and precision was both useful and motivating. We found that the ability to select terms directly from document text was useful, because the searchers could follow the terminology actually used in the database. For the TREC search tasks, the best working style seemed to be selection of the terms in the text and graphical markup for simple queries followed by textual commands to combine the simple queries into complex query. In later versions of the system we have since added this capability and found it to be very useful. The TREC experience made it clear to us that trying to build complex queries was just too difficult with the graphical querying methods that we had originally developed. The searchers also found that the documents judged to be relevant by experts were very useful and they consulted extensively in looking for appropriate search terms.

Future Extensions

Our experience in using ST-PatTREC in the TREC-3 experiment showed that the ability to form complex queries is fairly important in achieving good performance under the TREC-3 rules. The detailed search topics provided the relatively large number of documents that were relevant for queries (typically over 100), and the requirement to create a routing query tended to de-emphasize the role of browsing in the interaction in favor of query construction. We are currently looking at methods of reinstating the role of browsing, either within the TREC-3 rules, or else in terms of a modified approach that changes the task and evaluation structure so as to encourage more browsing.

One of the problems with large databases and topics that have large numbers of relevant items is that interactive browsing of the actual documents is too costly, in both time and effort. However, browsing of a summarized version of the document space might still be feasible. Two forms of summarization are the use of brief abstracts, and the use of visualizations of the concept space in which documents reside. Automatic abstracting is difficult, and articles such as those available in the TREC database tend not to have written abstracts. For some stories or articles, lead paragraphs might serve as abstracts or overall summaries, while for others, the simple heuristic of picking the first one or two paragraphs may not be adequate. Recently there have been proposals to use the cohesiveness of text (e.g., Halliday and Hasan, 1976) to develop paths or chains through the text. However, although promising, this research is still in the early stages.

In the near term concept visualization may be the most useful approach for text summarization. In one technique (Damashek, 1994), inter-document similarity is assessed based on frequencies of n-grams in documents. The advantage of using N-grams is that they are less affected by incorrect spellings (Cavnar, 1993). Using one of a number of scaling techniques (e.g., singular value decomposition, or multidimensional scaling) the documents can then be located in a low dimensional space and clustered.

We carried out an informal experiment at the TREC-3 conference to see how well the Acquaintance system (Damashek, 1994) would work as visualization tool for handling TREC-3 topics. We began by using ST-PatTREC to form a query. We used the final routing query that we had developed for topic 122. This query was (cancer AND drug) OR (cancer AND research). This query yielded about 15% precision and 74% recall. We then

took the 268 documents that were retrieved by this query for half of the training (Wall Street Journal) database and passed them to the Acquaintance system where they were visualized using the N-gram similarity scaling and clustering method. The document clusters were then examined for relevance to the topic and non-relevant items were pruned from the visualization. This pruning was done fairly quickly, taking less than five minutes. The resulting set contained 19 documents, of which 9 or roughly 47% were relevant. The visualization method was successful in improving the precision, something that had been a problem for the ST-PatTREC system in the TREC-3 experiment, but at the expense of recall (since only 9 of the 54 relevant documents or approximately 17% were relevant). While this recall is low, it should be possible to explore the visualization further and find additional clusters of relevant documents. Thus one future direction is to use text browsing to develop a relatively large set of candidate documents with high recall and low precision, and then pass this to a visualization process to prune irrelevant document clusters and thereby improve precision. We could then characterize the steps in this overall process as:

1. Text browsing
2. Query formulation
3. Visual Browsing of Document Clusters

We will also be carrying out research that investigates the role of browsing and querying in information exploration more closely. We are currently developing a revised information exploration system called BrowsIR that incorporates some of the lessons learned from our participation in TREC-3 as well as from subsequent usability studies using ST-PatTREC. We will then carry out a study to see how people use a flexible information exploration system when faced with different types of tasks. In particular, we will look at how instructions that emphasize browsing or querying styles of interaction tend to affect retrieval performance using a flexible information exploration system. The experiment will be carried out using TREC databases and topics.

One further avenue that we will explore is the coordination of text retrieval on local and distributed databases. We are currently connecting our text browsing tool to the world wide web. We will then carry out TREC searches on both the TREC and world wide web data. Our goal will be to study the effectiveness of wide area network (WAN) text searching as a supplement to the information contained in the TREC databases.

Conclusions

Overall we were satisfied with the performance of our system in TREC-3. Given the changes that we have since made to the system based on our experience, we expect that its effectiveness should be improved now that truncation and textual entry methods to combine simple queries into complex queries have been added.

One important result is the consistency between the evaluation scores for the training data and the test data. Our system seemed to work best with short queries consisting of a few highly diagnostic terms. The extreme case of this was two topics where the query consisted of only one term ("Hubble" for one topic and "genome" for the other). It is not clear how much this effect was due to the fact that construction of complex queries was too difficult in the version of our system that we were using, and how much it is a general effect. One clear diagnostic result is that a large number of querying iterations was a sign of trouble in our system. The searchers wrote down the query and the recall and precision scores when they felt that they had an interesting result. We used the number of entries that they wrote in the search log as indicators of the occurrence of iterations in the query formulation. While the implications of this result are unclear, one interpretation is that there is a law of

diminishing returns in interactive querying, and that searches which do not yield useful results relatively quickly are unlikely to improve with significant extra effort. Certainly our experience as searchers was that some searches were difficult, while others seemed easy. Putting a lot of extra time and effort into the difficult searches didn't seem to improve the results much in most cases.

Acknowledgments

This research was supported by the Information Technology Research Center of Excellence of Ontario, the National Science and Engineering Research Council of Canada, and by the Canadian International Development Agency. We would also like to thank Professor Frank Tompa of the University of Waterloo, and Open Text Corporation, for making the Pat text retrieval engine available to us for our research.

References

Belkin, N.J., Kantor, P., Cool, C., and Quatrain, R. (1993). Combining Evidence for Information Retrieval. In D. Harman (Ed.), *Proceedings of the Second Text REtrieval Conference (TREC-2)*, NIST Special Publication 500-215, National Institute of Standards and Technology, Gaithersburg, Maryland.

Cavnar, W.B. (1994). N-Gram-Based Text Filtering for TREC-2. In D. Harman (Ed.), *Proceedings of the Second Text REtrieval Conference (TREC-2).* NIST Special Publication 500-215, National Institute of Standards and Technology, Gaithersburg, Maryland.

Damashek, M. (1994). Gauging similarity via N-Grams: Language-Independent Sorting, Categorization, and Retrieval of Text. Unpublished manuscript, U.S. Department of Defense, Ft. George Meade, Maryland.

Fawcett, H. (1989). *PAT 3.3 User's Guide.* , Waterloo, Ontario, Canada: UW Centre for the New Oxford English Dictionary.

Golovchinsky, G. (1993).*Queries-R-Links: Query-based browsing in a full-text retrieval system.",* MASc thesis, University of Toronto

Golovchinsky, G. and M.H. Chignell. (1993). Queries-R-Links: Graphical markup for text navigation. in *Proceedings of INTERCHI '93.* . Amsterdam, The Netherlands:

Halliday, M.A.K. and Hasan, R. (1976) Cohesion in English. London: Longman.

Shneiderman, B. (1987). *Designing the User Interface.* Reading, Massachusetts: Addison Wesley.

Robertson, S.E., Walker, S., Jones, S., Hancock-Beaulieu, M.M., and Gatford, M. (1993). Okapi at TREC-2. *Proceedings of the Second Text REtrieval Conference (TREC-2)*, NIST Special Publication 500-215, National Institute of Standards and Technology, Gaithersburg, Maryland.

Waterworth, J.A. and Chignell, M.H. (1991). A Model of Information Exploration. *Hypermedia, 3(1)*, 35-58.

Appendix. Interactive System Description

I. System Description

Figure 4 shows a "typical" screen from the system. The screen contains a number of regions or windows which allow the user to build queries and view the results of queries within a single environment. One of the features of this system is that it does not have different modes (e.g., forming queries vs. viewing documents).

The window on the lower left is the text window where documents are displayed. Queries can be constructed directly by clicking on words in the text to add them to the query. Dragging lines between words then connects them in the query with an AND operator.

Words can also be added to a query by typing them in the margin. For instance, in Figure 4 the word "degenerative" has been typed in the margin. The current query is displayed in the small gray field above the text window. In this case the current query is (research AND cancer OR degenerative). This display uses parentheses to disambiguate the scope of the AND and OR operators. The query displayed as text corresponds to the graphical query marked on the text of the document. Each of the terms in the query is shown within a box. The terms cancer and research are linked by a line, which indicates the AND operator. The fact that "degenerative" is not linked with a line to the other terms shows that it is ORed with them.

Above the query display area is a quite button, and open file button (this button wasn't used during the TREC-3 experiment), and a new query button, which gave the user a quick way to initialize the query (the alternative was to explicitly delete the terms in the current query using point and click selections). To the right of these buttons was a pull-down menu showing a list of the TREC search topic statements. The pull-down menu is currently showing topic 104 as being selected. The topic statement was normally used as the starting point for each search. The topic statement could be marked up like any other document.

The search controls at the top of the screen contained a proximity slider and a relevance judgment slider. The proximity slider was used to vary the proximity. In addition to moving the slider, the user could also type the new proximity value directly into the field above the slider (thus making it possible to have proximities of greater than 1000). In practice, the default proximity of 500 characters was used most of the time, with proximity mainly being adjusted to fine tune the query.

Output from the online dictionary is shown in the lower right part of the display in a window labeled "definition". In this case the dictionary reports that there is no definition for "degenerative". The lack of useful feedback from the online dictionary in this case was fairly representative of its effectiveness throughout the searches.

The coordinate terms were shown in one of the two empty windows in the lower right of the screen. Above them is a window that provided the feedback on recall and precision of the current query. This information was updated every time a hit was selected. The window above this allowed users to access the online thesaurus. The information from the thesaurus was then displayed in the definition window (the same place that displayed the online dictionary information).

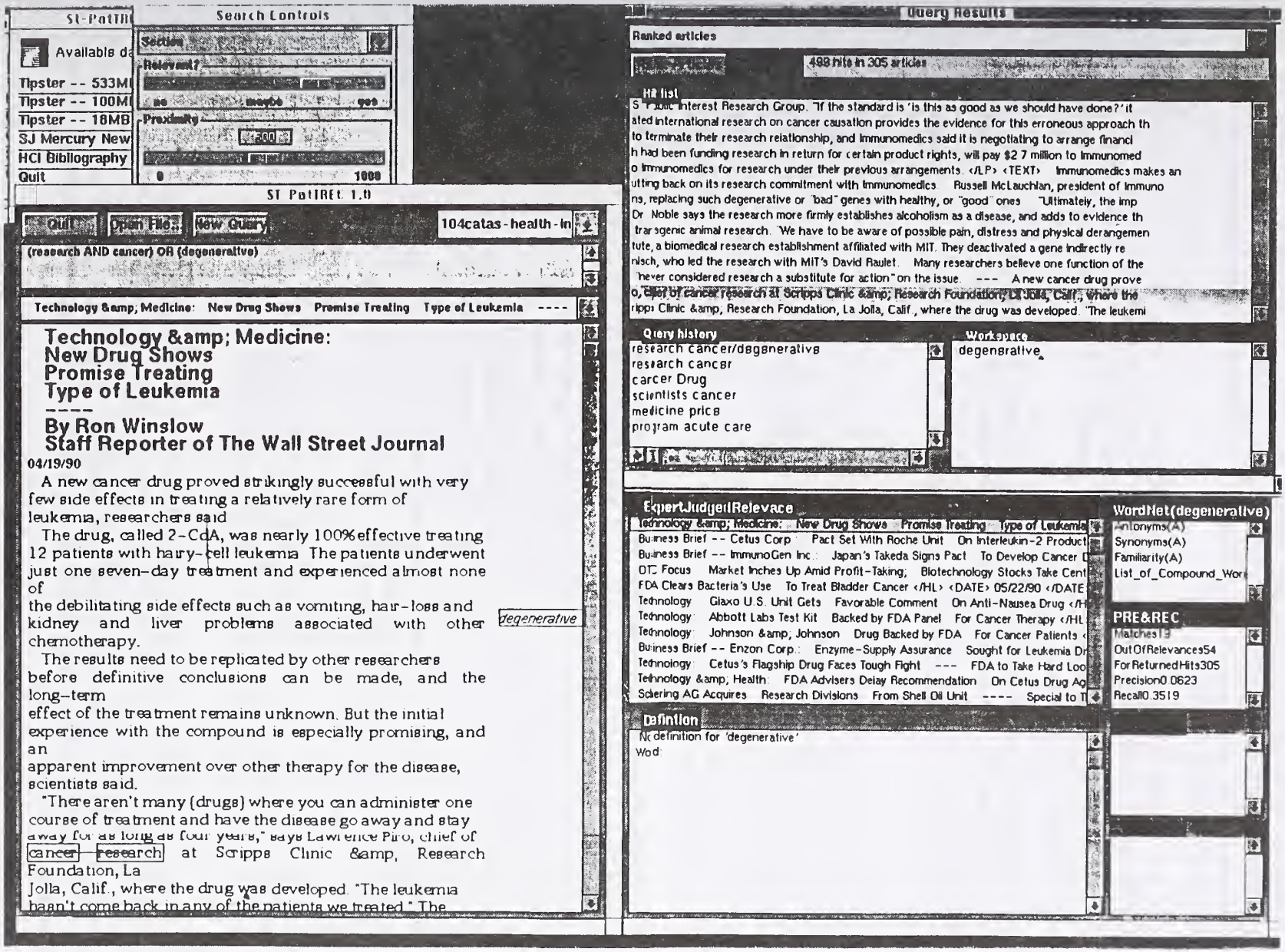


Figure 4. A Typical ST-PatTREC Screen.

The window to the left of the WordNet window displayed the list of documents judged to be relevant by NIST as per their procedure. Each line (choice) in the window represents a document. If the user clicks on one of the choices in the window, the corresponding document is shown in the text window.

The two windows above the expert judged relevance window are the query history window and the query construction workspace. The query history showed a listing of the previous queries that had been executed. The workspace provide a place where users could type words or terms that could later be pasted in to the margin. Alternatively the users could type words directly into the margin.

The large window above the query history showed the hit list. Each line in this list represented one document in the set of hits. When a line was clicked on in the hit list, the corresponding document was displayed in the text window, with the current query marked up on it. Thus users could edit the query by modifying the graphical query that overlaid the document in the text window.

The style of interface was a mixture of graphical user interface and command interface. Queries could be constructed using point and click interactions, but terms could also be typed in the margin or entered in the workspace. Since TREC-3 we have revised the system so that it retains the graphical user interface while adding more command-based functions and making it easier for users to construct complex queries.

II. Experimental Conditions

The TREC-3 experiment was carried out over an intense one week period. After some pilot testing with two experienced librarians, the actual searches were carried out by the two researchers. The two searchers worked on all fifty of the topics. One searcher was 29, the other 38 years of age. One of the searchers has had considerable searching experience, while the other searcher had only limited experience. However, the goal of our system was to minimize the need for experience and to make searching more productive for novices. To a large extent, there did not appear to be a significant difference between the search results obtained for the two searchers in our experiment. In terms of educational level, both searchers had more than eight years of post-secondary education. One searcher had an undergraduate degree in computer engineering, the other in psychology. Neither of the searchers had any experience or expertise with respect to any of the 50 TREC-3 topics. One searcher was an associate professor at the University of Toronto, the other was a Ph.D. student at the same university.

The searchers knew that they were participating in the TREC-3 evaluation and were familiar with the structures and the rules of TREC-3. Thus their goal was to construct a query that would maximize the recall and precision on the training data. To do this they could use any or all of the features and functions of ST-PatTREC.

The training process consisted of trying the system out and seeing what worked. Due to lack of time, and the fact that the searchers were also trying to iron out the bugs in the system, etc., there was little if any time for training. Our impression was that the earlier topics ended up being training for the later topics, and that our skill as searchers using ST-PatTREC increased significantly during the experiment. Thus we would say that no formal training was used.

III. Search Process

1. Clock time (i.e. real, elapsed time) per search, from the time the searcher was given the topic, until the final query was saved, in seconds.
Mean: 32.44 mins
Median: 28.50 mins
SD: 15.45 mins
Range: 12-60 mins

2. Number of documents "viewed" in during the search.
a. Definitions of viewing:
For the purposes of our system, viewing was defined as seeing a document in the text window. Users had to click on the hit list to bring a document into the text window and thus there was a clear indication of their intention to view documents.

b. Number of items viewed per search(repeat for each viewing category).
Mean: 18.18
Median: 16.5
SD: 7.912
Range: 3-42

3. Number of iterations per search.
a. Definition of iteration:
The searchers wrote up a search log that contained significant queries, along with recall and precision measures for those queries. An iteration was defined operationally as a new entry in the search log.

b. Number of iterations per search.
Mean: 8.9
Median: 8.5
SD: 3.388
Range: 3-18

4. Number of terms used in queries.
a. Number of terms in the first query of a search, per search.
Mean: 1
Median: 1
SD: 0
Range: 1

b. Number of terms in the final query of a search, per search.
Mean: 5.571
Median: 6
SD: 2.17
Range: 1-11

5. Use of system features.
Not available

6. Number of user errors made per search.
Not applicable

7. An Example of the system in Use: Topic 122

The following interaction shows how the interaction occurred for topic 122. This topic was concerned with new or alternative cancer drugs therapies. The title of the topic was "RDT&E of New Cancer Fighting drugs. The concepts provided in the topic statement included cancer, leukemia, drug, and chemotherapy. Figure 5 shows portions of the transaction log that our system produced for the interaction. The interaction started with the selection of the term "cancer" at 6:09 pm. on July 24. The transaction took almost exactly 20 minutes. This was a relative short interaction for the TREC-3 topics with our system, since the average topic took just over half an hour.

The user started by typing the words cancer, followed by drug, in the margin. The words were then linked together to create an AND. The term "development" was added to the query and then removed. At this stage an article was selected from the hit list. The user then added the term "research" to the query, first ORing it with cancer AND drug and then ANDing it to create the query research AND cancer AND drug. A document was then selected from the resulting hit list. According to the log, the user then had about 10 seconds to review the text of the document before the query was modified by removing the word "research". The user would also have noted the feedback for this query, which was a precision of .16 and a recall of .35. In contrast, the query "drug AND cancer" had provided a recall of .81 and a precision of .14, thus clearly being superior to the version of the query which added "research" to the AND combination. Although this log does not show it, the user had continuous updates on the precision and recall scores for each query executed. Thus, the evolution of the search strategy in constructing the test query was guided to a large extent by this feedback.

The user then tried adding "research" in a different version of the query, leading after an set of interactions over a period of almost three minutes to the query:

drug AND research AND cancer OR cancer AND development

A considerable amount of time was then spent trying to find a term that would improve the precision without adversely affecting recall, but without much success. The final enhancement to the query came when the term therapy was used (therapy was one of the coordinate terms for "cancer"). The final query eventually chosen for this topic was:

cancer AND drug OR cancer AND therapy

One interesting feature of this interaction is that of the four concept terms provided as part of the TREC-3 topic definition, only two (cancer, drug) were actually used in this session. Not using "leukemia" or "chemotherapy" was probably an oversight on the part of the user. When we amended the final query to include these terms , i.e., "cancer AND therapy OR cancer AND drug OR chemotherapy OR leukemia" the resulting precision and recall on the training database were .13 and .89, respectively. Thus "human errors" in forgetting to use, or consider, useful terms could have lowered the performance of queries using our system in some cases.

After this interaction, the final query submitted for the TREC-3 competition was "drug AND cancer" OR "cancer AND therapy". For the first 100 documents, 20 out of the 30 relevant documents were retrieved from this query, reflecting a recall of .67 and a precision of .20 for this subset of documents. This results were in line with the experiment gained in forming the query on the training data where using exact matching, the recall was .81 and the precision was .13. Expanding to 997 documents then yielded a further 6 relevant documents for a recall of .87. However the precision was then only .026. However, given that there were only 30 relevant documents, and 1000 documents were to be retrieved, the

maximum precision that could have been obtained at that point was only about 3 percent in any case.

```
18:09:06 query cancer
18:09:06 addterm cancer
18:09:17 query drug OR cancer
18:09:17 addterm drug
18:09:22 query drug AND cancer
18:09:22 addop drug AND cancer
18:09:46 query development OR drug AND cancer
18:09:46 addterm development
18:09:54 query drug AND cancer
18:09:54 removeterm development
18:10:02 selecthit hitlist 3124737 3122285
18:10:02 selectdoc 3122285
18:11:45 selecthit hitlist 28740917 28736070
18:11:46 selectdoc 28736070
18:12:43 query research OR drug AND cancer
18:12:43 addterm research
18:12:51 query research AND drug AND cancer
18:12:51 addop research AND drug
18:12:52 selecthit hitlist 20201575 20196494
18:12:53 selectdoc 20196494
18:13:03 query drug AND cancer
18:13:03 removeterm research
18:13:12 query drug AND cancer OR research
18:13:12 addterm research
18:13:18 query drug AND cancer OR research OR cancer
...(this section of the log deleted to save space)...
18:24:51 query therapy AND cancer OR cancer
18:24:51 addterm cancer
18:24:55 query therapy AND cancer OR cancer OR drug
18:24:55 addterm drug
18:25:00 query therapy AND cancer OR cancer AND drug
18:25:00 addop cancer AND drug
18:25:01 selecthit hitlist 10646564 10646443
18:25:02 selectdoc 10646443
18:25:35 selecthit hitlist 7206040 7205634
18:25:36 selectdoc 7205634
18:26:24 newquery
18:26:30 query therapy cancer/cancer drug
18:26:30 query therapy AND cancer OR cancer AND drug
18:26:37 selecthit hitlist 3124615 3122285
18:26:37 selectdoc 3122285
18:28:05 query development Cancer/research cancer/Cancer drug/Cancer test
18:28:06 query development AND Cancer OR research AND cancer OR Cancer AND
         drug OR Cancer AND test
18:28:51 selecthitoffset2 31463575 31463482
18:28:51 selectdoc 31463482
18:29:06 selecthitoffset2 117419014 117418921
18:29:07 selectdoc 117418921
logout
```

Figure 5. The Transaction Log for Topic 122.

Interactive Document Retrieval Using TOPIC®
(A report on the TREC–3 experiment)

Richard M. Tong
Verity, Inc.
1550 Plymouth Street, Mountain View, CA 94043

1 Introduction

This paper contains a description of the experiments performed by Verity, Inc. as part of the Third Text Retrieval Conference (TREC-3).[1]

Verity participated as an Interactive Category A system and performed the full set of routing and adhoc experiments, submitting complete sets of results for both experiments.

Section 2 of the paper contains a review of the TOPIC system itself and the data structures it produces. Section 3 of the paper contains a description of our experimental procedure. Section 4 contains an analysis of our official results. Section 5 contains some general comments on overall performance and a brief discussion of possible future directions. The Appendix contains additional details of our procedures, as well as an analysis of topic 122, the interactive "focus topic."

2 TOPIC System Configuration for TREC-3

TOPIC® is a commercial full-text retrieval system that is available for a wide range of hardware and software environments. For the TREC-3 experiments we used version 4.0 of the TOPIC client running on either Sun SPARC 2 class machines with an X/Motif GUI, or on 386/486 class machines running Windows. The TREC data and indexes themselves were stored on a Sun SPARC 2 file-server with approximately 4.6 Gbyte of disk storage. This file sever was used exclusively for TREC during the period of the experiments.

We used the standard set of Verity database administration tools to build the TREC database. We chose to extract only the document ID (i.e., the <DOCNO> field) and a "title" (e.g., the <HEAD> field in the AP collections, the <HL> field in the WSJ collections) for the structured elements in our databases, and then treated each document as a regular full text document. We did not use a stop word list, although we did remove the SGML tags, and so we chose to create just an inverted index on word positions within documents (as opposed to an index that also recorded sentence and paragraph positions) to keep the index at a reasonable size. The final index (for all three disks) was approximately 1.5Gbyte and took approximately 112 hours to build.

We used the training data provided for disks 1&2 with respect to topics 101–150 to define a series of "ground truth" datasets that we modelled as Topics. These ground-truth Topics were made available to all searchers as part of the interactive routing query development process. We also used TOPIC's launch facility to allow searchers to access the official TREC scoring program so that the performance of the routing queries (with respect to the training data) could be assessed as they were being developed.

3 The TREC-3 Experiments

For TREC-3 we participated in Category A and produced two sets of results in each of the routing and adhoc experiments. These results sets are labelled, naturally enough, TOPIC1, TOPIC2, TOPIC3 and TOPIC4.

TOPIC1 and TOPIC2 are the routing results (i.e., the results for topics 101 through 150) — TOPIC1 corresponding to the initial, manually generated, set of queries made available to all searchers participating in the experiment, and TOPIC2 corresponding to the final routing queries developed by the searchers. TOPIC3 and TOPIC4 are the adhoc results (i.e., the results for topics 151 through 200) — TOPIC3 corresponding to a set of automatically generated queries that we made available to all searchers, and TOPIC4 corresponding to the final adhoc queries developed by the searchers. TOPIC2 and TOPIC4 are thus our "official" submissions. This information is summarized in Table 1.

1. Requests for further information about the TREC-3 experiments, or Verity's line of information retrieval products, should be directed to the author either at the address above or electronically to `rtong@verity.com`.

Table 1: Results Identification

Result Set	TREC Category	Method	Queries
TOPIC1	Routing A	Manual	Initial set of queries; based on Verity's adhoc queries from TREC-2 [1].
TOPIC2	Routing A	Interactive	Final set of queries; the official routing queries for TREC-3.
TOPIC3	Adhoc A	Automatic	An automatically generated set of queries; based on a simple transformation of the TREC information need statements.
TOPIC4	Adhoc A	Interactive	Final set of queries; the official adhoc queries for TREC-3.

Since our approach to developing queries is based on the construction of conceptual representations of the domain of the queries, the final queries for both the routing and adhoc experiments are generally the result of several interactions between the developer of the query and the test collections. Our approach is thus fundamentally interactive, and for that reason we elected to participate in TREC-3 as an interactive system.

3.1 Experimental Procedure

The experimental procedure for Verity's TREC-3 experiments is briefly summarized here. It consisted of the following main steps:

- **Training of searchers.** All searchers participated in a short training session in which the basic TREC experiments were described and the test and evaluation procedures discussed. The novice searchers were also given a short tutorial introduction to TOPIC and query building. Those who needed it were given personal instruction in the use of TOPIC for the TREC experiment. Searchers were not, however, given any specific guidelines as to how exactly they should develop queries (i.e., no attempt was made to impose a standard procedure or a Verity "style"); they were simply instructed to make use of whatever features of the TOPIC query building language they found most useful.
- **Interactive routing query development.** All searchers had access to the ground-truth Topics, the general purpose thesaurus shipped as part of the standard TOPIC product, and the initial set of routing queries based on Verity's TREC-2 adhoc queries. Each searcher was asked to "sign up" for one of more topics and was then free to work on the queries to the extent that their individual schedules permitted. No constraints were placed on the amount of time searchers could (or should) spend developing queries, nor were any constraints placed on the number of "iterations" that needed to be performed. Searchers were asked to keep notes on progress in query construction. Query development ceased on a specific date (defined as one week before the routing results were due at NIST), at which time searchers submitted whatever they had developed up to that point.
- **Interactive adhoc query development.** The only assistance provided to searchers in the adhoc experiment was the set of automatically generated adhoc queries and the standard thesaurus. Searchers were given a fixed amount of time to develop queries corresponding to the new adhoc topics, and were asked to keep notes on their query construction efforts. Again, no special constraints were imposed on the process by which queries should be developed, and searchers submitted whatever they had developed by the end of the time period.
- **Generation of test data.** Once the query development phase was completed, the individual queries were merged to form the "official" query set. We then used an interactive, command-line version of TOPIC, called VSH, to generate the official results sets.[2]

3.2 Searcher Characteristics

Twelve Verity staff members participated in the TREC experiments as searchers. They ranged widely in TOPIC experience and job function, and they developed different

2. We discovered after the fact that this occasionally results in official scores that are different from the ones computed from within TOPIC v4.0 (i.e., the version of the product used by the searchers). This is due to the *order* in which results are generated within a relevance bracket. VSH generates them in the reverse order from TOPIC v4.0, so that if there are many documents in the relevance bracket at which the 1000 document cut-off occurs, it is possible that relevant documents could be above the cut-off in the TOPIC v4.0 results list, but below the cut-off in the VSH results list. This usually makes only a small difference, but occasionally can have significant impact on the recall scores.

numbers of queries (both routing and adhoc) as their schedules permitted. Summaries of these characteristics are given in Table 2. The column labels #R and #A denote, respectively, the number of routing and adhoc queries developed by each searcher. The designation as "expert" or "novice" refers to experience with TOPIC the product, and does not necessarily imply that the searchers were expert searchers, although several were. None of the searchers had any special expertise with respect to the subject domains of the routing and the adhoc topics.

Table 2: Searcher Characteristics

ID No.	Experience Level	Job Function	#R	#A
01	Expert	Engineering	23	29
02	Expert	Engineering	–	1
03	Expert	Education	1	–
04	Expert	Sales	5	5
05	Expert	Tech. Support	–	5
06	Novice	Engineering	1	–
07	Novice	Engineering	3	4
08	Novice	QA	2	3
09	Novice	QA	1	1
10	Novice	QA	–	1
11	Novice	Education	1	–
12	Novice	Documentation	–	1

4 Discussion of Official Results

This section contains an analysis of the official results generated by NIST. Section 4.1 discusses the results from the routing experiment, Section 4.2 discusses the results from the adhoc experiment.

4.1 The Routing Experiment

Of the 50 queries submitted for the official routing experiment, only 36 were actually the result of interactive development. Of these 36, 29 were developed by TOPIC experts and 7 by TOPIC novices. The remaining 14 were just the initial manual queries.

An analysis of the notes kept by searchers suggests that the average time spent developing queries varied widely, from as little as 30 minutes to as much as 20 hours. On average though, experts spent significantly less time than novices on query development; approximately 90 minutes per query versus approximately 400 minutes per query.[3]

A comparison of the initial and final queries shows that most changes were relatively minor, in that they typically involved adding specific query terms and adjusting weights and operators. However, in approximately 10 queries there was more significant changes, including complete rewrites and major structural modifications.

The results generated by the two sets of queries with respect to the *training* data (designated TOPIC0 — corresponding to the initial queries, and TOPIC00 — corresponding to the final queries) indicate a generally small, but positive, improvement in scores. Table 3 shows the main performance measures. Inspection of the results on a per-query basis do not reveal any significant patterns of behavior *vis a vis* these measures, and so we are not able to draw any particular conclusions about the query building strategies of searchers. Anecdotal evidence suggests that most searchers adopted a strategy that amounted to getting the Rel_Ret number to a "reasonable" level and then focusing on the Prec. @N measures.

Table 3: Routing Queries on Training Data

Performance Measure	TOPIC0	TOPIC00	Δ%
Rel_Ret	6599	7243	+9.76
Av_Prec.	0.2510	0.2913	+16.06
Prec. @10	0.5840	0.6160	+5.48
Prec. @100	0.4582	0.4766	+4.02
R-Prec.	0.3174	0.3459	+8.98

Table 4 shows the same performance measures for the official routing test data. Notice that the revised queries again outperformed the initial queries. The improvements in performance are, however, significantly larger than on the

3. The times are very approximate, based as they are on the notes kept by the searchers. For the novices, the time spent often reflects lack of familiarity with both the product and the general problem of searching large text collections. No searcher kept detailed enough records that we can say anything definitive about the "number of iterations" or the "number of documents read" during the query building process, or about the features of the product that users found especially useful or frustrating.

Table 4: Routing Queries on Test Data

Performance Measure	TOPIC1	TOPIC2	Δ%
Rel_Ret	5371	5952	+10.82
Av_Prec.	0.2243	0.2774	+23.67
Prec. @10	0.4740	0.5280	+11.39
Prec. @100	0.3382	0.3880	+14.73
R-Prec.	0.2786	0.3343	+19.99

training data. The absolute level of the performance is less than on the training data (as one would expect) — although perhaps not significantly so; Av_Prec. is reduced by 4.77% and R-Prec. is reduced by 3.35% when we compare TOPIC00 to TOPIC2.

Examination of the results on a per-query basis shows a strong correlation between performance improvements on the test data and improvements on the training data. That is, an increase in the performance of a query on the training data usually corresponded to a similar increase on the test data, although not always to the same degree. This suggests that the changes made by the searchers were not in response to any specific characteristics of the training data, but really did reflect an improved understanding of the intent of the queries.[4]

Comparison of our results with the summary results for Category A systems, shows that most of the TOPIC2 queries were at the median or somewhat below on the average precision measure. Of the rest, two were complete failures (that is, they returned no relevant documents), five were close to the minimum, and three posted the maximum average precision score. Figure 1 shows the differences between the TOPIC2 results and the medians for average precision.

Preliminary analysis indicates little correlation between performance and searcher expertise or job function. In fact, of the two queries that were complete failures (for topics 131 and 141), one was done by an expert and one by a novice; and of the three queries that did best in category A (for topics 105, 111, and 145) two were done by experts, and one was done by a novice.

We also did not find any interesting correlation between performance and the "hardness" of topics 101–150. In her RIAO'94 paper [2], Harman defines hardness in terms of the relative recall, which is a measure designed to show how systems perform at the early stage of retrieval, and while this appears to be a reasonable predictor of the median average precision scores for the TREC-3 routing experiment, we found it to be less effective in predicting our own average precision scores.

The overall results for the routing experiment are somewhat surprising since we had predicted that we would achieve significance performance gains *vis a vis* the TREC-2 results. Our tentative explanation is that while our interactive approach can be made to work very well indeed (*vide* the performance on topics 111 and 145), human searchers are not, in general, very effective at exploiting large amounts of training data compared with the statistical techniques used by the majority of the automatic methods participating in TREC-3.

4.2 The Adhoc Experiment

All 50 of the adhoc queries for the TOPIC4 results set were generated interactively by the searchers, and of these, 40 were generated by TOPIC experts and 10 by novices. As in the routing experiment, the average time spent by novices and experts in developing queries was markedly different; approximately 80 minutes per query for experts versus approximately 274 minutes per query for novices. While this difference is of a similar magnitude as in the routing experiment, the much reduced novice times are at least partly explained by the fact that those novices who took part in both experiments had, by the time of the adhoc experiment, achieved some skill with TOPIC, thus enabling them to develop queries more efficiently.

The anecdotal evidence from the notes kept by searchers suggests that the nature of the adhoc test also acted to reduce the amount of time spent on each query. That is, since there is no "ground truth" against which to evaluate the results of the query building effort, searchers often seemed satisfied, or at least were not willing to spend any further effort, after relatively little interaction.[5]

The searchers in the adhoc experiment were not constrained in any way with respect to the number of iterations they could perform, the number of documents they could view, the number of times they could view any given document, or in any other aspect of their interactions with the

4. As of this writing, we have not performed a detailed statistical examination of these correlations, so these observations should be treated with some caution.

5. Again, the searchers' notes are not detailed enough to determine specific figures for the number of iterations or the number of documents read. An examination of the final adhoc queries shows, however, that many of them are very simple (both in terms of the number of terms they include and the degree of structuring they exhibit), which supports the conjecture that little effort was spent developing them.

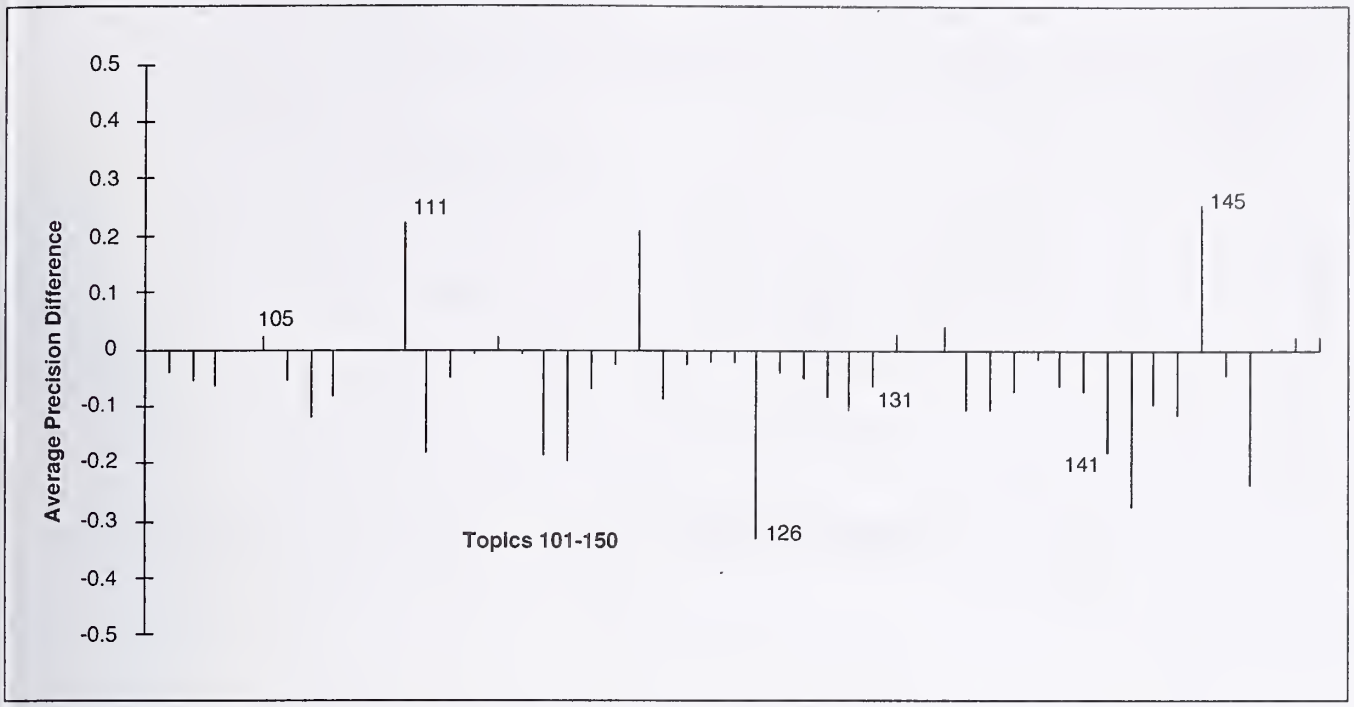


Figure 1: Average Precision Difference (Topics 101–150)

system. This is very much a "free style" approach to query development, in contrast with more formal methodologies such as the various form of frozen rank testing, but is strongly representative of how our system is used in practice. The results of the adhoc experiment (i.e., the TOPIC4 results set) should therefore be viewed as an instance of how a manual, interactive system can perform, rather than as a set of data that is directly comparable with, say, the automatic systems participating in the adhoc experiment.

The overall results for the adhoc experiment are shown in Figure 2. Notice that compared to the median average precision scores for all Category A systems the results look quite good. There is much more variability than in the routing experiment, but on balance the scores are above median, with five of the queries (for topics 158, 159, 172, 181 and 183) posting the best average precision score. As with the routing experiment though, we do not find any significant correlation between searcher experience and query performance. Somewhat counterintuitively, however, the results do seem to be generally lower (relative to the median) and more variable as the median scores increase. We have no immediate explanation for this.

5 Commentary

Verity's approach to full-text information retrieval is based on a belief that users need to be provided with a query language that enables them to built personalized semantic models of their information needs. Within this context, we see query development as a form of conceptual modelling, and thus have designed the TOPIC system to be fundamentally interactive. Although we provide a number of automatic query expansion methods, our philosophy is that users need to be actively engaged in the query development process if they are to be convinced of both the soundness of the system and the quality of the search results.

The results of the TREC-3 experiment show that this approach can be very effective, especially in situations where the searcher cannot draw upon large numbers of exemplars. However, the TREC-3 results also show that automatic systems that are able to extract significant search terms and phrases from large sets of training data performed better, on average, than the manually generated TOPIC queries. We should probably not be surprised by this. Humans are not particularly well suited to the task of determining statistical correlations among large sets of data, so that to the extent that good retrieval performance depends on finding those correlations, automatic methods will outperform manual ones. We note, though, that even in the routing experiment, the TOPIC queries outperformed all others in three cases. This suggests that human expertise and judgement can be the critical factor in developing effective queries. Further analysis of our queries, as well as a comparison with those generated by automatic systems, will of course be required to verify this conjecture.

The experimental data we collected with respect to the performance of individual searchers was, unfortunately, not particularly informative. The only definitive statements we

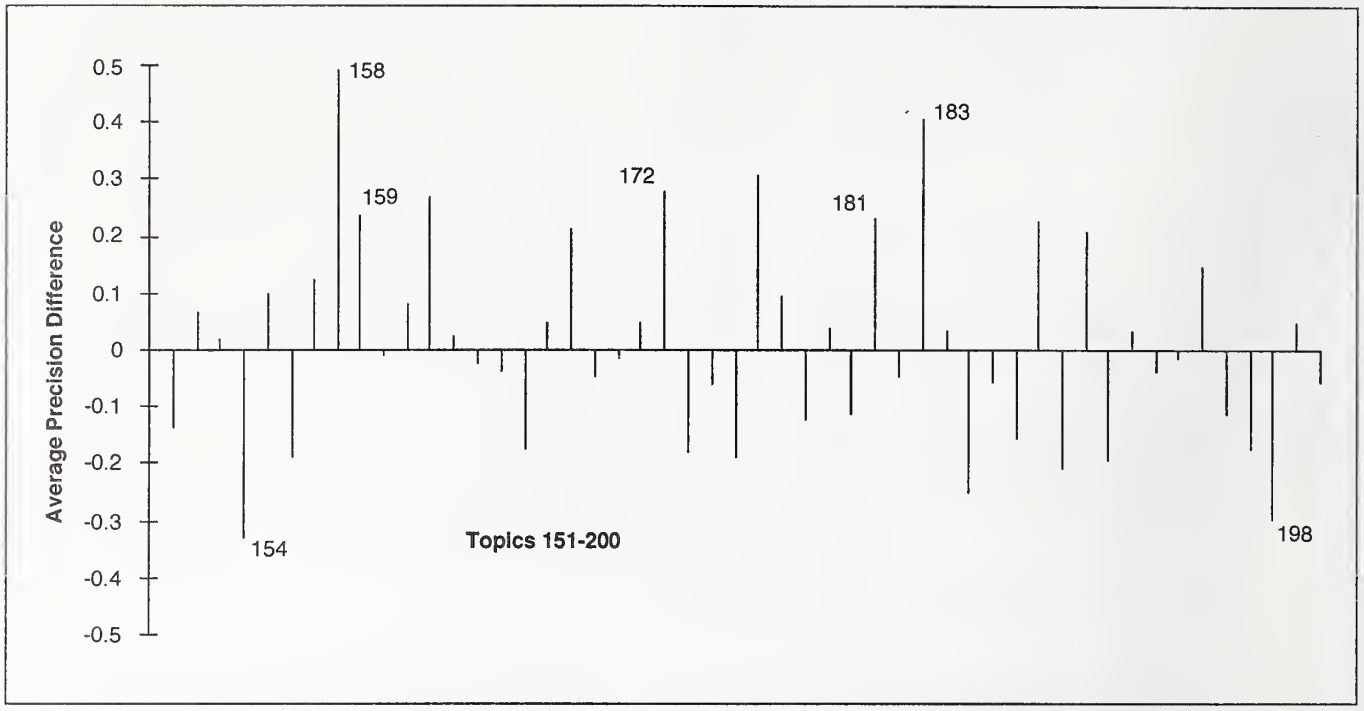


Figure 2: Average Precision Differences (Topics 151–200)

can make are that "experts" spent less time developing queries on average than "novices," and that less time on average was spent on the adhoc queries than on the routing queries. We found no significant correlations between performance and user expertise, nor between performance and query "hardness." Neither did we find any correlation between performance on individual queries and the amount of time spent developing them. In the case of the routing experiment, we did find that the performance improvements realized on the training data were also realized on the test data, which suggests to us that the searchers were indeed building conceptual models, rather than just responding to surface level characteristics of the training data.

Overall, we feel that this was a worthwhile exercise for us. While our results on the routing experiment were not as strong as we had anticipated, we did observe some searcher behaviors with respect to our product that will allow us to make improvements to the Topic construction process. The adhoc results were very satisfactory, despite the relatively small amounts of effort most queries received, and show that the TOPIC query modelling language can be used to great effect.

References

1. Lehman, J. W., Reid, C. A., et al. Knowledge-based searching with TOPIC. In: Harman, D. K., ed. The Second Text Retrieval Conference (TREC-2). NIST Special Publication 500-215, Gaithersburg, MD: March 1994.

2. Harman, D. Analysis of Data from the Second Text REtrieval Conference (TREC-2). Proc. RIAO'94: Intelligent Multimedia Information Retrieval Systems and Management, pp 699–709. Rockefeller Univ., New York, NY. October 11–13, 1994.

A Appendix

This appendix contains some additional details of the experimental procedures used in the routing experiment. In particular, it contains a description of the TOPIC user interface together with a description of the main functional elements in that interface, and a detailed analysis of the results for topic 122.

A.1 The TOPIC User Interface

TOPIC is a commercial full-text retrieval system available for a wide range of hardware and software platforms, and can be configured in a number of ways for different environments. A typical query development screen from the X/Motif version of the user interface is shown in Figure 3.

The display is divided into two main sections. The upper portion contains the query (in this case a partially expanded Topic tree for topic 146), and the lower portion contains the hit list. The relative sizes of these two panes are completely controllable by the user, as is, of course, the overall geometry of the TOPIC window.

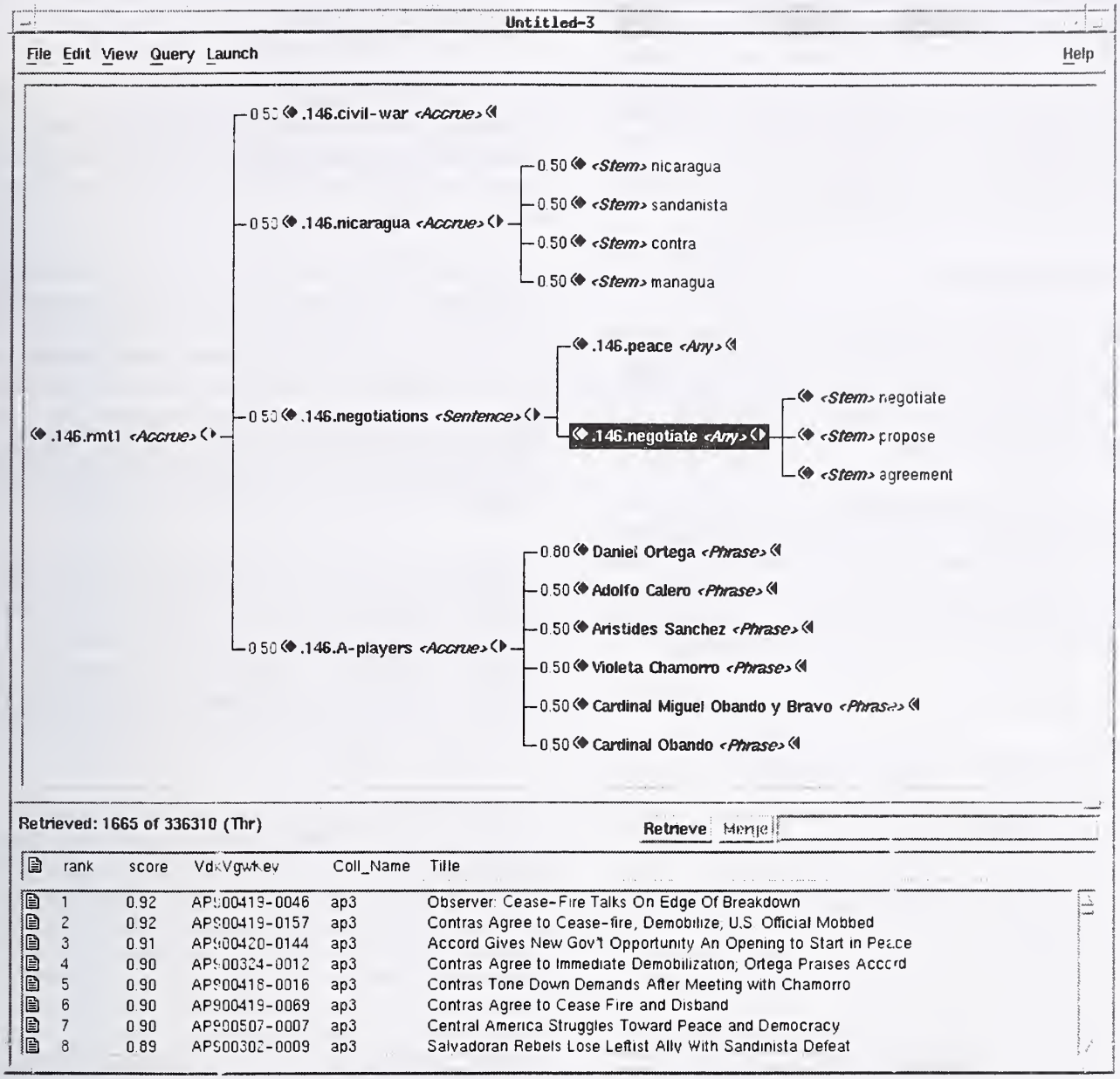


Figure 3: Typical TOPIC Interface Screen

The pull-down menus at the top of the window provide access to a large number of support functions, including the usual array of system level commands (the File menu), commands for editing and manipulating queries and associated data structures (the Edit menu), commands that control the way in which the query screen appears (the View menu), commands for setting parameters associated with the query itself (the Query menu), and commands that are calls to applications outside of TOPIC proper (the Launch menu).

The query pane itself is interactive. All commonly required editing functions are available via a mixture of point-and-click, pop-up menus and drag-and-drop operations. Thus a searcher can quickly build a sophisticated query that can be viewed graphically. The interface also supports some other views of the query, including a, so called, simple query in which users can enter unrestricted query terms, and a form-based query in which constraints can be placed on the searches to be performed on the fielded components of the documents.[6]

The results list is also configurable, but is shown with the default for the TREC experiment. That is, we show the document rank, its relevance score, its ID, the collection it comes from, and its "title." To view a document, the

searcher simply double-clicks on required item and the corresponding full-text of the document is displayed, with highlights, in a separate window.

We collected little data on the use of specific features in TOPIC. As we would predict, the anecdotal evidence suggests that the experienced TOPIC users made use of more features than the novices, but we can draw no specific conclusions from the limited data we have.

A.2 The Focus Topic

An important component of the interactive test for TREC-3 was the idea of using topic 122 as a "focus topic." For this focus topic we provide some specific details about query construction behavior, as well as a closer examination of the actual query developed.

Unfortunately, a logistical error in collecting the queries resulted in an intermediate query being used for the generation of the official TREC-3 results (i.e., the TOPIC2 results set), and so we present here an analysis of the final query the searcher actually produced. Therefore, of course, the performance measures differ from those officially reported.

The statement of information need for topic 122 is:

```
<dom> Domain: Medical & Biological
<title> Topic: RDT&E of New Cancer
  Fighting Drugs
<desc> Description:
  Document will report on the research,
  development, testing, and evaluation
  (RDT&E) of a new anti-cancer drug
  developed anywhere in the world.
<narr> Narrative:
  A relevant document will report on any
  phase in the worldwide process of
  bringing new cancer fighting drugs to
  market, from conceptualization to
  government marketing approval.  The
  laboratory or company responsible for the
  drug project, the specific type of
  cancer(s) which the drug is designed to
  counter, and the chemical/medical
  properties of the drug must be
  identified.
<con> Concept(s):
  1. cancer, leukemia
  2. drug, chemotherapy
```

We see that this is a complex topic that requires a document to contain a number of specific data items if it is to be considered relevant. Most information retrieval systems can approach this topic only indirectly, and TOPIC is no exception.

The initial formulation of the query (i.e., the one used to generate the results in the TOPIC1 results set) is essentially disjunctive with two major components — one that searches for variations of the "new anti-cancer drug" concept; and one that searches for information related to anti-cancer drugs, including the names of specific known drugs, the names of various cancers, the names of agencies concerned with regulating drugs, and synonyms for "treatment" and "research."

The final formulation of the query retains the basic elements of the initial query, but reorganizes them so that a retrieved document must contain evidence for the "new anti-cancer drug" concept (i.e., a necessary condition), and thereafter the degree of relevance is a function of the amount of related information found. This is modelled in TOPIC using a proprietary construct that can be thought of as a weighted conjunct.

The performance of these two formulations on the training data (i.e., the performance the searcher was responding to in their interaction with the system) are shown below. The scores for the initial formulation are labelled TOPIC0; the scores for the final formulation are labelled TOPIC00.

```
Queryid (Num):       122
                   TOPIC0      TOPIC00

Total number of documents over all queries
    Retrieved:     1000         1000
    Relevant:       119          119
    Rel_ret:         71           72
Interpolated Recall - Precision Averages:
    at 0.00       1.0000       1.0000
    at 0.10       0.4828       0.5517
    at 0.20       0.3429       0.3871
    at 0.30       0.2628       0.2975
    at 0.40       0.1582       0.2134
    at 0.50       0.1202       0.1760
    at 0.60       0.0000       0.0940
    at 0.70       0.0000       0.0000
    at 0.80       0.0000       0.0000
    at 0.90       0.0000       0.0000
    at 1.00       0.0000       0.0000

Average precision (non-interpolated) over
  all rel docs
```

6. The simple query mode is mostly used in the initial stages of query development when the searcher is more concerned with determining an appropriate set of retrieval terms than with the structuring of these terms. Once a reasonable set of terms have been identified, searchers usually switch to the graphical mode. The form query was not especially useful for TREC-3, since we provided only a limited number of searchable fields (i.e., the Coll_Name and Title fields). Indeed, an analysis of the TREC-3 data shows that no searcher used them in the final formulations of the submitted queries.

```
                    0.1782       0.2226

   Precision:

  At    5 docs:    0.6000       0.8000
  At   10 docs:    0.6000       0.8000
  At   15 docs:    0.6000       0.6667
  At   20 docs:    0.5000       0.5500
  At   30 docs:    0.4667       0.5333
  At  100 docs:    0.3000       0.3200
  At  200 docs:    0.2050       0.2300
  At  500 docs:    0.1200       0.1280
  At 1000 docs:    0.0710       0.0720

R-Precision (precision after R (= num_rel
  for a query) docs retrieved):
    Exact:         0.2857       0.2941
```

Inspection of these data shows that the revised query did indeed outperform the initial query, although not dramatically so. Overall recall remained the same, but average precision and precision@N were significantly better.

The searcher reports that this was a difficult query to develop and, although many variations were tried,[7] there was no query that generated a significantly higher recall while at the same time preserving "reasonable" precision. The searcher felt that the precision improvements exhibited by the selected variant were significant enough that the lack of recall improvement was "acceptable."

The main difficulty with this topic is, of course, that it requires relevant documents to describe a process (i.e., the development and marketing of a new drug) and identify specific entities involved in that process (i.e., the drug company, the properties of the drug, and its target). Since the process is not completely described by the information need statement (and was not known to the searcher) and there is no enumeration of the companies, drugs and cancers, the searcher developed a query that attempts to model various kinds of collateral indicators. As we would expect, this was a difficult, and essentially non convergent, task that required the searcher to amass a collection of relatively weak indicators of relevance (i.e., search terms). Given that no one group of indicators dominates, finding the right combinations and weights reduces to a search in a high-dimensional space — something humans are not very good at. This is, we believe, the reason why the searcher found this to be a hard topic, and ultimately found it difficult to build a satisfactory query.

7. The searcher's notes show that twelve major variants of the query were tested. A variant could include either a restructuring or a redefinition of base-level query terms, or both, and was marked as such if the searcher preformed a complete retrieval (i.e., ran the query over the entire training corpus) and then generated set of retrieval results using the TREC evaluation program.

The performance of these two queries on the actual routing experiment are shown below. The original query generated the results labelled TOPIC1, and the final query generated the results labelled TOPIC2.

```
Queryid (Num):       122
                   TOPIC1       TOPIC2

Total number of documents over all queries
    Retrieved:      1000         1000
    Relevant:         63           63
    Rel_ret:          61           44
Interpolated Recall - Precision Averages:
    at 0.00        1.0000       1.0000
    at 0.10        0.9091       0.9231
    at 0.20        0.7000       0.8235
    at 0.30        0.2159       0.3231
    at 0.40        0.1561       0.2593
    at 0.50        0.0862       0.0831
    at 0.60        0.0862       0.0588
    at 0.70        0.0862       0.0000
    at 0.80        0.0759       0.0000
    at 0.90        0.0715       0.0000
    at 1.00        0.0000       0.0000

Average precision (non-interpolated) over
  all rel docs
                   0.2900       0.3026

   Precision:

  At    5 docs:    0.8000       0.8000
  At   10 docs:    0.9000       0.9000
  At   15 docs:    0.6667       0.8000
  At   20 docs:    0.7000       0.7000
  At   30 docs:    0.5333       0.5333
  At  100 docs:    0.1900       0.2500
  At  200 docs:    0.1400       0.1400
  At  500 docs:    0.0800       0.0700
  At 1000 docs:    0.0610       0.0440

R-Precision (precision after R (= num_rel
  for a query) docs retrieved):
    Exact:         0.2857       0.3175
```

What is striking about these results is the very significant reduction in the number of relevant documents retrieved by the revised query. In contrast, the precision measures of interest (i.e., average precision, precision@10, precision@30, and R-precision) are either the same or higher for the revised query.

These data suggest, perhaps, that the reformulated query was a good model of those documents in the test collection that were similar in character to the training data, but that the necessary condition built into the revised query was too restrictive. However, as of this writing, we have not done a detailed enough analysis of the results to determine the exact causes for the low recall score.

TREC-3 Ad Hoc Retrieval and Routing Experiments using the WIN System

Paul Thompson
Howard Turtle
Bokyung Yang
James Flood
West Publishing Company
Eagan, MN 55123

1 Introduction

The WIN retrieval engine is West's implementation of the inference network retrieval model [Tur90]. The inference net model ranks documents based on the combination of different evidence, e.g., text representations, such as words, phrases, or paragraphs, in a consistent probabilistic framework [TC91]. WIN is based on the same retrieval model as the INQUERY system that has been used in previous TREC competitions [BCC93, Cro93, CCB94]. The two retrieval engines have common roots but have evolved separately – WIN has focused on the retrieval of legal materials from large (>50 gigabyte) collections in a commercial online environment that supports both Boolean and natural language retrieval [Tur94]. For TREC-3 we decided to run an essentially unmodified version of WIN to see how well a state-of-the-art commercial system compares to state-of-the-art research systems.

Some modifications to WIN were required to handle the TREC topics, which bear little resemblance to queries entered by online searchers. In general we used the same query formulation techniques used in the production WIN system with a preprocessor to select text from the topic in order to formulate a query.

WIN was also used for routing experiments. Production versions of WIN do not provide routing or relevance feedback so we were less constrained by existing practice. However, we decided to limit ourselves to routing techniques that generated normal WIN queries. These routing queries could then be run using the standard search engine.

In what follows, we will describe the configuration used for the experiments (Section 2) and the experiments that were conducted (Sections 3 and 4).

2 System Description

The TREC-3 text collection was indexed in essentially the same way for both the ad hoc and routing experiments. Some fields within each document were not indexed; these fields include: CO, DESCRIPT, DOC, DOCID, DOCNO, FILEID, FIRST, G, GV, IN, MS, NS, RE, SECOND. These fields were excluded either because they contained manually indexed terms (which cannot be used under the TREC rules) or because the were considered to be

noise. A bounded paragraph algorithm [Cal94] was used to identify paragraph boundaries. Natural paragraphs were used subject to the constraint that a paragraph had to contain a minimum of 50 and a maximum of 200 words.

All of the text not contained in these fields was indexed except for Federal Register documents. Federal Register documents tend to be very long and to contain a great deal of noise. In an attempt to identify text that was a reasonable description of document content we indexed only the "SUMMARY" paragraph if the document contained one, otherise we indexed only the first kilobyte of text in a Federal Register document. Since no Federal Register documents were contained in the routing test collection all text except for the excluded fields was indexed.

3 Ad hoc experiments

The ad hoc experiments used queries that were automatically created from the topic text. The retrieval algorithm used combined document and top paragraph scoring. It was observed that the a priori likelihood of relevance for a document varied from collection to collection. Furthermore each collection's likelihood of relevance given the value of domain field, varied, as well. Some experiments were done in an attempt to exploit these observations.

3.1 Query Processing

A WIN query consists of concepts extracted from natural language text. Rather than extracting concepts from the full topic only the Title field, the Description field, and the first sentence of the Narrative field were used. Each occurrence of a term, or concept, was counted and weighted by field. A term appearing in Title was given a weight of 4, while terms appearing in Description and Narrative were given weights of 2 and 1, respectively.

Normal WIN query processing eliminates introductory clauses and recognizes phrases and other important concepts for special handling. Many of the concepts ordinarily recognized by WIN are specific to the legal domain (e.g., legal citations, West Key Numbers) and were not used in these experiments.

WIN ordinarily makes use of a dictionary of introductory clauses (e.g., "Find cases about ...", "I'm interested in statutes that ...") that don't bear directly on the content of the query. The set of introductory clauses was expanded to include 170 new clauses (e.g., "A relevant document must describe ...") identified in the Description and Narrative fields in the training set. In addition the string "e.g" was added to the set of query stopwords.

WIN also expands some query terms automatically. For example "usa", "us", "u.s", and "united states" were all replaced with the synonym class #syn(ac:us #+1(united states)) that will conflate common variants. Twenty nine new synonym classes were added for automatic expansion.

WIN ordinarily uses a legal dictionary to find phrases in queries. For TREC-3 the dictionary was expanded with phrases extracted from the machine-readable Collins Dictionary. The normal WIN dictionary incorporates information about how a phrase identified in a query is to be matched in document text. For example, query stopwords are generally not

		AP	DOE	FR	WSJ	Ziff
Topics	% of all documents	22.2	30.5	6.2	23.3	17.8
1- 50	% of relevant docments	20.1	0.8	3.1	33.9	42.2
		0.91	0.03	0.50	1.45	2.37
51-100	% of relevant docments	37.2	7.5	3.1	38.0	14.2
		1.68	0.25	0.50	1.63	0.80
101-150	% of relevant docments	41.3	5.8	3.5	39.2	10.2
		1.86	0.19	0.56	1.68	0.57
Total	% of relevant docments	31.4	4.4	3.2	36.7	24.3
		1.41	0.14	0.52	1.58	1.37

Table 1: Collection bias in relevance judgments

considered to be significant, but for some phrases (e.g., "at will") they are used. None of the phrases extracted from the Collins Dictionary used any special recognition features.

3.2 Experiments with different likelihoods of relevance based on collection

In the TREC training set, the likelihood that a document will be judged relevant depends heavily on the collection in which it is found. Table 1 shows the distribution of documents among the five TREC collections and the distribution of relevant documents among the five collections. The AP collection, for example, contains 22.2% of all documents in the TREC collection, but it contains 31.4% of all relevant documents in the TREC collection. Table 1 shows that, for all topics, documents in two of the collections (DOE and Federal Register) are substantially less likely to be judged relevant as would be expected if there were no collection bias whereas documents from the Wall Street Journal, AP, and Ziff collections are much more likely to be judged relevant than expected.

Table 1 also shows that the distribution of relevant documents among the collections varies for different topic sets. For example, Ziff documents are much more likely to be judged relevant than expected for Topics 1-50, but less likely than expected for the remaining two topic sets.

A set of experiments was conducted in which the prior probability of relevance was set to the observed probability of relevance for each of the TREC collections rather than a default probability that was the same for all documents. This essentially biased retrieval in favor of AP, Wall Street Journal, and Ziff documents and against DOE and Federal Register documents. These experiments showed a slight drop in retrieval effectiveness because the priors computed for the entire topic set rarely match the priors computed for individual topics.

A second set of experiments was conducted to determine whether it would be possible to predict the appropriate collection biases based on the characteristics of individual topics. Approaches were tried using both the language contained in the topics and the domain field contained in many of the training topics (note, however, that the test topics do not contain domain fields). None of these approaches significantly improved performance, but the amount of effort devoted to these experiments was limited. We regard this as a promising

line of future research.

4 Routing Experiments

The routing experiments used the same techniques as the ad hoc experiments to index the text collection, except that idf values were derived differently. Since the test collection was to be used as a simulation of routing, the TREC guidelines do not allow use of any collection wide statistics, such as idf. Accordingly, the idf values from the CD-1 training set were used instead. Query processing, or profile creation, however, was done in a substantially different manner. No attempt was made to use the observed likelihoods of relevance of different collections, as was done with the ad hoc queries. The routing experiments were based on query expansion. No term reweighting was done.

4.1 Profile Processing

As was the case with the ad hoc queries, only certain portions of the topic text were used for profile creation. These were the Title and Concepts fields. As before, each occurrence of a term, or concept, was counted and also weighted by field. A term appearing in the Title field was given a weight of 2, while a term appearing in the Concepts field received a weight of 1. Any term not appearing in any of the relevant training documents, was removed. Consideration was given to increasing the weight of any term appearing in relevant, but not in irrelevant documents. This had no effect. Only one term met this condition. As a form of normalization the maximum weight that a term could attain was set. This weight was variously set at 5, 6, 7, and 8. This maximum included the contribution provided by the term expansion process, which was always 1 for a selected term, or 0 for a non-selected term (see below). A term might appear multiple times in the Concepts field, thus resulting in a unnormalized term weight that exceeded the maximum.

None of the usual WIN query formulation aids used with ad hoc queries (elimination of introductory clauses, use of replacement strings, and use of a phrase dictionary) were used for profiles. The Title and Concepts fields did not contain any introductory phrases, or clauses. Simple acronyms, such as "RISC" or "MIPS" that were found in the text of relevant training documents during query expansion were identified as acronyms in the profiles, so that they would be treated as instances of the same concept in subsequent processing.

4.2 Query expansion

The focus of the routing experiments was on query expansion. Three different approaches to query expansion were used: "best entire document", "best rntidf top 200 paragraphs", and "best rntidf top paragraph". Ultimately these three approaches were combined in the "best overall" approach. The approaches were themselves based on three methods of term selection: "rdfidf", "rntidf", and "rtfidf" [HC93]. The rdfidf score of a term was calculated by multiplying its idf value by the number of relevant training documents in which the term occurred. The rntidf score for a term was calculated by multiplying its idf value by the summation over all relevant training documents of the ratio of the term's frequency to

the frequency of the maximally-occurring term for that particular document. The rtfidf score was simply the multiplication of the term's idf value by the number of occurrences of the term within relevant training documents. The rtfidf score did not perform as well as the other two term selection methods, and so was not used as part of the final runs. For each term selection method, terms selected were those with the highest scores. Terms were only selected from relevant documents. The term scores were only used for term selection, not for term reweighting. A selected term was given a weight of 1 in the expanded query. If the term duplicated a term already represented in the topic profile, then 1 was added to that term's current score.

A baseline run was made using the profile creation process described above, but with no term expansion. Each of the three expansion approaches was then run with terms added by one or both of the remaining term selection methods, i.e., excluding rtfidf. For each approach runs were made with from 5 to 50 terms added, in increments of 5. The "best entire document" approach used both the rdfidf and the rnfidf methods of term selection, with terms selected from any part of the document. The term selection that performed better on the training set was selected on a topic per topic basis. With the "best rntidf top 200 paragraphs", and the "best rntidf top paragraph" approaches, as their names imply, only the rntidf method was used, as it provided better results. For the "best rntidf top 200 paragraphs" approach searches were done for each topic using the baseline, i.e., unexpanded, profile as a query against the training collection. For each topic the top 200 scoring paragraphs from relevant documents were identified, using the WIN paragraph scoring method. Terms were then selected from these paragraphs using the rntidf method, rather than from the entire text of the relevant documents. For the "best rntidf top paragraph" approach a similar procedure was followed, except that instead of using the top 200 paragraphs from any relevant documents, the top scoring paragraph of each relevant document was used as a source for rntifd term selection. For each of the three approaches the maximum weight allowed for a term, i.e., 5, 6, 7, or 8 (see section 4.1), on a topic by topic basis, was the weight that gave the best performance on the training set.

Finally, the method of query expansion used on the officially submitted run, "best overall", was a combination of the three approaches described above. This method was to select the best query expansion provided by any of the three approaches on a topic per topic basis. Rather than simply selecting the best approach per topic in this manner, some consideration was given to trying to combine the results of the different methods [FS94, BKCQ94], but no experiments were carried out.

5 Summary

WIN was able to achieve strong performance on both the ad hoc retrieval and routing tasks without any major modifications being made to its retrieval engine. The ad hoc results show the effectiveness of its basic indexing and retrieval operations. Some techniques that were expected to give improved performance, did not lead to much improvement. In some cases this may be because only limited investigations could be done, e.g., when using the collection-dependent likelihood of relevance. In other cases, such as the failure of phrases, to yield much improvement, the result may indicate the difficulty in effective use of a feature

which has given good results on smaller collections on a collection the size of the TREC collection [Har93].

References

[BCC93] John Broglio, James P. Callan, and W. Bruce Croft. INQUERY system overview. In *Proceedings of the TIPSTER Text Program (Phase 1) Workshop*, pages 47–67, Morgan Kaufmann, September 1993. ISBN:1-55860-337-9.

[BKCQ94] N. J. Belkin, P. Kantor, C. Cool, and R. Quatrain. Combining evidence for information retreival. In Donna K. Harman, editor, *The Second Text Retrieval Conference (TREC-2)*, pages 35–44, National Institute of Standards and Technology, March 1994. Proceedings available as NIST Special Publication 500-215.

[Cal94] James P. Callan. Passage-level evidence in document retrieval. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proceedings of the Seventeenth Annual International Conference on Research and Development in Information Retrieval*, pages 212–221, Springer-Verlag, London, July 1994.

[CCB94] W. Bruce Croft, Jamie Callan, and John Broglio. TREC-2 routing and ad-hoc retrieval evaluation using the INQUERY system. In Donna K. Harman, editor, *The Second Text Retrieval Conference (TREC-2)*, pages 75–84, National Institute of Standards and Technology, March 1994. Proceedings available as NIST Special Publication 500-215.

[Cro93] W. Bruce Croft. The University of Massachusetts TIPSTER project. In Donna K. Harman, editor, *The First Text Retrieval Conference (TREC-1)*, pages 101–105, National Institute of Standards and Technology, March 1993. Proceedings available as NIST Special Publication 500-207.

[FS94] Edward A. Fox and Joseph A. Shaw. Combination of multiple searches. In Donna K. Harman, editor, *The Second Text Retrieval Conference (TREC-2)*, pages 243–252, National Institute of Standards and Technology, March 1994. Proceedings available as NIST Special Publication 500-215.

[Har93] Donna Harman. Document detection summary of results. In *Proceedings of the TIPSTER Text Program (Phase 1) Workshop*, pages 33–46, Morgan Kaufmann, September 1993. ISBN:1-55860-337-9.

[HC93] David Haines and W. Bruce Croft. Relevance feedback and inference networks. In Robert Korfhage, Edie Rasmussen, and Peter Willett, editors, *Proceedings of the Sixteenth International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2–11, June 1993.

[TC91] Howard Turtle and W. Bruce Croft. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems*, 9(3):187–222, July 1991.

[Tur90] Howard Turtle. *Inference Networks for Document Retrieval.* PhD thesis, Computer Science Department, University of Massachusetts, Amherst, MA 01003, 1990. Available as COINS Technical Report 90-92.

[Tur94] Howard Turtle. Natural language vs. Boolean query evaluation: a comparison of retrieval performance. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proceedings of the Seventeenth Annual International Conference on Research and Development in Information Retrieval*, pages 212–221, Springer-Verlag, London, July 1994.

Latent Semantic Indexing (LSI): TREC-3 Report

Susan T. Dumais
Bellcore
445 South St.
Morristown, NJ 07960
std@bellcore.com

1. Abstract

This paper reports on recent developments of the Latent Semantic Indexing (LSI) retrieval method for TREC-3. LSI uses a reduced-dimension vector space to represent words and documents. An important aspect of this representation is that the association between terms is automatically captured, explicitly represented, and used to improve retrieval.

We used LSI for both TREC-3 *routing* and *adhoc* tasks. For the *routing* tasks an LSI space was constructed using the training documents. We compared profiles constructed using just the topic words (no training) with profiles constructed using the average of relevant documents (no use of the topic words). Not surprisingly, the centroid of the relevant documents was 30% better than the topic words. This simple feedback method was quite good compared to the routing performance of other systems. Various combinations of information from the topic words and relevant documents provide small additional improvements in performance. For the *adhoc* task we compared LSI to keyword vector matching (i.e. using no dimension reduction). Small advantages were obtained for LSI even with the long TREC topic statements.

2. Overview of Latent Semantic Indexing

Latent Semantic Indexing (LSI) is a variant 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 MatchPlus system described by Gallant et al. (1992) is similar 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 (Caid, Dumais and Gallant, in press). More recently, a number of other groups have developed corpus-based association or similarity thesauri for use in automatic query expansion (e.g., Jing and Croft's (1994) PhraseFinder; Strzalkowski's (TREC-3) conceptual hierarchy; or Qui and Frei's (1993) similarity thesaurus). As with LSI and MatchPlus, the idea in all these systems is to discover and exploit corpus-specific inter-item associations to improve retrieval.

3. LSI and TREC-3

We used the previous TREC conferences 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 TREC collection. (See Dumais, 1993, 1994 for details.) We were able to compute the 200-300 largest singular triples of 75k docs x 90k words matrices without numerical or convergence problems on a standard Sparc10 workstation. Because of these limits on the size of the matrices we could handle, we divided the documents into separate subcollections (TREC-1) or computed the SVD of only a sample of documents (TREC-2). For TREC-3 we used the same sampling approach we tried last year.

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 572 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.

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.

3.2 SVD analysis

The "ltc" matrix described above was used as input to the SVD algorithm. The SVD program takes the ltc 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 350 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.

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 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 38175 unique documents with relevance judgements. We used both relevant and non-relevant documents. The SVD analysis was computed on the relevant 38175 document x 78746 term subset of the above ltc matrix, containing 8869743 non-zero cells. A 346 reduced-dimension approximation took 22 hrs of CPU time to compute on a Sparc10 workstation. This 346-dimension representation was used for matching the profiles to the new documents in the routing experiments.

For the **adhoc** experiments, we took a random sample of 70000 documents. A reduced-dimension SVD approximation was computed on a 69997 document x 82968 term matrix (7666044 non-zeros). A 199-dimension approximation was computed and 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 was 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. For TREC, however, the scaling is based on less than 10% of the total corpus.

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 and this is time consuming for large databases. It is, however, straightforward to split this matching across several machines or to use parallel hardware since all documents are independent.)

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 18-20 hours for a 60k x 80k matrix.)

3.4 TREC-3: 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. The lsir2 submission was judged. 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 215 relevant documents per topic, with a range of 25 (topic 140) to 742 (topic 109). Note that this method makes **no** use of the topic words. (Cooper, Chen and Gay (TREC-3) have also described a method which makes minimal use of the query terms when enough feedback is available.) 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. Although it is not implemented as such, the lsir2 method can be thought of a kind of "massive query expansion", which other groups have found quite useful for TREC routing tasks. Recall that each LSI document is located at the weighted average of its constituent terms. The lsir2 routing vector is at the centroid of all relevant documents - that is, at the centroid of all terms in all relevant documents.

The topic and reldocs filters provide two extreme 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 346-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 346-dimensions. The 1000 best matching documents for each filter were submitted to NIST for evaluation. The lsir2 run was judged.

3.4.1 Results

The main routing results are shown in Table 1. The two submitted runs, lsir1 and lsir2, 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 30% (.3737 vs. .2880). Users would get an average of 2 additional relevant documents in the top 10 returned using the lsir2 method for filtering (6.7 vs. 4.6). We suspect that this would be a noticeable improvement for end users and would provide a good starting place for feedback methods. Note, however, that the 30% difference between lsir1 and lsir2 is probably an overestimate of the benefits of lsir2 since some of the lsir1 documents were not judged (1241 of the 10000 top-200 lsir1 documents were not judged).

We also performed the same routing experiment in TREC-2, with amazingly similar results. In TREC-2, lsir2 was 31% better than lsir1 (.3442 vs. .2622), and compared to other systems, lsir2 was better than the median for 40 of the 50 topics. Thus, the LSI method of representing routing profile information as the centroid of known relevant documents appears to be quite robust and useful. As noted above, our centroid method is related to the massive query expansion method used successfully by Buckley and others, and to Cooper et al.'s use of relevance-associated stems. Because each document is represented as a vector in LSI space, our implementation is quite efficient and does not involve explicit term-expansion (since this has already been done in creating the reduced-dimension LSI representation).

Table 1: TREC-3 LSI Routing Results

	lsir1 (topic wds)	lsir2 * (rel docs)	best(r1,r2) (best)	r1+r2 (sum)	.75r1+ .25r2 (sum)	.25r1+ .75r2 (sum)
Rel_ret	6252	6878	7058	7078	6930	7010
Prec at 10	.4620	.6720	.6500	.6840	.5540	.6500
Prec at 100	.3532	.4544	.4520	.4400	.4118	.4590
Avg prec	.2880	.3737	.3963	.3792	.3561	.3827
R-prec	.3386	.3950	.4266	.4054	.3997	.4007
LSI >= Median	21 (4)	41 (10)	42 (13)	37 (2)	32 (4)	40 (4)
LSI < Median	29 (2)	9 (0)	8 (0)	13 (0)	18 (0)	10 (0)

Table 1: LSI Routing Results. Comparison of topic words (lsir1) vs. relevant documents (lsir2) as routing filters. The * indicates the judged run.

Compared to other TREC-3 systems, LSI does quite 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 (Pr at 100) for 41 of the 50 topics, and has the best score for 10 topics. The lsir2 results are especially good at low recall. 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.

We also examined the best that one could do using the r1 and r2 vectors. For each query we chose *either vector r1 or r2*, depending on which had the best average precision for that query. Fourteen of the topics were represented by the lsir1 vector and the remaining 36 by the the lsir2 vector. These results are shown in the third column of Table 1, labeled **best(r1,r2)**. Average precision for best(r1,r2) is 6% better than lsir2. These scores are at or above the median performance for 42 of the 50 topics and the best for 13 topics.

The lsir1 and lsir2 runs provide baselines against which various combinations of query information and relevant document information can be measured. We 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 topic and used this as the new profile vector. The results of this analysis are shown in the fourth 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 one topic 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.)

We examined two other combinations of r1 and r2, varying the contributions of the individual vectors from .75 to .25. Neither combination was as good as the average. These results are shown in the last two columns of Table 1.

The r1+r2 methods 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 produces improvements that are comparable to those obtained by Buckley et al. (1994, TREC-3) using more traditional relevance feedback methods. Average precision for the r1+r2 method is 32% better than for lsir1 which used only the topic words (.3792 vs. .2880), and this is somewhat better than the 24% improvement reported by Buckley, et al. (TREC-3) for their richest routing query expansion method (.3699 vs. .2985).

The above combination methods could be described as *query vector combinations*. In all cases the filter is a single vector. We have also started to look at combination methods that use multiple filters for each query, but do not have results to report at this time. One method involves *data fusion* in which the results of the lsir1 and lsir2 matches are combined in various ways. A related method involves *query splitting*. We have previously conducted experiments using a relevance density method for cumulating information from several separate vectors for each query and

would like to apply this to TREC. (See Kane-Esrig et al., 1991 or Foltz and Dumais, 1992, for a discussion of multi-point interest profiles in LSI.)

Finally, we are beginning to look in detail at the successes and failures of the LSI system. LSI does quite **well** on some queries:

107 (Japanese Regulation of Insider Trading)
108 (Japanese Protectionist Measures)
113 (New Space Satellite Applications)
120 (Economic Impact of International Terrorism)
125 (Anti-Smoking Actions by the Government)
138 (Iranian Support for Lebanese Hostage-takers)
140 (Political Impact of Islamic Fundamentalism)

and quite **poorly** on others:

109 (Find Innovative Companies)
115 (Impact of the 1986 Immigration Law)
149 (Industrial Espionage)

It is not entirely clear what distinguishes between these topics. 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.

3.5 TREC-3: Adhoc experiments

We submitted two sets of adhoc queries - lsia0mf and lsia0mw20f. The lsia0mw20f run was judged. The same SVD analysis was used for both runs. The SVD was based on a random sample of 70k of the 752k documents from CD-1 and CD-2. The SVD was computed on this sample and the remaining documents were "folded in". The lsia0mf run is a baseline. The lsia0mw20f run is the same, but omits documents that have fewer than 20 characters from the returned list. The results from these two runs are shown in the first two columns of Table 2.

The difference between the two lsia0 runs is not large. Omitting short documents never hurt performance and improved it substantially on two topics (156 and 187). In terms of absolute levels of performance, both lsia0 runs are a little below average. Performance does not deviate much from the median.

We also compared our LSI runs to a SMART run using exactly the same pre-processing. For this run we used the same term-document matrix that LSI begins with but we did not do any dimension reduction. These results are shown in the third column of Table 2. The advantage of LSI over

Table 2: TREC-3 Adhoc Results

	lsia0mf	lsia0mw20f *	smart
Rel_ret	6045	6090	5857
Avg prec	.2325	.2393	.2220
Pr at 100	.3130	.3246	.3084
Pr at 10	.4340	.4520	.4040
R-prec	.3030	.3071	.2932
Q >= Median	18	19	18
Q < Median	32	31	32

Table 2: LSI Adhoc Results. The * indicates the judged run.

traditional vector matching is about 5% for these queries. This is smaller than the advantages observed for other test collections. However, as we and others have previously noted (Dumais, 1994; Jing and Croft, 1994; Lu and Keefer, TREC-3; Voorhees, 1994), the TREC topics are quite long and smaller advantages for query expansion methods are to be expected. The average TREC-3 query had 35 content words in it. Based on TREC-2 results, we would expect larger advantages over keyword matching if the queries were shorter, as real adhoc queries typically are.

In both TREC-2 and TREC-3 our SMART runs were worse than those reported by Buckley et al., Fuhr et al., or Voorhees. This is because we used slightly different pre-processing options, non-optimal weightings ("ltc" rather than "ltn" for documents), did not use phrases, and used only words occurring in more than 4 documents (for comparability with the LSI analyses). For TREC-3, for example, our .2220 average precision is 19% worse than Voorhees et al.'s baseline using lnc document weights (.2643) and is 28% worse than Buckeley et al.'s baseline using both lnc and phrases (.2842). Thus, we believe that we could improve the absolute level of performance in all our conditions by using phrases and selecting optimal document weights. In addition, the LSI adhoc analysis used only 199 dimensions, and this is probably too few for good performance with large, diverse corpora (see section 4.2.1 below).

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

In LSI retrieval query vectors are compared to *every* document. 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 process is linear in the number of documents in the We know of no practical and efficient algorithms for finding nearest neighbors in 200- or 300-dimension vector spaces. Methods like kd-trees which work well in 2 or 3 dimensions are not very helpful for more than 10 dimensions.

We are exploring 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 substantially 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.

4.2.1 Number of Dimensions

One important variable for LSI retrieval is the number of dimensions in the reduced dimension space. In previous experiments we 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-3 routing performance using *fewer dimensions* than the 346 reported above and consistently found worse performance for both the total number of relevant documents returned and average precision measures -

346 dimensions (7078, .374)
300 dimensions (6831, .371)
250 dimensions (6751, .367)

We could easily improve performance simply by increasing the number of dimensions some. The same is also true for the adhoc runs which used only 199 dimensions and could be improved substantially by increasing the number of dimensions in the LSI space.

4.2.2 Term Weighting

We still need to experiment with different term weighting methods. For the routing and adhoc experiments we used SMART's "ltc" weighting for both documents and queries (i.e., ltc.ltc). Buckley and others have found that alternative weightings ("lnc" for documents, lnc.ltc) are more effective in TREC for the standard vector method. We suspect that LSI would benefit from optimal weighting as well.

4.2.3 Document Sampling

The size of the SVD we can compute on standard workstations is still limited. Computing the SVD of a 60k x 80k matrix takes about 18-20 hours of CPU time and 250 meg of memory. Larger SVD analyses are just not practical at this time. We have used two approaches to overcome this limitation - in TREC-1 we divided the collection into several subcollections,

and in TREC-2 and TREC-3 we computed the SVD of only a sample of documents. For the routing task, we used a sample chosen from the training data, and this worked quite well. For the adhoc task, we chose a small random sample (70k of 752k), and this was not as successful. The 70k sample represents less than 10% of the documents and this may not be sufficient to accurately represent the variability of topics encountered in the adhoc queries. We would like to try larger samples.

4.2.4 Retrieval failures

In order to better understand retrieval performance we will 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. We have only examined these retrieval failures for a few topics so far.

4.2.4.1 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. 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 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 the 50 routing queries and 21 of the 50 adhoc queries contain some negation in the topic statement. We do nothing about this. In fact, we include all the negated terms in the query. The use of logical connectives is another example of inappropriate query processing. 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.

For the TREC routing tasks there are additional sources of false alarms that would likely be minimal in real routing applications. LSI often returned many documents on the same topic that were all judged to be not-relevant. In a real routing application, one would get immediate feedback that a particular topic was not of interest, and presumably subsequent similar documents would not be returned. In addition, for some topics there appears to be a lack of correspondence between judgements for training and test documents. Documents on what appear to be the same topic were judged relevant in the training set but not relevant in the test set. This may be the result of different people making the relevance judgements at the two points in time, or it may simply reflect slightly changing criteria over time.

4.2.4.2 Misses. For this analysis we examine a random subset of relevant articles that were not in the *top 1000* returned by LSI. Many of the relevant articles found by other groups were fairly highly ranked by LSI, but there were also some notable failures that would be seen only by the most persistent readers.

Most of the misses 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 or divisions of documents into smaller sections should help in identifying less central themes in documents. This should be especially easy in some cases where documents are tagged to indicate separate news briefs.

Some misses were also attributable to poor text (and query) pre-processing and tokenization.

4.2.4.3 Examples. Many of these ideas are illustrated more concretely by considering some specific examples of retrieval failures. We examined routing topics for which LSI (the lsir2 run) does poorly relative to other TREC groups on the average precision measure. For two of these topics (109 and 149), we miss more than 50% of relevant documents found by other groups. For other topics, we find all relevant documents but precision is poor. Finally, for some topics both retrieval failures and poor precision contribute to poor performance.

Topic 149 (Industrial Espionage) - LSI misses 224 of 310 (72%). There are many false alarms (e.g., 21 of

the top 25) which also contributes to the poor precision. Several of the false alarms appear to be relevant to my eye (e.g., Niedorf distributing proprietary documents, a few directly about industrial espionage). Others are about computer security (information privacy, hacking, pirated software, etc.) but do not involve a specific company. Finally, several are about counter intelligence and government security breaches, again not aimed at a specific company. Almost all of the missed articles involve the insider trading cases of Boesky, Milken, et al. LSI's performance on this topic mirrors the training documents - most were about computer fraud, Soviet spying, and only a few about Milken and Boesky.

Topic 109 (Find Innovative Companies) - LSI misses 707 of 1191 (59%). This topic involves the matching of 5 specific company names. LSI does not do literal string matching and really suffered here. Many of the misses involved the company being mentioned in an article that was primarily about something else (e.g., a MIPS stock holder listed in a chart). It is trivial to solve this particular problem with the appropriate string matching tools.

Topic 115 (Impact of the 1986 Immigration Law) - LSI misses none (of 56), but precision is poor. Most of the false alarms are about immigration problems but do not mention the 1986 Immigration Act. This lack of specificity is a fairly typical kind of LSI failure.

Topic 142 (Impact of Government Regulated Grain Farming on International Relations) - LSI misses 213 of 638 (33%). The misses for this topic were generally in the top 5000, although this is no consolation to an end user. The false alarms result from the same lack of specificity noted for the previous topic. The irrelevant items near the top of the list tended to be about domestic (vs international) effects of farm regulations, or about farm exports but not also about regulation. One might be able to increase the likelihood that many concepts contribute to the match by limiting the contribution of any one term to the overall similarity.

Topic 144 (Management Problems at the United Nations) - LSI misses none of 8, but precision is poor. This topic seems to suffer from inconsistent relevance judgements. Several of the LSI false alarms appear to me to be quite similar to two of the relevant documents as well as training documents (U.S. failing to pay U.N. dues because of inefficiencies and budget over runs). In addition, half to the relevant documents are questionable in my mind. There is just not much right on target here.

Topic 121 (Death from Cancer) - LSI miss 127 of 258 (49%), although precision is reasonable. Almost all the misses come from the SJM where obituaries are formatted as long lists rather than individual articles for each person. LSI's centroid representation is problematic here. This could easily be solved by splitting articles on multiple topics, especially those that are tagged as such. All the top false alarms here are about death from cancer, but no specific type of cancer is mentioned.

4.3 Future research

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, one for each subcollection, for the TREC-1 experiments. For TREC-2 and TREC-3 we used a single scaling based on a small sample of the CD-1 and CD-2 documents. Although the sampling has worked quite well, we believe that there are many practically reasons for using subscaling based on topically coherent collections. First, we believe that subscalings will result in more dimensions being devoted to discriminating among objects. Many fine-grained discriminations can be made among computer documents if 200 dimensions are used in a Ziff subscaling. In a larger analysis, many fewer dimensions will be devoted to distinguishing among computer related topics. Both term weights and the LSI space will be based on topically coherent subcollections. Second, for distributed or rapidly changing collections, separate analyses may be necessary. We have not yet had time to compare the subscaling and sampling results but would like to examine this issue in more detail.

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. Since the computational complexity of this

method increased with the number of vectors, query splitting should be used only in cases where relevant documents or query terms are not too similar to each other.

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. References

[1] Berry, M. W. Large scale singular value computations. *International Journal of Supercomputer Applications*, 1992, *6(1)*, 13-49.

[2] Buckley, C., Salton, G. and Allan, J. The effect of adding relevance information in a relevance feedback environment. In *Proceedings of SIGIR'94*, 1994, 292-300.

[3] Buckley, C., Salton, G., Allan, J. and Singhal, A. Automatic query expansion using SMART: TREC 3. To appear in: *Proceedings of TREC-3*.

[4] Caid, W. R., Dumais, S. T. and Gallant, S. I. Learned vector-space models for document retrieval. To appear in: *Information Processing and Management.*

[5] Cooper, W., Chen, A. and Gey, F. Experiments in the probabilistic retrieval of full text documents. To appear in: *Proceedings of TREC-3.*

[6] 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.

[7] 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.

[8] Dumais, S. T. Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments and Computers*, 1991, *23(2)*, 229-236.

[9] 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.

[10] Dumais, S. T. Latent Semantic Indexing (LSI) and TREC-2. In D. Harman (Ed.) *The Second Text REtrieval Conference (TREC-2).* NIST special publication 500-215, 105-116.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] Jing, Y. and Croft, W. B. An association thesaurus for information retrieval. In *RIAO 4 Conference Proceedings*, 1994.

[16] 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.

[17] Lu, X. A. and Keefer, R. Query expansion/reduction and its impact on retrieval effectiveness. To appear in: *Proceedings of TREC-3.*

[18] Qiu, Y. and Frei, H. P. Concept based query expansion. In *Proceedings of SIGIR'93*, 1993, 160-169.

[19] Salton, G. and McGill, M.J. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[20] Strzalkowski, T., Carballo. J. P., Marinescu, M. Natural language information retrieval: Trec-3 report. To appear in: *Proceedings of TREC-3.*

[21] Voorhees, E. Query expansion using lexical-semantic relations. In *Proceedings of SIGIR'94*, 1994, 61-70.

[22] Voorhees, E., Gupta, N. K., and Johnson-Laird, B. The collection fusion problem. To appear in: *Proceedings of TREC-3.*

6. 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.

Query Expansion/Reduction and its Impact on Retrieval Effectiveness

X. Allan Lu and Robert B. Keefer
Mead Data Central, Inc.
P.O. Box 933
Dayton, OH 45401
(alan,robk)@meaddata.com

1.0 Introduction

Query expansion should help improve information retrieval effectiveness. Reported studies [1-5] using the TREC data generally support this position, though earlier studies [6-8] using smaller test databases did not obtain any conclusive results. Three important research questions remain open. First, assuming that a thesaurus will be used, should a general or data specific thesaurus be used in query expansion? Second, at what point does query expansion cease to add value? Finally, to what degree does query expansion help improve retrieval effectiveness?

A general thesaurus has been found to have little effect on information retrieval results, and may even cause negative results. Prior studies [1-5] have observed that a thesaurus derived from the data on which the retrieval tasks will be performed tends to add useful terms to the query, and as a result, tends to improve retrieval effectiveness. To address this question an associative thesaurus was developed using the TREC data (disks 1 and 2) for our TREC3 query processing.

The second question, at what point does query expansion cease to add value, is a real challenge to those studying manual query expansion. The primary concern is that excessive expansion may dilute the original query and result in the retrieval of nonrelevant documents. The answer to this problem is beyond the scope of this report.

A proposed alternative to manual query expansion is automatic query expansion. This alternative is not a very viable option in the on-line service environment because automatic query expansion largely excludes the user from the query formulation process. This may cause the user some confusion since he does not know how the system has modified his query. Another alternative would be to combine manual and automatic query expansion. The user is an intelligent being, not merely a mechanical receptor of data. The user's intelligence can be channeled into the information retrieval system to strengthen the quality of the query and increase its effectiveness.

The primary focus of this report will be on the degree to which query expansion helps improve retrieval effectiveness. Previously published studies of TREC data [1-5] showed improvements ranging from 0% to 20%. We view these findings as inconclusive due to

the characteristics of the TREC queries. To those working in the on-line information service industry, the TREC queries are unusually long, structured and descriptive. In other words, the current users of the commercial on-line services rarely, if ever, type in such lengthy natural language queries. To illustrate this point, Table 1 compares the TREC2 and TREC3 *ad hoc* queries to query statistics gathered over a 3 day period by FREESTYLE™, the non-Boolean on-line information retrieval service provided by Mead Data Central, Inc.

Query	Average Length	Longest	Shortest
TREC2	170	319	89
TREC3	105	180	49
FREESTYLE™	7	32	1

TABLE 1. TREC Ad Hoc Queries and FREESTYLE™ Query size (# words)

These basic descriptive statistics indicate the significant difference in query size between the TREC queries and those used in FREESTYLE™. Assuming that these statistics show an order of magnitude difference and that the FREESTYLE™ queries are typical of those processed by most on-line information services, two basic questions are raised. How should the TREC research results be interpreted and generalized? How much can query expansion help improve retrieval effectiveness? To address these questions, we designed and performed a series of experiments using four sets of shorter TREC2 *ad hoc* queries. A subset of these experiments were repeated in our TREC3 efforts.

2.0 TREC2 Query Reduction Experiments

The TREC2 topics (including summary and definition fields) were used as the ideal queries (baseline). Then four different generations of modifications were made to this baseline. First, the summary and definition fields were deleted. The second experiment had the concept field deleted from the base queries. The third experiment was conducted using the recomposed, short version of the queries. This version of the topics are queries consisting of one to three sentences. This process was completed by a professional data analyst who read the entire topic and then composed a shorter version of the topic. The goal was to condense the TREC2 topics down to a size that was closer to that of the FREESTYLE™ queries. The final set of queries contained only the description field of the TREC2 topics. Table 2 compares the sizes of the five generations of topics studied. The retrieval results from our experiments with the short topics may shed some light on the potential risks involved in generalizing the TREC results.

Query	Average Length	Longest	Shortest
Topics	170	319	89
Topics-Sum-Def	138	297	62
Topic-concept	135	273	64
ShortTopic	33	63	18
DescTopic	19	42	6

NOTE: The "-" should be viewed as deletion

TABLE 2. TREC2 Query Reduction

We used the SMART system developed by Cornell University [9] to perform our TREC2 experiments. This system was chosen for its general availability and acceptance over the past 30 years. Specifically, we used the default version of SMART. We did not utilize a customized stop list, phrase capability, sophisticated stemming, or thesaurus of any kind. We used a subset of automatic indexing strategies available in SMART to simulate different indexing systems. The selected indexing strategies are the top 10 ranked according to the 11-point averages in TREC2 (i.e., lnc.ltc, lnc.atc, lnc.mtc, ltc.ltc, mtc.ltc, ltc.atc, ltc.lnc, mtc.atc, ltc.mtc, ltc.nnn [10]). With this design, we could focus our research efforts on the two factors of interest: the size of the queries and the indexing/retrieval methods.

2.1 Impact of Query Size on Retrieval Effectiveness

Accepting the hypothesis that query expansion is beneficial to information retrieval, we may also hypothesize that query reduction would be detrimental to retrieval results. In other words, by demonstrating that query reduction has a negative effect on the retrieval results, we may infer that query expansion would have a positive effect on the retrieval results.

Associated with the query reduction hypothesis is the hypothesis that query reduction has a different degree of negative effect on individual indexing and retrieval systems. In other words, some information retrieval systems that perform effectively with large queries may not perform as well with small queries. In addition, the deterioration rates among the particular indexing and retrieval systems are different in processing a series of gradually smaller queries.

The three tables in this section summarize the experimental results. The numbers were generated by the SMART system based on the top 1000 retrieved documents for each TREC2 test query. The measure in Table 3, 11-point recall average precision, is a recall-biased measure. The measures in Tables 4 and 5 are precision at the fixed rank points, top 5 documents and top 15 documents, respectively, and are precision-biased measures. The deterioration rates, computed using the Topic column as the baseline, are in parentheses next to each precision measure.

In Table 3, deleting the summary and definition fields did not have any noticeable effect on the measure of 11-point recall average. However, when the concept field was deleted, retrieval effectiveness was reduced by an average of 23%. Use of the recomposed, short

TREC2 topics reduced the results by another 10%. Finally, use of the description field as the query reduced the results by still another 18%. The results in Table 3 support the hypothesis that a shorter query is strongly related to significantly lower retrieval performance.

Indexing	Topic	Top-Sum-Def	Topic-concept	ShortTopic	DescTopic
lnc.ltc	0.3405	0.3413(0%)	0.2611(-23%)	0.2208(-35%)	0.1526(-55%)
lnc.atc	0.3199	0.3208(0%)	0.2527(-21%)	0.2093(-35%)	0.1536(-52%)
lnc.mtc	0.3099	0.3127(1%)	0.2464(-20%)	0.2142(-31%)	0.1504(-51%)
ltc.ltc	0.3040	0.3055(0%)	0.2319(-24%)	0.2099(-31%)	0.1588(-48%)
mtc.ltc	0.2953	0.2958(0%)	0.2187(-26%)	0.1921(-35%)	0.1511(-49%)
ltc.atc	0.2952	0.2957(0%)	0.2278(-23%)	0.2093(-29%)	0.1583(-46%)
ltc.lnc	0.2858	0.2926(2%)	0.2057(-28%)	0.1790(-37%)	0.1108(-61%)
mtc.atc	0.2849	0.2839(0%)	0.2168(-24%)	0.1902(-33%)	0.1527(-46%)
ltc.mtc	0.2750	0.2797(2%)	0.2181(-21%)	0.2045(-26%)	0.1567(-43%)
ltc.nnn	0.2662	0.2760(4%)	0.2002(-25%)	0.1727(-35%)	0.1081(-59%)
Average	0.2977	0.3004(1%)	0.2279(-23%)	0.2002(-33%)	0.1453(-51%)

TABLE 3. Impact of Query Reduction: TREC2 11-Point Recall Average Precision

Analysis of the precision biased measures in Tables 4 and 5 produces similar observations to those in Table 3. The absence of summary and definition fields did not have observable impact on either of the two precision results. However, by removing the concept field or by using the two shortened topics, both of the precision percentages showed significant deterioration. Tables 4 and 5 further support the hypothesis that query reduction is detrimental to retrieval effectiveness.

Indexing	Topic	Top-Sum-Def	Topic-concept	ShortTopic	DescTopic
lnc.ltc	0.6200	0.6040(-3%)	0.5320(-14%)	0.3960(-36%)	0.3600(-42%)
lnc.atc	0.5760	0.5640(-2%)	0.5160(-10%)	0.4360(-24%)	0.3480(-40%)
lnc.mtc	0.5320	0.5520(4%)	0.4800(-9%)	0.4040(-24%)	0.3520(-34%)
ltc.ltc	0.5200	0.5240(1%)	0.4320(-17%)	0.3960(-24%)	0.2840(-45%)
mtc.ltc	0.5520	0.5480(-1%)	0.4880(-12%)	0.4480(-19%)	0.3920(-29%)
ltc.atc	0.5240	0.5160(-2%)	0.4560(-12%)	0.3920(-24%)	0.2880(-45%)
ltc.lnc	0.5200	0.5320(2%)	0.4480(-14%)	0.4000(-23%)	0.2920(-44%)
mtc.atc	0.5480	0.5440(-1%)	0.4880(-11%)	0.4440(-19%)	0.3840(-30%)
ltc.mtc	0.4720	0.4760(1%)	0.3960(-16%)	0.3760(-20%)	0.2800(-41%)
ltc.nnn	0.5000	0.4440(-11%)	0.4040(-19%)	0.3760(-25%)	0.2880(-42%)
Average	0.5364	0.5304(-1%)	0.4640(-13%)	0.4068(-24%)	0.3268(-39%)

TABLE 4. Impact of Query Reduction: TREC2 Precisions at top 5 Documents

Indexing	Topic	Top-Sum-Def	Topic-concept	ShortTopic	DescTopic
lnc.ltc	0.5680	0.5667(0%)	0.4933(-13%)	0.4147(-27%)	0.3307(-42%)
lnc.atc	0.5653	0.5693(1%)	0.4853(-14%)	0.4133(-27%)	0.3387(-40%)
lnc.mtc	0.5213	0.5240(0%)	0.4573(-12%)	0.3947(-24%)	0.3173(-39%)
ltc.ltc	0.4947	0.5013(1%)	0.4107(-17%)	0.3800(-23%)	0.2800(-43%)
mtc.ltc	0.5240	0.5160(-2%)	0.4507(-14%)	0.4227(-19%)	0.3493(-33%)
ltc.atc	0.4880	0.4893(0%)	0.4240(-13%)	0.3800(-22%)	0.2827(-42%)
ltc.lnc	0.5147	0.5240(2%)	0.4067(-21%)	0.3520(-32%)	0.2440(-52%)
mtc.atc	0.5027	0.5013(0%)	0.4480(-11%)	0.4213(-16%)	0.3533(-30%)
ltc.mtc	0.4587	0.4653(1%)	0.3787(-17%)	0.3547(-23%)	0.2800(-39%)
ltc.nnn	0.4440	0.5080(14%)	0.3680(-17%)	0.3320(-25%)	0.2347(-47%)
Average	0.5081	0.5165(2%)	0.4323(-15%)	0.3865(-24%)	0.3010(-40%)

TABLE 5. Impact of Query Reduction: TREC2 Precisions at top 15 Documents

2.2 Impact of Query Size on Performance Consistency

Associated with the query reduction hypothesis is the hypothesis that query reduction has a varying degree of negative effect on individual indexing and retrieval systems. In other words, some information retrieval systems that perform effectively with large queries may not perform as well with small queries. To test the hypothesis that information retrieval systems perform inconsistently on different sized queries, a series of simple correlation analyses was performed using the results in Tables 3, 4 and 5. If an information retrieval system performs consistently over an array of gradually smaller queries, performance on large queries should be a good predictor of performance on smaller queries as measured by linear correlation. Stated another way, one could expect a high linear coefficient of determination, i.e. high R^2 value. Conversely, performance of an information retrieval system would be considered inconsistent if it produced a low R^2 value. To illustrate this point, the weak correlations between the retrieval performance of the topic and the recomposed topics in Table 6 suggest that the selected indexing/retrieval systems did not consistently handle the large and small TREC2 queries. The results, however, are preliminary and to some extent inconclusive due to the small sample size and the SMART system constraints.

Correlation	11-Point Avg	Precision at top 5	Precision at top 15
Topic vs. Top-Sum-Def	R^2=0.97	R^2=0.79	R^2=0.75
Topic vs. Top-Concept	R^2=0.85	R^2=0.90	R^2=0.88
Topic vs. ShortTopic	R^2=0.53	R^2=0.25	R^2=0.57
Topic vs. DescTopic	R^2=0.23	R^2=0.47	R^2=0.43

TABLE 6. TREC2: Linear Correlation analysis of indexing/retrieval consistency

3.0 TREC3 Query Expansion and Reduction Experiments

One of the two submitted TREC3 entries, ASSCTV1, was created to further test the two hypotheses that query expansion in general is beneficial to information retrieval and that

different retrieval systems receive different levels of help from query expansion. Indexing and retrieval conditions identical to the TREC2 query reduction experiments were maintained, i.e. the default version of SMART and the same set of automatic indexing methods were used. Unlike the TREC2 experiments, the TREC3 *ad hoc* queries were expanded using an associative thesaurus. The expanded queries were loaded into the SMART system to retrieve the top 1000 ranked documents. We then waited for the query relevance information in order to conduct further experiments.

Three additional experiments were performed: one using the TREC3 topics; one using the recomposed, shorter TREC3 queries; and one using only the description field of the TREC3 queries. Table 7 provides the statistics of the original, expanded and recomposed TREC3 *ad hoc* queries. The Expanded Topics results shown in Table 7 reflect an actual query expansion experiment, as opposed to a query reduction experiment. In expanding the TREC3 topics, we tried to maintain the rule to not expand to more than 150% of the original topic size. This rule is merely a guide to prevent over expansion.

Query	Average Length	Longest	Shortest
TREC3 Topics	105	180	49
Expanded Topics	135	194	73
ShortTopics	24	41	16
DescTopics	23	43	9

TABLE 7. TREC3 Query Expansion and Reduction

Tables 8 through 10 describe the impact of query expansion and reduction on the TREC3 *ad hoc* retrieval results. Note that the baseline in these three tables is composed of the searches using the original TREC3 *ad hoc* topics. The heading "Expanded Topic" is for the expansion experiment; the headings "ShortTopic" and "DescTopic" are for the two reduction experiments. In Table 8 the expanded queries enhanced the recall-oriented performance an average of 33%, while the two sets of short queries reduced the performance an average of 34% and 39%, respectively. Moreover, Tables 9 and 10 show the expanded queries and the short queries enhanced or reduced the precision-oriented performances an average of approximately 30%.

Indexing	Topic	Expanded Topic	ShortTopic	DescTopic
lnc.ltc	0.2930	0.3685(26%)	0.1775(-39%)	0.1738(-40%)
lnc.atc	0.2835	0.3551(25%)	0.1787(-37%)	0.0915(-68%)
lnc.mtc	0.2624	0.3375(29%)	0.1737(-34%)	0.1708(-35%)
ltc.ltc	0.2343	0.3181(36%)	0.1564(-33%)	0.1527(-35%)
mtc.ltc	0.2279	0.3001(32%)	0.1674(-27%)	0.1630(-28%)
ltc.atc	0.2261	0.3086(36%)	0.1582(-30%)	0.1557(-31%)
ltc.lnc	0.2067	0.3140(52%)	0.1172(-43%)	0.1149(-44%)
mtc.atc	0.2203	0.2930(33%)	0.1663(-25%)	0.1619(-26%)
ltc.mtc	0.2243	0.2895(29%)	0.1549(-31%)	0.1510(-32%)
ltc.nnn	0.2000	0.2877(44%)	0.1140(-43%)	0.1121(-44%)
Average	0.2379	0.3172(33%)	0.1564(-34%)	0.1447(-39%)

TABLE 8. Impact of Query Expansion/Reduction: TREC3 11-Point Recall Average Precision

Indexing	Topic	Expanded Topic	ShortTopic	DescTopic
lnc.ltc	0.5640	0.7080(26%)	0.3720(-34%)	0.3480(-38%)
lnc.atc	0.5840	0.7000(20%)	0.3840(-34%)	0.1880(-68%)
lnc.mtc	0.4560	0.5520(21%)	0.3560(-22%)	0.3320(-27%)
ltc.ltc	0.3920	0.5680(45%)	0.2400(-39%)	0.1527(-61%)
mtc.ltc	0.5080	0.6720(32%)	0.3760(-26%)	0.3840(-24%)
ltc.atc	0.3960	0.5920(50%)	0.2560(-35%)	0.2600(-34%)
ltc.lnc	0.3920	0.6360(62%)	0.2360(-40%)	0.2160(-45%)
mtc.atc	0.4600	0.6400(39%)	0.3800(-17%)	0.3880(-16%)
ltc.mtc	0.3360	0.4600(37%)	0.2480(-26%)	0.2440(-27%)
ltc.nnn	0.3120	0.4600(47%)	0.2200(-30%)	0.2000(-36%)
Average	0.4400	0.5988(36%)	0.3068(-30%)	0.2712(-38%)

TABLE 9. Impact of Query Expansion/Reduction: TREC3 Precisions at top 5 Documents

Indexing	Topic	Expanded Topic	ShortTopic	DescTopic
lnc.ltc	0.5227	0.6253(20%)	0.3333(-36%)	0.3267(-38%)
lnc.atc	0.5280	0.6147(16%)	0.3427(-35%)	0.1947(-63%)
lnc.mtc	0.4480	0.5267(18%)	0.3173(-29%)	0.3120(-30%)
ltc.ltc	0.3827	0.5347(40%)	0.2827(-26%)	0.2733(-29%)
mtc.ltc	0.4587	0.5840(27%)	0.3573(-22%)	0.3533(-23%)
ltc.atc	0.3960	0.5333(35%)	0.2893(-27%)	0.2800(-29%)
ltc.lnc	0.3827	0.5560(45%)	0.2187(-43%)	0.2253(-41%)
mtc.atc	0.4320	0.5800(34%)	0.3640(-16%)	0.3587(-17%)
ltc.mtc	0.3467	0.4573(32%)	0.2747(-21%)	0.2667(-23%)
ltc.nnn	0.3200	0.4493(40%)	0.2067(-35%)	0.2133(-33%)
Average	0.4218	0.5461(29%)	0.2987(-29%)	0.2804(-34%)

TABLE 10. Impact of Query Expansion/Reduction: TREC3 Precisions at top 15 Documents

Correlation analysis was conducted using the results in Tables 8, 9 and 10. Table 11 summarizes the analysis results. Similar to the TREC2 results, the selected indexing/retrieval methods were inconsistent in processing the large and small queries as indicated by the low R^2 values. Note that the "predictor" in Table 11 is the expanded TREC3 topic instead of the original TREC3 topic. These results are preliminary and to some extent inconclusive due to the small sample size and the SMART system constraints.

Correlation	11-Point Avg	Precision at top 5	Precision at top 15
ExpTop vs. Topic	R^2=0.78	R^2=0.69	R^2=0.74
ExpTop vs. ShortTopic	R^2=0.20	R^2=0.29	R^2=0.18
ExpTop vs. DescTopic	R^2=0.21	R^2=0.25	R^2=0.23

TABLE 11. TREC3: Correlation analysis of indexing/retrieval consistency

The associative thesaurus used in these experiments was compiled automatically using the data that was to be used for the *ad hoc* query portion of TREC3. For this particular version of the thesaurus, the computation took approximately 80 clock hours on a shared SUN Sparc 1000 machine. The actual computation time depends on the number of natural language processing tasks to be performed on the data set. The constructed thesaurus contains uncontrolled index terms. A user interface was built for accessing the thesaurus.The associative thesaurus was also used in the query expansion for another TREC3 entry (ASSCTV2) which was an internal study on search engines.

4.0 Summary

The results of the TREC2 and TREC3 query reduction experiments support the concept that reducing queries is detrimental to information retrieval results. Moreover, the results of the TREC3 *ad hoc* query expansion experiments support the concept that expanding queries using a thesaurus derived from the local data tends to improve retrieval effective-

ness. Depending on the length of the original query, the improvement could be very significant. The optimal level of manual query expansion is still an open question. In addition, both sets of experiments suggest that different indexing/retrieval systems perform inconsistently in processing the various sizes of queries. This observation suggests a cautious attitude toward efforts to generalize the current TREC results.

ACKNOWLEDGMNTS: The authors of this paper would like to thank Rita Freese for her analysis work. The hours spent recomposing the queries for this research did not go unnoticed.

REFERENCES:

1. Callan, J. P. and Croft, W. B. "An evaluation of query processing strategies using the TIPSTER collection." Proceedings of ACM SIGIR International Conference on Research and Development in Information Retrieval, pp.347-356, 1993.
2. Evans, D. A. and Lefferts, R. G. "Design and evaluation of the CLARIT-TREC-2 system." The Second Text REtrieval Conference (TREC-2), NIST Special Publication 500-215, Edited by D. Harman, pp.137-150, 1993.
3. Voorhees, E. M. "On expanding query vectors with lexically related words." The Second Text REtrieval Conference (TREC-2), NIST Special Publication 00-215, Edited by D. Harman, pp.223-231.
4. Efthimiadis, E. N. and Biron, P. V. "UCLA-Okapi at TREC-2: Query expansion experiments." The Second Text REtrieval Conference (TREC-2), NIST Special Publication 00-215, Edited by D. Harman, pp.279-289.
5. Buckley, C., Salton, G. and Allan, J. "The effect of adding relevance information in a relevance feedback environment." Proceedings of ACM SIGIR International Conference on Research and Development in Information Retrieval, pp.292-300.
6. Lesk, M. E. "Word-word associations in document retrieval systems." American Documentation, Vol.20, pp.8-36, 1969.
7. Sparck Jones, K. and Barber, E. O. "What makes an automatic keyword classification effective." JASIS, Vol.22, pp.166-175, 1971.
8. Harman, D. "Towards interactive query expansion." Proceedings of ACM SIGIR International Conference on Research and Development in Information Retrieval, pp.?, 1988.
9. Buckley, C. "Implementation of the SMART information retrieval system." Technical Report 85-686, Computer Science Department, Cornell University, Ithaca, New York, May 1985.
10. Salton, G. and Buckley, C. SMART 11.0, Computer Science Department, Cornell University, Ithaca, New York, July 1992.

Improving a Basic Retrieval Method by Links and Passage Level Evidence

Daniel Knaus, Elke Mittendorf, Peter Schäuble
(knaus|mittendorf|schauble)@inf.ethz.ch
Swiss Federal Institute of Technology (ETH)
CH-8092 Zürich, Switzerland

Abstract

A conventional text retrieval method is improved by two additional sources of evidence of relevance: first, by a passage retrieval method that is based on Hidden Markov Models and second, by a so-called link method which was originally developed for hypertext retrieval. The results show that the two additional sources of evidence improve a conventional vector space retrieval method both in a consistent and in a complementary way. The link method has the ability to retrieve relevant documents whose description vectors are not very similar to the query description vector whereas the passage retrieval method has the ability to improve an existing ordering of the documents.

1 Introduction

Our approach is aimed at improving a conventional text retrieval method by means of two additional sources of evidence of relevance. The conventional retrieval method is called the *basic method B*. It consists of a well-known weighting scheme which is sometimes referred to as *lnc.ltc* (Knaus & Schäuble, 1993).

Passage retrieval is used as a second source of evidence of relevance. We used our *passage retrieval method P* which is based on Hidden Markov Models (Mittendorf & Schäuble, 1994). This approach has the advantage that the passages are not restricted to a fixed length. Furthermore, the parameters can be estimated automatically by means of the Baum–Welch reestimation formula.

A so-called *link method L* is used as a third source of evidence of relevance. The link method was originally developed for hypertext retrieval (Frei & Stieger, 1995). Since the TREC documents are not linked, we automatically create appropriate links before applying the hypertext retrieval method.

We combined the three sources of evidence of relevance in an ad-hoc way, i.e. by trying several combinations to find the best combination. When combining several sources of evidence of relevance the scores of the particular sources have to be normalized. We circumvented the normalization problem by using rank numbers. We believe that the problem of combining several sources of evidence of relevance needs further investigations.

2 Multiple Levels of Evidence

2.1 Basic Method

TREC-2 has shown that using a general stoplist, a word normalization algorithm, and an appropriate weighting scheme already leads to fairly good results in terms of retrieval effectiveness (Knaus & Schäuble, 1993). Our basic method for TREC-3 uses the stoplist of the SMART System (Ide & Salton, 1971) consisting of 571 stopwords. The remaining words are normalized by Porter's suffix stripping algorithm (Porter, 1980). Thus, our indexing features consist of Porter reduced words which do not occur in the SMART stoplist. The analysis of all documents of the document collection provides the feature frequencies $\mathrm{ff}(\varphi_i, d_j)$ and the document frequencies $\mathrm{df}(\varphi_i)$. The feature frequency $\mathrm{ff}(\varphi_i, d_j)$ denotes the number of occurrences of the indexing feature φ_i within the document d_j and the document frequency $\mathrm{df}(\varphi_i)$ denotes the number of documents containing the feature φ_i. It should be noticed that the document frequencies are always determined by disk 1 and disk 2 (D1D2) independent of whether they are used for the ad-hoc or the routing task. Hence, query features not occurring in D1D2 have a document frequency $\mathrm{df}(\varphi) = 0$ even though they may occur in disk 3 (D3).

The retrieval function is equivalent to the cosine measure. The indexing method uses *lnc.ltn* feature weighting. This weighting scheme showed the best retrieval effectiveness in our experiments at TREC-2 and it has the advantage that features of routing documents not occurring in the training collection are independent of collection statistics such as inverse document frequencies. Throughout the paper we refer to this method as

the basic method B.

query q
documents d_j, $0 \leq j < n$
indexing features φ_i, $0 \leq i < m$
feature frequencies $\mathrm{ff}(\varphi_i, d_j)$, $\mathrm{ff}(\varphi_i, q)$
document frequencies $\mathrm{df}(\varphi_i)$
normalized inverse document frequencies
$\mathrm{nidf}(\varphi_i) := 1 - \frac{\log(\mathrm{df}(\varphi_i)+1)}{\log(n+1)}$
query feature weights (ltn weighted)
$b_i := (1 + \log(\mathrm{ff}(\varphi_i, q))) * \mathrm{nidf}(\varphi_i)$
document feature weights (lnc weighted)
$a_{ij} := (1 + \log(\mathrm{ff}(\varphi_i, d_j)))$
document lengths
$|\vec{d_j}| := \sqrt{\sum_{i:\varphi_i \in d_j} {a_{ij}}^2}$
Retrieval Status Values
$RSV_B(q, d_j) := \frac{1}{|\vec{d_j}|} * \sum_{i:\varphi_i \in q, d_j} a_{ij} * b_i$

2.2 Link Method

The link method is based on Stieger's hypertext retrieval method (Frei & Stieger, 1995). Hypertext retrieval methods do not only take into account the relevance of a node but also the relevance of its neighbors (Frisse, 1988). The main ideas of Stieger's hypertext retrieval method are the following.

1. We take into account the relevance of a link with respect to the query. Below we will elaborate on what we mean by the relevance of a link.

2. We reduce the hypertext to a substructure consisting only of links that are sufficiently relevant to the query.

3. We reduce the variation of how much the retrieval status value of a node is affected by the retrieval status values of its neighbors.

Since the TREC data does not contain links between the documents we first generate links automatically before applying Stieger's hypertext retrieval method.

We generate a link from document d_j to document d_k if they contain a common phrase λ which satisfies the following three conditions.

1. The common phrase $\lambda = \{\varphi_h, \varphi_i\}$ consists of two indexing features φ_h and φ_i which must occur in both documents d_j and d_k in such a way that only short non-stop words may occur in between where a word is considered as short if it consists of one or two letters. Examples are "exploration of the universe" which becomes "EXPLOR UNIVERS" or "flight spectrometer" which becomes "FLIGHT SPECTROMET".

2. In addition, the two words φ_h and φ_i which constitute the link between d_j and d_k must occur sufficiently many times.

$$\begin{aligned} \mathrm{ff}_{min} &\leq \min\{\mathrm{ff}(\varphi_h, d_j), \mathrm{ff}(\varphi_i, d_j)\} \\ \mathrm{ff}_{min} &\leq \min\{\mathrm{ff}(\varphi_h, d_k), \mathrm{ff}(\varphi_i, d_k)\} \end{aligned}$$

   In this way, we require that the phrase is an important phrase in both documents d_j and d_k.

3. Finally, the document frequency of the phrase λ is within a certain interval.

$$\mathrm{df}_{min} \leq \mathrm{df}(\lambda) \leq \mathrm{df}_{max}$$

This is in accordance with Luhn's ideas for selecting good indexing features (Luhn, 1958). By adding such links to the document collection we obtain a hypertext to which the link method is applied. Subsequently, a link is identified by the corresponding phrase $\lambda = \{\varphi_h, \varphi_i\}$.

In what follows, we describe the hypertext retrieval method. First, the hypertext is reduced to a substructure consisting only of links that are sufficiently relevant to the query. In particular, we obtain the corresponding substructure by restricting ourselves to those links which satisfy the following requirements.

1. The two reduced words φ_h and φ_i constituting the link λ must occur in the query.

2. The relevance of the link $\lambda = \{\varphi_h, \varphi_i\}$ with respect to the query q must be above the threshold c,

$$RSV_{lnk}(q, \lambda) \quad > \quad c,$$

   where the relevance is estimated by

$$RSV_{lnk}(q, \lambda) \quad := \quad \frac{\sum_{i:\varphi_i \in \lambda} b_i \mathrm{nidf}(\varphi_i)}{\sqrt{\sum_{i:\varphi_i \in q} b_i^2}}.$$

Using the estimated relevance of the links, the neighborhood $D_j(q)$ of a document d_j with respect to the query q is defined by

$$(d_k, \lambda) \in D_j(q) \quad :\Longleftrightarrow \quad \begin{cases} d_k \in D1D2 \\ d_j \xrightarrow{\lambda} d_k \\ \lambda \subseteq q \\ RSV_{lnk}(q, \lambda) > c \end{cases}$$

where $d_j \xrightarrow{\lambda} d_k$ means that a link λ from d_j to d_k exists which satisfies the three conditions given above. Finally, the retrieval status values of the link method L are defined by

$$RSV_L(q, d_j) := \frac{1}{|D_j(q)|} \sum_{(d_k, \lambda) \in D_j(q)} RSV_{lnk}(q, \lambda) RSV_B(q, d_k)$$

parameter	value
ff_{min}	2
df_{min}	3
df_{max}	7000

Table 1: Parameters.

where the parameters of the link method we have chosen are shown in Table 1. Because the parameter c was adapted to the different tasks its values are shown in Section 3.

2.3 Passage Level Evidence

In this section we describe how relevant passages are extracted from documents and how the documents are scored using the scores of the passages. Hidden Markov Models (HMM) and the Viterbi-algorithm provide a natural method for retrieving passages without knowing anything about the structure of documents and without assuming anything about the format and the size of the passages. Document and passage retrieval based on Hidden Markov Models are described in (Mittendorf & Schäuble, 1994).

For our purposes we roughly assume that—with respect to a query—each document can be segmented into three passages: an irrelevant passage, followed by a relevant passage, again followed by an irrelevant passage. The HMM which models these assumptions is visualized in Figure 1. An HMM can be regarded as a set of

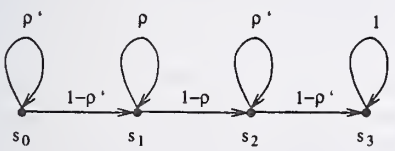


Figure 1: HMM for modeling a document

stochastic production rules for documents. These can be described by a stochastic syntax diagram. So in our case a document is produced while traversing the syntax diagram in Figure 1. Upon each arrival in a state a token is generated according to a probability distribution that belongs to this state. The next state is chosen according to the transition probabilities ρ and ρ'. In the states s_0 and s_2, tokens are generated which belong to irrelevant passages, in the state s_1, tokens are generated which belong to a relevant passage. In the state s_3 and only in this state, a token is generated that marks the end of the document.

A document is considered as a sequence of tokens. To gain query independence and a higher stability of the model parameters a description step is introduced: Each token of the sequence is mapped to a value that describes the similarity between the token and the query. If the token is an occurrence of the feature φ_i, we used

$$\mathrm{sim}(\varphi_i, q) := \mathrm{ff}(\varphi_i, q) \cdot \mathrm{nidf}^2(\varphi_i)$$

as the corresponding similarity value. Thus, the HMM can be considered as a probabilistic production rule for sequences of real values that describe the similarity of a token and a given query q.

The output probabilities are adjusted automatically by the Baum–Welch algorithm (Baum, 1972). We used the queries 1–50 and D1D2 for training. For the distribution of the similarity values in the states s_0 and s_2 we use irrelevant query-document pairs as input for the Baum–Welch algorithm, and for the state s_1 we use relevant query-document pairs. The transition probabilities ρ and ρ' are set to 0.5.

Given a document d_j, a query q, and the corresponding sequence of similarity values, the Viterbi-algorithm performs a probabilistic parse through the given HMM. It determines a path through the model which has a maximal probability of producing the given sequence of similarity values. There is a subsequence that is produced in the state s_1 while traversing the HMM along this path. The passage $p_j(q)$ which belongs to this subsequence is a subsequence of the document d_j. We will call this passage the best passage of d_j with respect to q.

The retrieval status value of a document is the similarity between the query and the best passage determined by the Viterbi-algorithm. The feature weighting is implemented as follows:

$$a'_{ij} \quad := \quad 1 + \log(\mathrm{ff}(\varphi_i, p_j(q))).$$

Since passages from relevant documents are likely to be longer than passages from irrelevant documents, it is not reasonable to normalize, as this is done for example for method B, therefore the inner vector product is chosen to be the retrieval function, i.e.

$$RSV_P(q, d_j) := \sum_i a'_{ij} b_i.$$

This passage retrieval based method is referenced to as method P.

2.4 Combination

In this section we describe how the basic method B, the link method L and the passage method P are combined. There are several possibilities for combining two sets of retrieval status values to a new set of retrieval status values. Given a method X and a method Y they can be combined by using a linear combination of the corresponding retrieval status values. The resulting combination method is denoted by XY. The basic method

B and the link method L are combined in this way:

$$RSV_{BL}(q, d_j) := \alpha RSV_B(q, d_j) + (1 - \alpha) RSV_L(q, d_j).$$

Table 2 and Table 4 show which values are chosen for parameter α.

For this kind of combination the retrieval status values of two different methods have to be on the same scale. If they are not, they could be normalized by e.g. the median or a maximal value. But sometimes one method might be rather on a logarithmic scale and the other on a quadratic scale or some scale not easy to recognize. In these cases a very sophisticated normalization is required.

Another possibility of combination is the combination of the rank numbers instead of the combination of retrieval status values (Bartell et al., 1994). In this way normalization can be avoided but some information will be lost. For a given method X and a given method Y the new method that is derived by the combination of rank numbers is denoted by $X \circ Y$. Since the retrieval status values of the method BL and of the method P are not on a comparable scale, this kind of combination was used to combine the lists of ranked documents derived from RSV_{BL} and RSV_P.

If the document d_j is ranked on position r_{BL} with respect to RSV_{BL} and if the passage p_j which belongs to d_j is ranked on position r_P with respect to RSV_P, then

$$RSV_{BL \circ P}(q, d_j) := \beta(-r_{BL}(q, d_j)) + (1 - \beta)(-r_P(q, d_j)).$$

For efficiency reasons only those documents are analyzed that are retrieved either by the method L ore by the method B, i.e.

$$RSV_P(q, d_j) := \begin{cases} \sum_i a'_{ij} b_i, & \text{if} \quad r_B(q, d_j) \leq 1000 \\ & \text{or} \quad r_L(q, d_j) \leq 1000. \\ 0 & \text{otherwise.} \end{cases}$$

Also for efficiency reasons, $r_{BL}(q, d_j)$ was assumed to be 1001 if d_j was retrieved by P and not retrieved by BL (i.e. rank greater than 1000). Similar, $r_P(q, d_j)$ was assumed to be 1001 if d_j was retrieved by BL but not by P.

If a document does not appear among the top 1000 documents on the list of ranked documents of the methods BL or P, its rank number r_{BL} respectively r_P is assumed to be 1001. The value β was chosen according to Table 2.

3 Results

We submitted the results of two ad-hoc experiments (ETH001 and ETH002) and the results of two routing experiments (ETH003 and ETH004). ETH001 and ETH003 were determined by linear combination of the retrieval status values obtained from the basic method B and from the link method L. ETH002 and ETH004 were determined by combining the ranks of the documents obtained from the combination BL and from the passage retrieval method P. An overview of all *ad-hoc* experiments is given in Table 2. Table 3 shows the retrieval effectiveness of these experiments. The numbers within brackets are comparisons to the basic method B. The following facts can be derived from Table 3:

method	description and parameter values
RSV_B	basic method nidf derived from D1D2
RSV_L	link method links derived from D1D2 $c = 0.1$
RSV_P	passage retrieval method HMMs trained using topics 1-50 and D1D2
RSV_{BL}	linear combination of RSVs from methods RSV_B and RSV_L $\alpha = 0.4$
$RSV_{B \circ L}$	linear combination of ranks from methods RSV_B and RSV_L $\beta = 0.83$
$RSV_{B \circ P}$	linear combination of ranks from methods RSV_B and RSV_P $\beta = 0.47$
$RSV_{BL \circ P}$	linear combination of ranks from methods RSV_{BL} and RSV_P $\beta = 0.8$

Table 2: Ad-hoc experiments (151-200 versus D1D2).

- Method L is much less effective than method B but 510 new relevant documents are found that have not been retrieved by the basic method B (i.e. that are not among the 1000 top ranked documents).
- Comparing the most effective BL combination ($\alpha = 0.6$) and the most effective $B \circ L$ combination ($\beta = 0.1$) shows that combining retrieval status values is cleary more effective than combining ranks. The optimal α and the optimal β have been determined using TREC-2 data.
- The combinations perform much better than the methods B, L, or P perform on their own.
- The improvements of the simple combinations BL and $B \circ P$ are complementary. The overall combination $BL \circ P$ has a better retrieval effectiveness than both simple combinations.

retrieval method	avg. precision		exact R-precision		# rel. retr. (max. 9805)		# rel. not retr. by B	submitted as
RSV_B	0.2578		0.3247		6033		-	-
RSV_L	0.1045	(-59.5%)	0.1800	(-44.6%)	3071	(-2962)	510	-
RSV_P	0.1833	(-28.9%)	0.2557	(-21.3%)	6087	(+54)	396	-
RSV_{BL}	0.2737	(+6.3%)	0.3374	(+3.9%)	6202	(+169)	364	ETH001
$RSV_{B \circ L}$	0.2616	(+1.2%)	0.3217	(-0.9%)	6134	(+101)	324	-
$RSV_{B \circ P}$	0.2853	(+10.7%)	0.3435	(+5.8%)	6156	(+123)	330	-
$RSV_{BL \circ P}$	0.2916	(+13.1%)	0.3475	(+7.0%)	6237	(+204)	381	ETH002

Table 3: Retrieval effectiveness of the ad-hoc experiments.

method	description and parameter values
RSV_B	basic method nidf determined from D1D2
RSV_L	link method links determined from D1D2 $c = 0.075$
RSV_P	passage retrieval method HMMs trained using topics 1-50 and D1D2
RSV_{BL}	linear combination of RSVs from methods RSV_B and RSV_L $\alpha = 0.3$
$RSV_{BL \circ P}$	linear combination of ranks from methods RSV_{BL} and RSV_P $\beta = 0.8$

Table 4: Routing experiments (101-150 versus D3).

retrieval method	avg. precision		exact R-precision		# rel. retr. (max. 9353)		submitted as
RSV_B	0.2993		0.3460		6024		-
RSV_{BL}	0.3092	(+3.3%)	0.3440	(-0.1%)	6242	(+218)	ETH003
$RSV_{BL \circ P}$	0.3154	(+5.4%)	0.3468	(+0.2%)	6295	(+271)	ETH004

Table 5: Retrieval effectiveness of the routing experiments.

The same methods have been applied to the *routing* experiments. We did not take advantage of the relevance assessments that were available for the routing topics. The parameter values used in the routing experiments are given in Table 4. Table 5 summarizes the results. The same statements can be derived as from the ad-hoc experiments but the improvements are somewhat smaller.

4 Conclusions

The results of our experiments seem to indicate that the link method and the passage retrieval method improve conventional vector space retrieval in a *complementary* way. The link method has the ability to retrieve relevant documents whose description vectors are not very similar to the query description vector. On the other hand, the passage retrieval method has the ability to improve an existing ordering of the documents. At this moment it is not clear whether it is possible (desirable) to measure how complementary two retrieval methods are. Finally, we lack a theoretical framework to combine several retrieval methods. The current adhoc approach to find an optimal combination is rather time consuming. There may exist more efficient ways to optimize combinations of retrieval methods.

For TREC-3, we concentrated on the ad-hoc experiments. In the routing task, we did not use the relevance information of the training collection. Thus, our methods could be improved by applying a simple relevance feedback method or by modifying the neighborhood $D_j(q)$ of the link method L.

Acknowledgements: The authors would like to thank Roger Aeppli for the implementation of the link method, for running most of the experiments and for his support in writing this paper. We would also like to thank Rene Notter for implementing the passage retrieval method.

References

Bartell, B. T., Cottrell, G. W., & Belew, R. K. (1994). Automatic Combination of Multiple Ranked Retrieval Systems. In *ACM SIGIR Conference on R&D in Information Retrieval*, pp. 173–181.

Baum, L. E. (1972). An Inequality and Associated Maximization Technique in Statistical Estimation of Probabilistic Functions of Markov Processes. *Inequalities*, 3,1–8.

Frei, H. P., & Stieger, D. (1995). The Use of Semantic Links in Hypertext Information Retrieval. *Information Processing & Management*, 31(1),1–14.

Frisse, M. E. (1988). Searching for Information in a Hypertext Medical Handbook. *Communications of the ACM*, 31(7),880–886.

Ide, E., & Salton, G. (1971). Interactive Search Strategies and Dynamic File Organization. In Salton, G., editor, *The SMART Retrieval System—Experiments in Automatic Document Processing*, chapter 18. Prentice-Hall Inc.

Knaus, D., & Schäuble, P. (1993). Effective and Efficient Retrieval from Large and Dynamic Document Collections. In *TREC-2 Proceedings*, pp. 163–170.

Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Systems Journal*, 2,159–165.

Mittendorf, E., & Schäuble, P. (1994). Document and Passage Retrieval Based on Hidden Markov Models. In *ACM SIGIR Conference on R&D in Information Retrieval*, pp. 318–327.

Porter, M. F. (1980). An Algorithm for Suffix Stripping. *Program*, 14(3),130–137.

TREC-3 Ad-Hoc, Routing Retrieval and Thresholding Experiments using PIRCS

K.L. Kwok & L. Grunfeld
Computer Science Dept., Queens College, CUNY,
Flushing, NY 11367. email: kklqc@cunyvm.cuny.edu

D. D. Lewis
AT&T Bell Labs, 600 Mountain Avenue,
Murray Hill, NJ 07974. email: lewis@research.att.com

ABSTRACT

The PIRCS retrieval system has been upgraded in TREC-3 to handle the full English collections of 2 GB in an efficient manner. For ad-hoc retrieval, we use recurrent spreading of activation in our network to implement query learning and expansion based on the best-ranked subdocuments of an initial retrieval. We also augment our standard retrieval algorithm with a soft-Boolean component. For routing, we use learning from signal-rich short documents or subdocument segments. For the optional thresholding experiment, we tried two approaches to transforming retrieval status values (RSV's) so that they could be used to partition documents into retrieved and nonretrieved sets. The first method normalizes RSV's using a query self-retrieval score. The second, which requires training data, uses logistic regression to convert RSV's into estimates of probability of relevance. Overall, our results are highly competitive with those of other participants.

1. INTRODUCTION

PIRCS is an experimental information retrieval (IR) system designed and implemented at Queens College. PIRCS allows both ad-hoc retrieval and routing of free-text documents based on natural language as well as Boolean form queries. In TREC-1, Queens College took part in Category B, with PIRCS returning highly competitive results on a 1/2 GB test collection. A major redesign of PIRCS allowed it to be applied to the full 2 GB Category A data set for TREC-2. However, only a subset of PIRCS' capabilities had been reimplemented in time, leading to PIRCS results for TREC-2 that were median and below our expectations.

For TREC-3, we employ an upgraded version of PIRCS to the full TREC-3 ad-hoc retrieval and routing of English data sets, as well as taking part in the optional experiment on thresholding for routing. For ad-hoc, we make use of several best-ranked subdocuments for query enhancement to improve retrieval results. For routing, different subsets of relevant subdocuments are used for learning. Two approaches to thresholding were tried. The first approach normalizes the raw scores using the query self-retrieval value and applies a fixed threshold. The second estimates the probability of relevance for each test document using a logistic function fitted to the training data. We show two ways to use these probability estimates for thresholding.

In Section 2, we review the salient features of PIRCS. Sections 3 and 4 discuss our ad-hoc and routing experiments respectively, while Section 5 is on thresholding. Our conclusion is in Section 6.

2. PIRCS' DESIGN

2.1 Effectiveness Aspects

PIRCS is based on the probabilistic indexing and retrieval models of [MaKu60, RoSp76] but extended with the concept of document components [Kwok90]. Both documents and queries are represented as lists of independent conceptual components that are approximated with content terms. These terms can be single words or two-word phrases and are obtained automatically from the corpus. Additionaly, queries can also have a Boolean expression structure. Term weights are based on conditional probabilities estimated from local term usage in individual items (documents or queries) and from global term usage in the whole collection. Thus, within-item term frequencies, item length, collection and inverse collection term frequencies (ICTF) and total number of tokens are effectively utilized. Moreover, PIRCS can bootstrap itself (i.e. define initial weights) based on the property that each individual item is consisted of conceptual components that are relevant to the item. The probability estimates relying

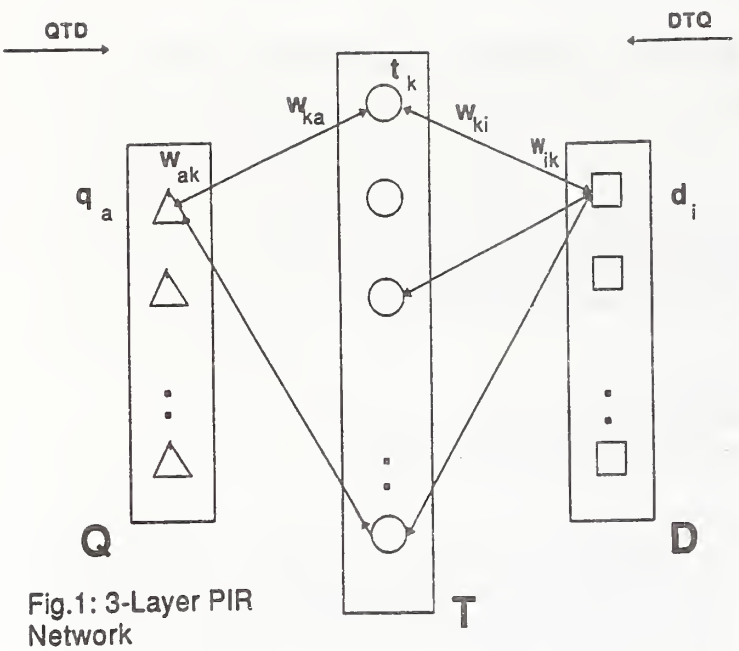


Fig.1: 3-Layer PIR Network

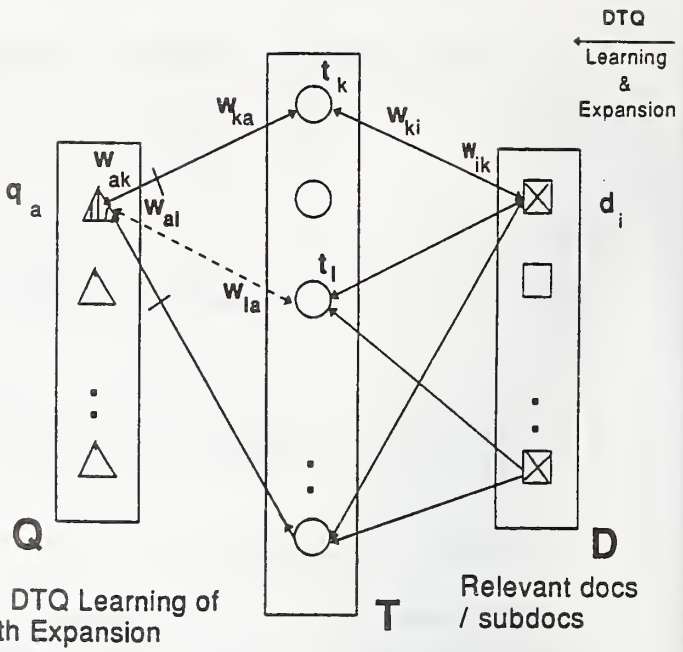


Fig.2: DTQ Learning of qa with Expansion

on item self-relevancy are quite rough because the sample of components from a single item is small at this initial stage, but may become more accurate when the system learns from larger relevant samples obtained via historical or interactive relevance judgment.

The system is implemented as a network shown in Fig.1, where the three layers of nodes (Q, T, D) denote the set of queries, terms and documents. Nodes of one layer are connected to those of an adjacent layer only, via bi-directional edges with the weights discussed earlier. In a heterogenous collection one has to deal with documents of widely different lengths, or that one document may contain several unrelated stories. We segment these documents into 'subdocuments' of more uniform sizes as before, and nodes on our network can be subdocuments or whole documents. This segmentation may lead to better retrieval, display, learning, and general efficiency in document processing as discussed in [KwGr94a,b].

During retrieval, PIRCS normally supports two modes: query-focused, when activation starts from a document and spreads towards a query under attention, or vice versa for the document-focused mode. This results in two retrieval lists for each query with different RSV's (retrieval status values) that are based on different statistics. When these RSV's are additively combined, the resultant ranking of documents almost always provides better effectiveness than either one alone. In addition, a third retrieval list can be provided if a Boolean expression tree replaces each of the set of query-term edges of the network. Thus, three types of evidence can be combined to enhance retrieval results.

An important feature of our network is that it supports training based on given relevant items. Thus, if a query has a known set of relevant documents as shown in Fig.2 (and vice versa, a document having a known set of relevant queries), the edge weights on either or both sides of the net can adapt incrementally based on term usage statistics in the relevants and attain values that would provide more effective retrieval results. Moreover, the network also supports query expansion whereby terms that are used in relevant items are selectively added with appropriate weights to the query under attention. Thus, these procedures can lead to higher recall because relevant documents having terms not contained in the original query may get retrieved, and higher precision because terms that are prevalent in relevant documents would attain larger weights and documents having them may get ranked earlier. With these learning capabilities, the network behaves like a 2-layer neural network with adaptive architecture [Kwok9x].

2.2 Efficiency Aspects

PIRCS is also unique in its processing design [KwGr94]. In our system a full inverted (posting) file is not produced as is normally done in many other systems, saving substantial disk space and reducing the 'dead time' between a collection being acquired and its availability for searching. Also, the problem of updating the full inverted file when a collection grows incrementally becomes non-existent. We create our network for learning and retrieval dynamically at query time from the direct file consisting of the list of terms within each document. The network is built based on the 'active term set' defined by the current queries and subcollection, ensuring that it is compact enough to fit into main memory. Having a network image of a textbase in RAM allows us to do learning and retrieval relatively fast.

Since TREC-2, we have reduced the time for building the network by over 80% by truncating and compressing the direct file. Depending on the query length, a single TREC topic can now be run against a 1/2 GB textbase in about 2 minutes clock time on a Sparc 10/30, producing a ranked list of 1000 subdocuments. Batching queries 10 at a time reduces the average to slightly more than one minute per query. Although this is not yet satisfactory for interactive retrieval, it is quite adequate for experimentation. We expect future upgrades will improve on this timing. For a raw text collection of x GB, we can support retrieval using only about 1.2x GB with our truncated and compressed direct file that does not include term position information. During preparation of the textbase however, more space is needed.

PIRCS also employs a master-subcollection file organization in such a way that a huge collection can be broken up into convenient subcollection sizes of say 0.5-1.0 GB each, while ensuring that documents are ranked exactly as if a single large collection and its statistics had been used. This gives us the flexibility to organize, retrieve and maintain collections of essentially unlimited sizes bounded only by available hardware. If PIRCS runs in a parallel environment with multiple processors each supporting one subcollection, then timing would also not be much more than that for a single subcollection. In addition, the previously mentioned subdocument segmentation approach to handling heterogenous documents of unconstrained lengths also contributes to the efficiency of processing.

3. AD-HOC RETRIEVAL EXPERIMENTS

3.1 Methodology

Ad-hoc experiments for TREC-3 involve creating a textbase from the revised Disks 1&2 data and retrieving with the new, unseen 4th topic set (151-200). We segment the documents into about 550-word chunks ending on a paragraph boundary. Four subcollections were produced, viz.: wa1 (274,110), fzd1 (422,660), wa2 (239,107) and fz2 (149,790) totalling 1,085,667 subdocuments. They are served by a master lexicon of 658,404 unique terms that includes 55,296 entries of our semi-automatic 2-word phrases. Our stop list remains as in TREC-2 with 630 words. The title, description and narrative sections of each topic are processed automatically like documents (but with non-content introductory phrases and 'not' sentences removed), resulting in query set Q4. A Boolean expression for each topic is also automatically generated by ORing two AND clauses that are constituted of the content terms in the title and description sections. A new feature in TREC-3 is that the concept section previously supplied with each topic is no longer available, making retrieval more difficult.

Our two submitted results are pircs1 (evaluated by NIST) and pircs2. An unsubmitted run called pircs0 consisting of our standard combination of query-focused and document-focused retrieval procedures (as discussed in Section 2) serves as a control for our internal comparison. In pircs1, we augment pircs0 by a new document-focused activation spreading which we called 'controlled recurrent processing'. In this process we let the system do some learning and query expansion from a few of the best-ranked sub-documents of each query. Ad-hoc is an initial retrieval and therefore no known relevant documents are available for training. However, previous experiments [KwPK93, KwGr94] have shown that PIRCS' ad-hoc operation can achieve average precision close to 60% in the first 5 to 10 retrieved. Hence such feedback without user judgments could be useful, and some of our limited experiments showed that it was indeed the case. It is fairly simple to implement the process in our network (Fig.1). Activation first spreads from a query node to the term layer and then to the document layer nodes. That would be our normal document-focused process. Instead, we initiate a choice operation that selects the m best-ranked document nodes with sufficient activation, and let them spread backwards to the term layer thus turning on many terms not originally connected to the query. Again, only the n highest activated terms are selected, and activation then spreads back and impacts on the document nodes providing the final RSVs for ranking. The process is very efficient; timing is practically unchanged from not using the controlled recurrent process. Moreover, it is done 'on-the-fly' so that future additions to the subcollection will influence the outcome in a dynamic fashion. We use a conservative setting of m=6 and n=30. The result is that only 40 out of 50 queries are modified, and for these an average of about 11 (443/40) new terms are added per query.

For our second ad-hoc submission pircs2, we augment our normal pircs0 with soft-Boolean retrieval. For each query a Boolean expression consisting of the OR of two AND clauses is automatically generated. Each of the AND clauses is composed of the content terms from the <topic> and <desc> sections of a topic respectively.

3.2 Results

A summary comparison of our approaches to ad-hoc retrieval averaged over the 50 queries in Q4 is tabulated in Table 1 below. Compared with our standard approach pircs0, both of our enhancements in pircs1 and pircs2 are successful. In particular, the use of a few best-ranked subdocuments for query training and enhancement (pircs1) improves over pircs0 by 8.6% in average precision, 6.6% in R-Precision and 0.6% in relevants retrieved (rel_ret). The effect on recall is negligible. Adding soft-Boolean processing is also effective in contrast to TREC-2, leading

	pircs0 (std)	pircs1 (Qexp)	pircs2 (Bool)
relev:	9805	9805	9805
rel_ret:	5981 (base)	6017 (+0.6)	6145 (+2.7)
av.precision:	.2764	.3001 (+8.6)	.2913 (+5.4)
precision at:			
5 docs	.6520	.6600	.6600
20 docs	.5090	.5670	.5240
100 docs	.3540	.3792	.3774
R-precision:	.3267	.3484 (+6.6)	.3402 (+4.1)

Table 1: Ad-Hoc Results

to increases of 5.4% in average precision, 4.1% in R-precision and 2.7% in rel-ret. We believe that the absence of the concept section in the topics (which normally has many specific keywords) has depressed the standard pircs0 result by something like 20 to 30%. This offers an opportunity for the controlled recurrent processing and the soft-Boolean operations to show some improvements.

Using pircs1 results, we observe that it recalls 61.4% of all relevants at 1000 documents retrieved. At 5 documents retrieved, one can expect more than 3 of them are relevant; at 20 documents, more than 11 are relevant; and at 30 more than half.

Comparisons with the MEDIAN values of submissions from other TREC-3 sites are summarized below in Table 2. It can be seen that both pircs1 and pircs2 results outperform the median. Since different sites make use of different techniques and methodologies, it would be interesting to see for each query which site has done best as tabulated in [Harm94] and why. We also introduce in TREC-2 MAXI-retrieval as a hypothetical system that returns the best performance for each query among all sites. This assumes that we have an intelligent agent who is able to choose the best retrieval system among this set of participants for each query, and would reflect the best we can do using our

	pircs1			pircs2		
	>	=	<	>	=	<
av. prec:	31(1)	2	17	36(0)	1	13
rel_ret @ 100:	27(1)	3	20	30(1)	6	14
rel_ret @ 1000:	30(2)	2	18	34(1)	3	13

(figure in paranthesis is number of queries equaling the best values)

Table 2: Comparison of Ad-Hoc Results with Median

collective wisdom at this time. This MAXI-system will return an average precision, precision at 100 docs and at 1000 docs of 0.4982, 0.5736 and 0.1600 respectively. Thus, Pircs1 achieves 60.2%, 66.1% and 75.2% and pircs2 achieves 58.5%, 65.8% and 76.8% respectively of these best values. Pircs1 seems to be better than pircs2 in absolute values of precision, but pircs2 outperforms pircs1 when we count the number of query results that are above median.

4. ROUTING RETRIEVAL EXPERIMENTS

4.1 Methodology

Routing involves the 3rd topic set (101-150) processing against Disk3 data. Our routing queries Q3 are defined using both the topic texts as well as their known sets of relevant subdocuments from Disks 1&2. No Disk 3 data were used for training as no relevant items from Disk 3 were available. As discussed before [KwGr94a,b], a judicious choice of training items can impact favorably on retrieval results. Short documents are the quality items and they are very effective for training. Selecting only them for training is also efficient because there is no need to do a ranking operation. However, there may not be enough of them in the known relevant set. Selecting the top-ranked subdocument of every known relevant ensures that every item is represented for training, and these subdocuments can be the most signal-rich portions of the relevants. This is an advantage of segmentation because using the top whole document may add noise in the learning process if the document happens to be long and most of it is irrelevant. However, a ranking operation is required. In [KwGr94b] we also show that the usual feedback strategy of using the x best-ranked items for training can lead to inferior retrieval results.

For TREC-3, we segment long documents and documents with multiple unrelated stories as previously, but use a larger chunk size of 550 words instead of 360 [KwPK93, KwGr94]. This saves some space by reducing the number of our network nodes. Also, if we get comparably good results as in our previous TREC-2 'further' experiments, it can be argued that segmentation size is not critical. We submitted two sets of results based on the two selection strategies discussed earlier. Pircs4 makes use only of the short relevant documents for each query, i.e. those having 160 or less unique words per document (after deleting stop words and conflating via stemming). They roughly correspond to those of TREC-2 'nonbreak' documents of about 360 raw words, and are independent of how we segment documents. There are 4,043 such short documents out of a total of 11,645 relevants. These relevants break up into 32,837 subdocuments. Thus, for pircs4 we make use of about 12% of the relevant texts only. In pircs3 which is

our official routing submission evaluated by NIST, we further merge this set of short training documents with the set of top-ranked relevant subdocuments obtained from 2,400 best-ranked subdocuments for each query. ('Top-ranked' means the set of subdocuments, only one from each document, that are ranked highest, while 'best-ranked' just means the highest ranked set of subdocuments). The 2,400 best-ranked's may not be all relevant and may have many subdocuments from the same document; so the top-ranked relevants obtained are less than 2,400 and we do not recover all 11,645. After merging with the short documents we end up with 11,559 subdocuments for training purposes. This represents about 35% of the available relevant texts. The actual process of training on our network is unchanged from TREC-2.

4.2 Results

Comparison of our two approaches for routing (see the appendix of this volume) shows that training by short relevant documents only (pircs4) is both effective and efficient, as observed previously [KwGr94a]. It gives an average precision of 0.3749, R-precision of 0.3899 and rel-ret at 1000 documents of 7318 (78.24% of all relevants). Augmenting the training document set by using a ranking operation to add top-ranked subdocuments (pircs3) involves much more work, and leads to average precision of 0.3887, R-precision of 0.3991 and rel_ret of 7640 (81.69% of all relevants). These are 3.7%, 2.4% and 4.4% better tban pircs4. If an application does not require the last few percent improvements in precision and recall, training with short documents will do. The precision at 15 documents or less retrieved is at least 60% on average. This means that one would expect 9 relevant documents in the first 15 retrieved averaged over the Q3 (101-150) query set; and in the first 5 retrieved, over 3.

Comparison with the MEDIAN values of submissions from other TREC-3 sites are summarized below in Table 3. It can be seen that both pircs3 and pircs4 results are substantially better than median. The MAXI-system returns

	pircs3			pircs4		
	>	=	<	>	=	<
av. prec:	41(4)	1	8	36(1)	0	14
rel_ret @ 100:	36(6)	8(1)	6	34(5)	5	11
rel_ret @ 1000:	38(10)	9(5)	3	33(7)	12(5)	5

(figure in paranthesis is number of queries equaling the values; some median values are also the best values)

Table 3: Comparison of Routing Results with Median

an average percision, precision at 100 and at 1000 documents of 0.4951, 0.5342 and 0.1631 respectively. Pircs3 achieves 78.5%, 86.2% and 93.7% and pircs4 achieves 75.7%, 84.4% and 89.8% respectively of these best results. This indicates that pircs performs comparatively well for exhaustive searches at the high recall end. Also pircs3 returns 7640 relevants retrieved at 1000 documents cutoff, the most of the automatic systems.

Tabulated below in Table 4 is the behavior of our routing algorithm at different levels of query expansion, and can be directly compared with our pircs3 results at expansion level 80. Effectiveness of our retrieval algorithm seems to plateau at the high end of the expansion. The behavior is quite similar to what we observed in our TREC-2 'further' experiments. We did not perform massive query expansion as done in [BuAS94].

Expansion Level:	0	40	80	120
relev:	9353	9353	9353	9353
rel_ret:	7015	7574	7640	7680
average precision:	.3256	.3782	.3887	.3960
Precision at:				
5 docs	.5360	.6480	.6400	.6560
10 docs	.5200	.6080	.6260	.6480
100 docs	.3844	.4490	.4606	.4698
R-Precision:	.3668	.3862	.3991	.4102

Table 4: Pircs3 Routing Results at Various Expansion Levels

5. THRESHOLDING EXPERIMENTS FOR ROUTING RETRIEVAL

Routing systems are presented with new documents in an ongoing fashion. Limited storage and/or limited communication bandwidth may force the system to discard all but the most relevant documents. The system must not only rank documents, but must strictly partition them into retrieved and nonretrieved sets, just as set-based retrieval (boolean query) systems and text categorization systems do.

For systems which produce RSV's, one approach to partitioning is to set a threshold and retrieve all documents with RSV's over that threshold. Thus the TREC-3 optional routing experiment requiring systems to declare which documents were retrieved was refered to as a "thresholding" experiment. It should be noted, however, that thresholding is not the only approach to partitioning [Jaco94,ToAp94].

In this section we begin by discussing some problems with the TREC-3 thresholding evaluation. We then present our approach to the evaluation, our results, and some observations.

5.1 Evaluation

A person setting up a routing system needs to know what users desire the routing system to do. This goal needs to be expressed in the form of some effectiveness measure which the system can be tuned to optimize. The goal for the TREC-3 thresholding experiment was defined by NIST as follows:

> "This threshold should be 'set' so that the system would retrieve R relevant documents, where R is the number of relevant documents that were found for this topic in TREC-2."

Retrieving a specified number of relevant documents is one of several plausible goals for a routing system. As defined above, however, the goal is unclear. Only the ideal partition of the test data (R relevant documents retrieved and, presumedly, no nonrelevant documents retrieved) is defined. It is not clear how non-ideal partitions should be evaluated. For instance, is retrieving 50% of the relevant documents and no nonrelevant ones better or worse than retrieving 90% of relevant documents plus 10% of the nonrelevant ones?

In any case, the thresholding experiment's goal was incompatible with the overall rules of the TREC-3 routing evaluation:

> "Special care should be used in handling the routing topics. In a true routing situation, a single document would be indexed and 'passed' against the routing queries. Since most of you will be indexing the test data set as a complete set, routing should be simulated by not using any test document collection information (such as IDF based on the test collection, total frequency based on the test collection, etc.) in the searching."

This meant, in particular, that routing methods were not allowed to take into account the number of documents in the test set or other global properties of the test set. But without such knowledge, it is impossible to define a rule that is expected to retrieve R relevant documents. To see this, consider a rule that retrieves R relevant documents on test set A. Now consider test set B which consists of 100 copies of the test set A. Clearly 100*R documents will be retrieved on the test set B, and under the TREC-3 rules there would be no way to distinguish the two test sets and thus, in general, no way to meet the goal of the thresholding experiment given the TREC-3 rules. (It is worth noting that there are effectiveness measures, notably decision theoretic loss measures [DuHa73], which could have been optimized under the TREC-3 restrictions.)

5.2 Approach

Despite the problems with the thresholding evaluation, we chose to test several approaches to it. A common goal of these approaches was to normalize the PIRCS RSV's so that a threshold could be determined automatically for each topic. In PIRCS, the retrieval status value that is calculated for each document is theoretically the log-odds value that the given document is relevant to the topic under attention. In practice, so many approximations are made that the raw RSV's are only useful for ranking.

5.2.1 Normalization Using Query Self-Relevance

Our first approach to normalizing RSV's relies on the property of 'self-relevance' as discussed in [Kwok85]. A query describes what a user needs. Suppose there were a 'document' identical to the query. We can presume this hypothetical document would be relevant to the query. Thus, given a query we always have one 'relevant document' as reference. We call the RSV for this 'document' the query self-retrieval RSV, or qRSV.

The qRSV for a query will normally be larger than the RSV for any true document, since true documents will only partially match the query. If we divide a document's RSV by qRSV, the result can be interpreted as a percentage of the RSV of a good 'relevant document'.

In practice, we use (constant * RSV/qRSV) as the normalized score. The hope is that this normalized score is more comparable between queries, so that to achieve a desired effect the same threshold can be used for all queries. For the TREC-3 thresholding experiment all test documents with normalized scores of 0.5 or above were considered to be retrieved. We call this strategy Self1. The threshold of 0.5 was chosen arbitrarily, and could be improved by learning from other queries, or from relevance judgments on a particular query. A related strategy, based on the score of the top ranked document rather than a self-retrieval value, has been used elsewhere [LPY94].

5.2.2 Normalization Using Logistic Regression

For our other approaches, the normalized RSV was an explicit estimate of the probability of relevance. We began by finding, for each topic, the RSV for all training documents which were judged for relevance to that topic. The resulting RSV's, along with the relevance data, were fit

to a logistic function. The result was a pair of parameters a and b for each topic, such that

$$\frac{\exp(a + b * RSV)}{1 + \exp(a + b * RSV)}$$

(where exp is the exponential function) should be a good estimate of P(T|D), the probability that document D is relevant to the topic [LeGa94]. The routing system can then generate an estimate of P(T|D) for new documents by first computing an RSV for the document and then applying the above formula with the appropriate a and b for that topic. Note that unlike Self1, this method requires relevance judgments.

Our submitted thresholding run, Prob1, was a compromise between the stated goal of the threshold evaluation and the somewhat contradictory restrictions on how routing could be done. We estimated P(T|D) for all training documents for a topic and sorted these estimates in descending order. A program then went down this sorted list, taking the cumulative sum of the P(T|D) values until the sum exceeded R (the number of documents judged relevant to the topic). The value of P(T|D) at that point was chosen as the threshold to be used on the test set. (Actually, this value was used in a normalization formula for RSV's, enabling 0.5 to be used as the threshold for all topics, but the effect was the same.) The rationale was that the summed P(T|D)'s provided a better estimate of the number of relevant above a threshold, than the number of judged relevant documents above a threshold.

As observed earlier, however, there was no reason to believe that retrieving a specified number of relevant documents was possible at all without knowledge of the size of the test set and the distribution of scores on it. We therefore tested a third method (Batch1). In this approach, the RSV's of test documents instead of training documents are converted to estimates of P(T|D), using the logistic parameters fit on the training data. (No information about test set relevance judgments is used.) Again a program sorts these estimates and sums them in descending order until a total of R is reached. Exactly the documents above that point were retrieved.

Note that Batch1 is not allowed under the TREC-3 rules, since all test documents must be scored and sorted before the decision is made to retrieve any individual document.

5.3 Results

Table 5 below shows for all three partitioning methods, the total number of documents retrieved (i.e. above threshold), estimated number of relevant retrieved (for methods that can estimate this), and the judged number of relevant retrieved, along with macroaveraged recall, precision, and R-precision as computed by the TREC-3 thresholding evaluation program.

	Total Ret	Estimated Rel Ret	Judged Rel Ret	Recall	Prec.	R-Prec.
Self1	23872	---	6708	0.7118	0.2658	0.2795
Prob1	10498	---	4562	0.5145	0.3916	0.4477
Batch1	43499	10155	7014	0.8362	0.1907	0.1765

Table 5: Thresholding Results

The desired number of relevant to retrieve (based on the training set) was 11,645. Batch1 comes closest to this target, with Self1, due to a fortuitous choice of threshold, close behind. Prob1, the submitted result, was inferior to the other two on this measure.

Batch1 is more effective than the immediate comparison of 7,014 relevant retrieved to 11,645 desired suggests. First, there were only 9,353 topic / test document pairs judged relevant at all, so this was the maximum number any system could get. Second, following the TREC-3 rules, we did not allow Batch1 to retrieve more than 1000 test documents for any topic, nor did we allow it to compensate for this limitation by attempting to retrieve more than the desired number of documents for other topics.

These limitations are reflected in the fact that the sum of the estimated probability of relevance for the 43,499 documents retrieved by Batch1 was approximately 10,155, which was lower than the target number of 11,645. Self1 was also affected by the restriction to 1000 documents retrieved since the 1000th document was still above the Self1 threshold for some topics.

Finally, not all documents which Batch1 retrieved had been judged for relevance. Of the 43,499 documents that Batch1 retrieved, only 29,193 were judged for relevance. Of those, Batch1 estimated that 8,037 were relevant, only a 14.6% overestimate compared with the true value of 7,014 relevant among these documents. Batch1 retrieved 14,306 documents which were not judged for relevance, and estimates that 2,118 of these are in fact relevant. Thus the effectiveness of Batch1, as well as of all the thresholding methods, is understated.

Batch1's 15% overestimation is similar to that observed by Cooper [CCG94] when applying a different logistic model to the TREC-2 data. A plausible explanation is that the training data, since it consists of best-ranked documents from a number of systems, contains more relevant

documents than a random sample with the same distribution of RSV's.

If we evaluate by R-precision, say, instead of by the stated thresholding goal of number of relevant documents retrieved, Batch1 is clearly inferior to both Self1 and Prob1. This reinforces our point that a meaningful evaluation of partitioning systems must define an actual effectiveness measure, not simply an ideal retrieval.

As a final obsevation, note that Batch1 estimates there are 2,118 unjudged relevant topic / document pairs among its retrieved data. Even allowing for the roughly 15% overestimation observed on the judged data, this would still suggest there are 1,840 unjudged relevant documents in just the Batch1 retrieved sets. This should be taken into account in interpreting the TREC-3 results.

6. CONCLUSION

The PIRCS retrieval system and its learning capabilities have been demonstrated to give excellent performance in a routing environment. In particular, we have shown that learning from known relevants can be beneficially affected by using only the signal-rich short documents and subdocuments for training and query expansion. For ad-hoc, the controlled recurrent activation spreading procedure, which enhances retrieval via a few initially best-ranked documents, appears useful. However, more investigation needs to be done to find the best parameters for this operation. For thresholding, the normalization of retrieval raw scores by the query self-retrieval value also seems to give reasonable results. An advantage of such an approach is that it needs not rely on past results for training (although it can make use of training data as well), and can be used in an ad-hoc environment for stopping retrieval. When training data is available, transformation to probabilities is useful for estimating and optimizing various effectiveness measures.

REFERENCES

[BuAS94] Buckley, C., Allan, J. & Salton, G. Automatic Routing and Ad Hoc Retrieval using SMART: TREC2. In: The Second Text REtrieval Conference (TREC-2). Harman, D.K. (Ed.). NIST Special Publication 500-215, 1994, pp. 45-55.

[CCG94] Cooper, W.S., Chen, A. & Gey, F.C. Full text retrieval based on probabilistic equations with coefficients fitted by logistic regression. In: The Second Text Retrieval Conference (TREC-2). Harman, D.K. (Ed.). NIST Special Publication 500-215, 1994, pp. 57-65.

[DuHa73] Duda, R.O. & Hart, P.E. (1973). Pattern Classification and Scene Analysis. New York, Wiley-Interscience.

[Harm94] Harman, D.K. Overview of the Second Text REtrieval Conference (TREC-2). In: The Second Text REtrieval Conference (TREC-2). Harman, D.K. (Ed.). NIST Special Publication 500-215, 1994, pp. 1-20.

[Jaco94] Jacobs, P.S. GE in TREC-2: Results of a boolean approximation method for routing and retrieval. In: The Second Text Retrieval Conference (TREC-2). Harman, D.K. (Ed.). NIST Special Publication 500-215, 1994, pp. 191-199.

[KwGr94a] Kwok, K.L. & Grunfeld, L. Learning from relevant documents in large scale routing retrieval. In: Proc. Human Language Technology Workshop, ARPA, Mar 8-11, 1994 Plainsboro, NJ. Morgan Kaufmann, San Francisco, 1994, pp. 358-363.

[KwGr94b] Kwok, K.L. & Grunfeld, L. TREC-2 Document retrieval experiments using PIRCS. In: The Second Text REtrieval Conference (TREC-2). Harman, D.K. (Ed.). NIST Special Publication 500-215, 1994, pp. 233-242.

[Kwok90] Kwok, K.L (1990). Experiments with a component theory of probabilistic information retrieval based on single terms as document components. ACM Transactions on Office Information Systems, 8:363-386.

[Kwok85] Kwok,K.L (1985). A probabilistic theory of indexing and similarity measure based on cited and citing documents. J. of American Society for Information Science, 36(5):342-351.

[Kwok9x] Kwok, K.L (199x). A network approach to probabilistic information retrieval. ACM Transactions on Office Information Systems, to appear.

[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, 1993, pp. 153-172.

[LeGa94] Lewis, D.D. & Gale, W.A. A sequential algorithm for training text classifiers. In: SIGIR 94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Croft, W.B. and van Rijsbergen, C.J. (Eds.). Springer-Verlag, London, 1994, pp. 3-12.

[LPY94] Liddy, E.D., Park, W. and Yu, E.S. (1994). Text categorization for multiple users based on semantic features

from a machine-readable dictionary. ACM Transactions on Office Information Systems, 12(3):278-295.

[MaKu60] Maron M.E & Kuhns, L.J (1960). On relevance, probabilistic indexing and information retrieval. J. ACM 7:216-244.

[RoSp76] Robertson, S.E. & Sparck Jones, K. (1976). Relevance weighting of search terms. J. of American Society for Information Science, 27:129-146.

[ToAp94] Tong, R.M. & Appelbaum, L.A. Machine learning for knowledge-based document routing (a report on the TREC-2 experiment). In: The Second Text Retrieval Conference (TREC-2). Harman, D.K. (Ed.). NIST Special Publication 500-215, 1994, pp. 253-264.

Searching For Meaning With The Help Of A PADRE

David Hawking and Paul Thistlewaite
Co-operative Research Centre For Advanced Computational Systems
Department Of Computer Science
Australian National University
{dave,pbt}@cs.anu.edu.au

December 1, 1994

Abstract

Full-text scanning offers significant advantages over other methods of document retrieval but is normally too slow for use on large collections. The Fujitsu AP1000 parallel distributed-memory machine has been used to reduce the time penalty for full-text scanning to acceptable interactive levels. The query language for the retrieval software (called PADRE) is described herein and differences between PADRE and traditional systems are highlighted. The advantages of the full-text scanning in broader retrieval contexts are outlined. TREC precision-recall results are discussed and timings are reported.

1 Introduction

The computational cost of full-text scanning as a method of document retrieval is usually regarded as prohibitive for large document collections. The availability of a large-memory parallel machine challenges this wisdom by permitting a full scan of a TIPSTER-sized collection to be carried out in of the order of one second.

Full-text scanning offers the following advantages over other methods:

- It preserves all the information in the document collection.
- It allows queries to be constructed from completely general operations over the text.
- It permits rapid response to changes in the document collection, such as the addition or deletion of documents.

PADRE is a document retrieval system designed and implemented by the author for the Fujitsu AP1000 parallel distributed memory supercomputer. The architecture of this machine is described in [1, 2, 3]. Fuller details of the design and implementation of PADRE are to be found in [9]. PADRE's capabilities are constructed on top of a small set of basic pattern matching operations.

2 Capabilities Of The PADRE Query Language

The PADRE query language is more fully described in the User Manual [7]. Below is a brief summary of its capabilities. Note that the query language is not very user friendly. It is intended that PADRE queries will be generated by a more friendly, perhaps graphical interface. Indeed a graphical front-end for PADRE does exist (see Hawking and Bailey [6]) but it is only capable of accessing a subset of PADRE functionality.

2.1 Primary Terms

The basic pattern matching operations implemented by PADRE are as follows:

- Literal strings (using Boyer-Moore-Gosper (BMG) string matching);
- Numeric ranges (matches strings of digits whose numeric value lies in a nominated range, eg. 1776..1859); and
- Regular expressions (using GNU regular expression code)

2.1.1 Word-Start Mode

By default, matches for literal strings are constrained to start at the beginning of a word. For example, searching for `"ice"` will find only words beginning with that string. This behaviour can be changed by changing the value of the `wsmode` variable to any in which case matching is independent of word starts. In this case, `ice` would match `nice` and `spices` as well as `iced`.

2.1.2 Case Sensitivity

The `casesensitive` variable allows the searcher to specify whether searches for literal strings are to be considered case sensitive or otherwise.

2.1.3 Spaces In Patterns

A space in a PADRE literal term matches any single non-alphameric character.

This allows the searcher to locate phrases, such as `"wool exports"`, and word suffixes, such as `"ize "` which with `wsmode=any` will match any word ending in `"ize"` even if the character following is a punctuation mark.

Introducing a meta-character to match an arbitrary sequence of non-alphamerics would be a significant improvement but this has not yet been implemented in the BMG context. A regular expression can achieve this end but the computational cost is much greater. Alternatively, the searcher can make multiple BMG searches for the pattern with one space, two spaces, ..., etc. and then combine the results.

2.2 Regular Expressions

PADRE's `regexp` command allows the searcher to locate all occurrences of strings matching a regular expression using the Unix *egrep* syntax. PADRE incorporates the Free Software Foundation GNU regular expression code almost unmodified. The following example matches the most common pattern of vehicle registration (license) plate from the Australian Capital Territory:

```
R1: regexp "\<Y[A-Z][A-Z][0-9][0-9][0-9]\>"
```

2.3 How PADRE Represents The Results Of A Search

The result of a PADRE search takes the form of an ordered array of pointers to the first character of each match in the text collection. Such an array is called a *match set.* Match sets are used directly by the commands which display lexicographic context and are also used to calculate a component of the cumulative relevance measure for a document. Note that the match set is distributed across all processors in the parallel machine.

2.4 Compound Terms

Compound terms modify the behaviour of primary terms or combine the match sets resulting from searching for other terms, either primary or compound. There are three types of compound term:

- Component search (eg. *pattern* **within component** *component name*);
- Proximity search (eg. *pattern1* **near** *pattern2*); and
- Set operations, which allow pairs of match sets to be combined using difference, union and intersection operations. Set operators allow named as well as immediate operands.

2.4.1 Component Searches

PADRE permits the searcher to define components such as title, abstract, definition, headword etc., by specifying start and end markers for each component. For example the title of a document may be enclosed by `<title>` and `</title>` markers.

Subsequent searches for primary terms may be constrained to lie within particular components. Currently, only literals may be used in component searches but there will be no particular difficulty in generalising this.

2.4.2 Proximity Searches

Proximity is currently expressed as a fixed number of characters; it would be useful, and may also be feasible, to express it in terms of paragraphs, sentences or words.

The proximity operators `near` n and `fby` n operate on the n most recently computed match sets and produce a result set consisting of all the members of the first match set which satisfy the appropriate *near* (or *followed by*) relationship with members of the other sets. The following group of commands creates a result match set indicating all the occurrences of "Clinton" which are followed within 200 characters by one or more occurrences of "Yeltsin".

```
>> {proximity 200}
>> "Clinton "
>> "Yeltsin "
>> near 2
```

Note that proximity operators can be applied regardless of how the match set operands were computed.

2.4.3 Set Operations

Union, intersection and difference operators may be applied to named or immediate operands. They produce a result match set by applying the appropriate operator to two match sets. The union operator is useful for matching a series of alternatives. The following command makes a set of pointers to all occurences of each of the strings.

```
>> "car " + "vehicle " + "sedan "
```

Difference can be used to exclude specific instances of a general pattern. For example, the following command finds all words starting with `comput` except those starting with `computer`.

```
>> "comput" - "computer"
```

2.5 Measuring Relevance

A document's estimated relevance is computed according to the following formula:

$$R_d = 10000 \times \sum_{t=1}^{k} (I_t \times r_{(t,d)}) \tag{1}$$

where:

R_d is the estimated relevance of document d,

$r_{(t,d)}$ is the relevance of a document d due to term t,

I_t is the manually assigned importance of term t, and

k is the number of terms.

$$r_{(t,d)} = \frac{f_{(t,d)}}{\sqrt{F_t \times l_d}} \tag{2}$$

where:

$f_{(t,d)}$ is the frequency of term t in document d,

F_t is the frequency of term t in the entire collection, and

l_d is the length of document d

The factor of 10,000 is introduced merely to scale the relevance measures into a more convenient range. It may be that a logarithmic function rather than a square root would be even more effective at preventing the bias against long documents and toward infrequent terms from being applied too severely.

2.6 Identifying and Displaying Relevant Documents

The commands `top` n and `retrieve` n respectively identify and retrieve the n documents with the highest accumulated relevance scores.

2.7 Displaying Lexicographic Context

The searcher may request the display of each match in the current match set with specified number of characters of pre and post context. The `pr` command displays all matches; the `sample` n command displays at most n of them.

2.8 Key Differences Between PADRE And Conventional Systems

1. Use of pattern matching as the basic operation.
2. Availability of manually assigned term importance weighting.
3. Ability to use negative or zero importance weights.
4. Different meaning of set operations.
5. Support of regular expressions.

3 TREC-3 Participation

PADRE was entered only in the Ad Hoc section of the TREC competition. Unfortunately, the generation of both Manual and Automatic queries was subject to significant constraints on human resources and time. A TREC-ready form of the PADRE query language was first implemented only a few weeks before the deadline for entries and insufficient time was available to benefit from the training topics.

3.1 Generation Of Manual Queries

Manual queries were generated by an unskilled researcher (the principal author). Proximity operators and manually assigned term importance weightings were used extensively to try to improve precision and word prefixes were used to match multiple variants of the same word. For example:

```
{weight 0}
"smoking" + "smokers" + "tobacco "
"ban " + "prohibit" + "disallow" + "forbid" + "outlaw" + "bann"
{weight 1000}
near 2
```

Negative term importance weightings were used to implement exclusions specified in the topics.

```
{weight 1}
"murder" + "homicide" + "kill" + "assassinate" + "strangle" + "manslaughter"
"motive" + "reason " + "because "
{weight 1000}
near 2
{weight 0}
"popular" + "author " + "fiction" + "reviewer" + "book " + "whodunnit"
{weight -900}
near 2
```

Manual thesaurus expansion was used to improve recall. Synonyms and specific instances of general terms were generated using general knowledge, dictionaries, thesauri, subject-specific material and a friend who knew the names of American restaurant chains. Terms were combined using the PADRE union (+) operator.

3.2 Generation Of Automatic Queries

The core of the automatic query generation software is an algorithm for generating an ordered list of key words and phrases from a document. The topic specification provided by TREC is processed by this algorithm, and the output is formatted to comply with the PADRE query language.

The algorithm is based on familiar statistical techniques using word and phrase frequency information, word stemming, and phrase subsumption. Currently, no thesaurus or semantic-net information is used, and the algorithm could clearly be strengthened by its incorporation.

The relative weights for the PADRE query are calculated by normalising the rank values generated by the keywording software. Equivalent words and phrases (e.g., "Australian live

sheep shipments", and "shipping live sheep from Australia") are detected by the algorithm, and the PADRE "+" operator is used to represent their synonymy. Phrases are not represented as strictly ordered sequences of words; rather the PADRE proximity and near operators are used to permit greater flexibility in word separation and order.

4 PADRE Performance

This year's results are relatively poor, though having achieved best results on three of the topic-specific measures is quite encouraging. We believe that the major area of potential improvement for next time lies in the process of query generation rather than in the basic design of PADRE.

The lack of a user-friendly query interface, coupled with lack of sleep on the part of the person generating the manual queries, caused a significant reduction in performance in the Manual category. No fewer than fourteen of the 50 Manual queries were later discovered to include obvious errors. Errors included neglecting to reset weight to zero before terms of no significance by themselves (12 times), leaving out a proximity operator (three times), and mistyping "santion". Correcting these errors (before looking at the documents retrieved) caused dramatic performance improvements on three topics, changing performance levels from worst or near worst to above median.

More generally, a lack of experience with the document collection and with generation of retrieval queries, shows up as rather poorly formulated queries.

4.1 Case Study - Topic 155, Manual

Right Wing Christian Fundamentalism - 42 relevant documents.

This was originally submitted with a missing "weight 0" meaning that any document mentioning any of "constitution", "Constitution", "civil liberties", "Civil Liberties", "US", "U.S.", "United States" or "USA" was regarded as highly relevant. Consequently, 1000 documents were retrieved of which none were relevant.[1] The topic as originally submitted appears below.

```
topic 155
{weight 0}
{proximity 100}
"right wing " + "fundamentalis"
"christian"
"grass roots " + "politic" + "religious agenda " + "elector"
{weight 500}
near 3
{casesensitive 1}
"constitution" + "Constitution" + "civil liberties" + "Civil Liberties"
+ "US" + "U.S." + "United States" +"USA"
{casesensitive 0}
{weight 1000}
near 2
top 1000
```

[1] Sincere apologies to the person who read the 200 documents erroneously retrieved due to this error!

After correction of the error, 27 documents were retrieved of which still only 5 were relevant. To investigate why, all documents officially judged relevant and the documents retrieved by the modified PADRE query were extracted and examined. It soon became clear that the query was defective in several ways:

- A number of words and phrases connoting the religious dimension were missing from the query partly due to the query author's lack of subject knowledge. Examples include "Moral Majority", "evangeli", "christian right", "prayer", "preacher", "ministry", "Pat Robertson", "Jerry Falwell".
- A number of words and phrases connoting the US political dimension were missing from the query partly due to the query author's lack of subject knowledge. Examples include "GOP", "Republican", "Democrat", "primaries", "Super Tuesday", "Campaign", "presidential", "nomination", "party".
- During initial query design, the author could not see how to enforce the requirement that the documents must relate to the U.S. It was hoped that by assigning greater weight to documents whose references to religion and politics occurred also near references to the United States, the constitution or the second amendment, this would cause actually relevant documents to achieve the highest rank, while allowing documents rendered irrelevant because of nationality to cause false hits lower down the list. This strategy did not work, largely because documents written in the United States about the U.S. political process do not generally name the U.S. A better strategy would attempt to include documents which mentioned names of U.S. states, U.S. cities, U.S. polititicians, U.S. religious figures and U.S. political parties and to exclude documents which frequently referred to foreign places and personalities. In fact, most of the erroneously retrieved documents could be excluded or down-weighted by assigning negative weights to places and people associated with Lebanon.
- The proximity of 100 was too tight when looking for associations between terms connoting the religious dimension and terms indicating the U.S. political dimension.

We are optimistic that a refined query addressing these defects would achieve dramatically better performance.

4.2 Speed

PADRE runs for the TREC competition were run on a 512-processor configuration of the Fujitsu AP1000, a machine with 8 gigabytes of RAM. Competition runs used full-text scanning exclusively. Table 1 shows the average times taken to process TREC-3 topics in this way.

Though it was not used in the TREC competition, PADRE does have the ability to build a distributed inverted file index. Table 2 shows the relative speeds of the three alternative PADRE methods for locating literal strings. Table 3 shows the time taken to build an inverted file index over the TREC data.

A time of 40 seconds to process a complex TREC query does not seem excessive. However, considerable improvement is possible. Dramatic improvements could be made by building an inverted file and using it to search for literal terms. The applicability of the inverted file could readily be extended.

Even without using an inverted file, improvements are likely to be possible in the following areas.

Query Type	No. terms per topic	No. set/proximity operations per topic	Mean time (sec.) per topic
Manual	11–110 (mean=27.4)	9–105 (mean=24.7)	38.1
Automatic	5–57(mean=18.7)	1–29 (mean=9.4)	42.3

Table 1: Elapsed time required to process TREC-3 topics on a 512-processor Fujitsu AP1000 using full-text scanning. Times are averaged over all 50 topics. They do not include time taken to load the data into memory across the network.

Search method	Eapsed time (sec.)
Boyer-Moore-Gosper	1.12
GNU Regular Expression	6.60
Indexed	0.02

Table 2: Elapsed time required to find all occurrences of a literal term using three different methods. Each time is the average of the times taken to locate the six strings: "ape", "fish", "cannibal", "Australia", "reject", and "unconditional". The data set comprised all three TREC CD-ROMs after purging manual indexing terms and the machine was a 512-processor Fujitsu AP1000.

Data set	Elapsed time (sec.)
CD1 and CD2 combined	92.18
CD1, CD2 and CD3 combined	128.41

Table 3: Elapsed time taken to build an inverted file index over the TREC-3 data. using a 512-processor Fujitsu AP1000. Note that the inverted file actually consists of 512 separate partial indexes which do not need to be merged.

- Replacing the multiple literal scans and multiple union operations used to find alternates with a single-pass finite state automaton method.
- Replacing the very inefficient algorithm for returning documents in order of decreasing rank.
- Introducing a more effective automatic method of balancing load across cells.

PADRE's ability to handle dynamic document collections has been reported elsewhere [9]. Retrieval downtimes caused by additions to or removals from the document collection did not exceed 20 seconds for changes of 10 megabytes or more.

5 Desirable PADRE Extensions For Future TRECs

1. Tuning up the software which generates automatic queries and adding thesaurus and semantic net capabilities are likely to produce the greatest benefit.
2. For manual query generation, a well-designed user interface is sorely needed.
3. It would be very useful if PADRE incorporated pre-defined *class terms* which could be efficiently located in the text. For example, the class term `US-states` might match any of Alaska, Arizona, etc.

6 Beyond TREC-3

The architecture of PADRE is capable of supporting retrieval based on any function computable over text. In its present form PADRE includes capabilities which were not exercised in the TREC-3 competition.

Numeric ranges are useful in conjunction with proximity operators. They would also be useful in component searches. It is planned to remove the current implementation restriction which prevents the latter.

```
>> "Nixon "
>> 1968..1972
>> near 2
..
>> 1968..1972 within component publication_date
```

Thought will also be given to extending PADRE to handle more general date comparison functions. For example:

```
>> published before June 1985
```

Below are some illustrations of PADRE's regular expression capabilities. R2 matches numbers from 1900 to 1999. R3 could be used to find documents containing Gaithersburg area telephone numbers. It matches several different forms, while R4 matches NIST email addresses.

R5 and R6 illustrate the use of regular expressions to match alternative spellings or variations on the same word. R6–R8 could be used to find documents of particular types, based on content. R6 matches a common form of LaTeX command, R7 matches the familiar SGML pattern `<blah> .. </blah>` and R8 matches C language comments.

```
R2: regexp "\<19[0-9][0-9]\>"
R3: regexp "\<\+?1?\-?\(?301\)?[- ][0-9][0-9][0-9][- ]?[0-9][0-9][0-9][0-9]\>"
R4: regexp "\<[a-zA-Z_\-]+@(ncsl.nist.gov)|(NCSL.NIST.GOV)\>
R5: regexp "\<judge?ment\>"
R6: regexp "\<ha(th)|d|s|(ve)|ving\>"
R6: regexp "\\begin{.*}"
R7: regexp "<([a-zA-Z]+)>.*</\1>"
R8: regexp "/\*.*\*/"
```

PADRE regular expressions can also be used to find pallindromic words of up to 19 characters in length, however this requirement arises rather infrequently in practical document retrieval applications.

The PADRE architecture could feasibly be extended to support retrieval based on stylistic properties. For example, retrieving documents in order of decreasing stylistic similarity to the Gettysburg address. A future PADRE might also be able to be used to retrieve documents which quote or paraphrase texts of interest.

Acknowledgements

Fujitsu Laboratories provided access to AP1000 machines in the Fujitsu Parallel Computing Research Facility and assisted in various ways. Robin Stanton gave support and advice. Peter Bailey and Michael Hiron worked on elements of PADRE code. GNU regular expression code from the Free Software Foundation is incorporated in PADRE. I extend my sincere thanks to all these people and organisations.

Recent work on PADRE has been supported by the Co-operative Research Centre for Advanced Computational Systems (ACSys).

References

[1] Horie, T., Ishihata, H., Shimizu, T. and Ikesaka, M. 'AP1000 Architecture And Performance Of LU Decomposition,' in *Proc. 1991 Int'l Conf. On Parallel Processing*, pp. 634-635, August 1991.

[2] Ishihata, H., Horie, T., Inano, S., Shimizu, T. and Kato, S. 'CAP-II Architecture,' in *Proceedings of the First Fujitsu-ANU CAP Workshop*, paper 1, Kawasaki, Japan, (Nov 1990).

[3] Horie, T. Ikesaka, M. and Ishihata, H. 'Interconnection Network For Multiprocessors', in *Proceedings of the First Fujitsu-ANU CAP Workshop*, paper 2, Kawasaki, Japan, (Nov 1990).

[4] Hawking, D.A. "High Speed Search of Large Text Bases On the Fujitsu Cellular Array Processor," *Proceedings of the Fourth Australian Supercomputing Conference* pp. 83-90. Gold Coast, Australia, (Dec 1991).

[5] Hawking, D.A. "PADDY's Progress (Further Experiments in Free-Text Retrieval on the AP1000)," in *Proceedings of the First Annual Users' Meeting of Fujitsu Parallel Computing Research Facilities* paper ANU-8, Kawasaki, Japan, (Nov 1992).

[6] Hawking, D.A. and Bailey, P. "Towards a Practical Information Retrieval System For The Fujitsu AP1000)," in *Proceedings of the Second Fujitsu Parallel Computing Workshop* paper P1-S, Kawasaki, Japan, (Nov 1993).

[7] Hawking, D.A. *PADRE User Manual* Department of Computer Science, Australian National University, Canberra, Australia, Oct 1994.

[8] Hawking, D.A. "PADRE — A Parallel Document Retrieval Engine," in *Proceedings of the Third Fujitsu Parallel Computing Workshop* paper P2-C, Kawasaki, Japan, (Nov 1994).

[9] Hawking, D.A. "The Design And Implementation Of A Parallel Document Retrieval Engine," *In Preparation*

Using An N-Gram-Based Document Representation With A Vector Processing Retrieval Model

William B. Cavnar
379 Brookside
Ann Arbor MI 48105
wcavnar@mail.msen.com

1.0 Abstract

N-gram-based representations for documents have several distinct advantages for various document processing tasks. First, they provide a more robust representation in the face of grammatical and typographical errors in the documents. Secondly, N-gram representations require no linguistic preparations such as word-stemming or stopword removal. Thus they are ideal in situations requiring multi-language operations.

Vector processing retrieval models also have some unique advantages for information retrieval tasks. In particular, they provide a simple, uniform representation for documents and queries, and an intuitively appealing document similarity measure. Also, modern vector space models have good retrieval performance characteristics.

In this work, we combine these two ideas by using a vector processing model for documents and queries, but using N-gram frequencies as the basis for the vector element values instead of more traditional term frequencies. The resulting system provides good retrieval performance on the TREC-1 and TREC-2 tests without the need for any kind of word-stemming or stopword removal. We also have begun testing the system on Spanish language documents.

2.0 Background

2.1 N-Gram Representations

An N-gram is an N-character slice of a longer string. For example, we could represent the word TEXT with the following N-grams:

- bi-grams: _T, TE, EX, XT, T_
- tri-grams: _TE, TEX, EXT, XT_
- quad-grams: _TEX, TEXT, EXT_

where the underscore character represents a leading or trailing space. N-grams provide a distributed representation, in which each word is represented by a set of N-grams. Words that are similar, say by virtue of having a different suffix or a spelling variation, will nonetheless share a large number of N-grams. For example, consider the words RETRIEVE, RETRIEVAL, and RETRIEVING, which share the bi-grams _R, RE, ET, TR, RI, IE, and EV. Likewise, RETRIEVE and its frequent misspelling RETREIVE share most of their N-grams.

We built several systems using this notion of N-gram-based representations to perform various retrieval tasks. The first was a simple text retrieval tool called *zview* which served as an informal corporate database at ERIM. *zview* provided very easy-to-use access to a set of ASCII documents since it tolerated spelling and word order variations in queries, and required no Boolean search expressions.

Another example of N-gram-based representations was in some of our work for the U.S. Postal Service [Cavnar93a]. In conjunction with USPS-sponsored research on address interpretation, we built an N-gram-based system for matching city, state and ZIP information off of an address in order to determine a set of relevant ZIP codes. This was quite important, given the realities of postal patrons' addressing errors, peculiar local addressing conventions, errors in the Postal databases, and errors in OCR systems attempting to read low-quality address images. Thus, one could not count on any one piece of information in an address as being correct. It was only the coordinated redundancy inherent in an N-gram-based representation that allowed this system to maintain a high retrieval performance in the face of multiple errors in the city, state and ZIP information.

2.2 TREC

The TREC competitions [Harman94] are a series of conferences sponsored by ARPA and NIST specifically to stimulate the advancement of the art of information retrieval by

- Providing large standardized datasets, query sets, and relevance assessments for measuring retrieval performance, and

- Providing a conference where both commercial vendors and academic researchers may meaningfully exchange approaches and results. Furthermore, these conferences are high in content and reasonably free of advertising hype.

For TREC-2, we fielded a retrieval system that used an N-gram-based representation, but with an adhoc retrieval model [Cavnar93b]. This system had several notable deficiencies:

- It was designed as a filter rather than as a true retrieval system. Thus, every run of the system required a complete pass of all of the data. This process took hours for a single query.

- The system built queries from the topic statements in a very simplistic way, using only the <concept> section of the topics. This was unfortunate, since many very relevant terms appeared only in the narrative portion of the topics. Furthermore, we knew that for TREC-3, at least some of the topic sets would not have <concept> sections at all.

- The system used a completely adhoc method of term weighting based on the order of the terms in the <concept> section of each topic and somewhat on the frequency of occurrence of the terms in each document. Although we tried several different variations, none proved at all adequate.

On the other hand, we did demonstrate that the system could tolerate some level of typographical errors in the queries. This level of errors would have rendered most, if not all, of the other TREC-2 systems completely helpless.

During the same time, we also did some research on using N-gram occurrence frequencies to categorize documents by language and subject [Cavnar94]. Basically, the idea was to compute an N-gram frequency profile for both a set of category samples and for a document. Using a simple distance measure, we then computed distances between the document and each of the category samples. For language categorization, this approach was remarkably successful. Using only the top 300 N-grams by frequency (where N = 1 to 5) we could correctly identify the language of a Usenet posting in the soc.culture hierarchy 99.8% of the time. The system was somewhat less successful in the subject categorization task we tested, but the system performed well enough on certain parts of the task to suggest that the approach merited further inquiry.

2.3 Vector Processing Retrieval Models

The most common type of text retrieval system uses complicated Boolean expressions to specify relevant documents by selecting on the presence or absence of terms and combinations of terms. Although researchers and vendors alike have been working on these kinds of retrieval systems for decades, there continue to be difficulties with using them. First, it is difficult for novice users to get satisfactory results because of the complexity, and in some cases, the opacity of Boolean term expressions. Moreover, even for experts, some kinds of information needs are difficult or tedious to specify in Boolean expressions.

Among the alternatives to Boolean retrieval systems is the vector processing retrieval model introduced by Salton over 30 years ago, and continuously researched by him and his group since then [Salton94]. The key concept in vector processing models is representing both documents and queries as vectors in very high dimensional spaces. Typically, each dimension in a vector represents the frequency of occurrence of a term in the document or query. Given all of the possible terms in a document collection, a vector may well have many thousands of dimensions, most of whose specific values are zero for a given document. Viewing documents and queries as vectors suggests one very natural measure of similarity: the angle between two vectors. That is, the smaller the angle between two vectors, the more similar we would consider their respective documents or queries to be. In practice, most vector processing systems actually use the cosine of the angle between the vectors as the actual measure since that is easy to calculate and yields a number between 1.0 and -1.0.

Viewing documents and queries as vectors has many attractions. For example, one could use a document itself as a query, which would enable the system to find documents similar to a given document. One could likewise form a vector to represent a whole set of documents by simply averaging the values of each of their dimensions or performing some other similarly plausible geometric operation on their vectors.

3.0 System Description

Given the success of using N-gram frequency information for text categorization that we mentioned earlier, we felt inspired to find a different approach to the TREC task that made better use of this information. The similarity in spirit between our N-gram frequency profiles and the vectors used in vector processing retrieval models suggested to us a new possibility. We decided to try using a conventional vector processing model, but with N-gram frequencies instead of term frequencies for the dimensions. We also decided to limit the system to just quad-grams for this version.

Although shorter N-grams work very well for name matching, our experience with the text categorization task showed that the longer N-grams are the ones that seem to carry more content or meaning. We used quad-grams composed of digits and letters, and also the leading and trailing space for each word.

This idea is very similar to one independently developed by Thomas and described in [Thomas]. Although that system also used N-grams and a vector processing model there were several notable differences:

- Thomas's system used only tri-grams of letters.
- It ignored all word boundaries due to punctuation, white space, or digits, and treated each line as a very long single word.
- It did not make use of the notion of inverse document frequency.

Also Thomas did not attempt the TREC task, but successfully performed a smaller relevance measurement task retrieving medical records. Our motivation for developing an N-gram-based vector processing system is very similar to that which inspired Thomas, and produces largely the same benefits.

Figure 1 gives a dataflow diagram for the system we designed. We will describe each component of these in some detail.

3.1 *ngram_stats* And The Quad-Gram Dictionary

The first bubble, labeled *ngram_stats,* represents a program which reads the documents and builds a dictionary of quad-grams and their associated information. Among the attributes that the dictionary associates with each quad-gram is its inverse document frequency (IDF). The IDF measures how focused a word is in the collection: a word with a high IDF would occur in few documents, whereas one with a low IDF would occur in most documents. See section 4.0 for a detailed description of the IDF calculation.

The quad-gram dictionary provides the mechanism which allows the system to keep track of each quad-gram in order to apply the appropriate term weighting. After building the dictionary, *ngram_stats* removes all quad-grams that occur only once. Data disks 1 and 2 together had 171514 entries left in the dictionary, while disk 3 had only 136414.

3.2 *ngram_vecs* And Document Vectors

The process represented by the bubble titled *ngram_vecs* reads the document files and produces for each document a vector giving the appropriate weights for each quad-gram with respect to that document. This program dealt with parsing the SGML structure of the document files, splitting the body of the text into quad-grams, counting the quad-grams, and building the actual document vectors according to the resulting quad-gram frequencies.

Even though most of the quad-grams had zero weights for any particular document, prompting us to use a sparse vector representation, we still had a problem with the space required to store the non-zero weights for a document. In general we found that the more quad-grams with non-zero weights that we kept in the vector, the better the retrieval performance. Unfortunately, that also meant that the vector file required more disk space. Happily, the IDF corresponds very strongly with the importance of the quad-gram, and thus makes it easy to threshold on the IDF as a way of limiting quad-grams in the vectors to the most important. We compromised on a minimum IDF threshold of 2.75, which maximized retrieval performance on the several test sets we had available while staying within disk availability. At this threshold, the vector for documents in disks 1 and 2 had an average length of 360 quad-grams out of an average of 1460 possible quad-grams. We did a few informal experiments on disk 3 reducing the minimum IDF threshold all the way down to 2.0. The best performance in this short set of experiments was at an IDF of 2.25. However, even at this level, the improvement was a few percent in precision, and that was at the expense of increasing the storage by over 40%.

The overall effect of this IDF thresholding was very similar to performing word-stemming and stopword removal. Quad-grams with low IDFs correspond precisely to the lexical components that word-stemming and stopword removal would eliminate. The great advantage of the N-gram-based approach is that this elimination is dictated entirely by the statistics of the language. There is no need for a step requiring detailed linguistic knowledge, and thus the system is immediately usable without change on any language that can be represented in ASCII. To illustrate this, we have provided some frequency statistics tables in Appendix I. Table 1 in the Appendix lists the top 20 quad-grams by frequency from disks 1 and 2. Notice that nearly all of these quad-grams are components of words or suffixes that would have to have been eliminated by the more traditional linguistic preprocessing. The one difference is the presence of very frequently occurring prefixes, such as 'PRO', 'COM', and 'CON'. Given their extremely high frequency, these prefixes actually contribute very little semantic content. Likewise, Table 2 shows similar statistics for the Spanish data, and again, the very frequent quad-grams represent those

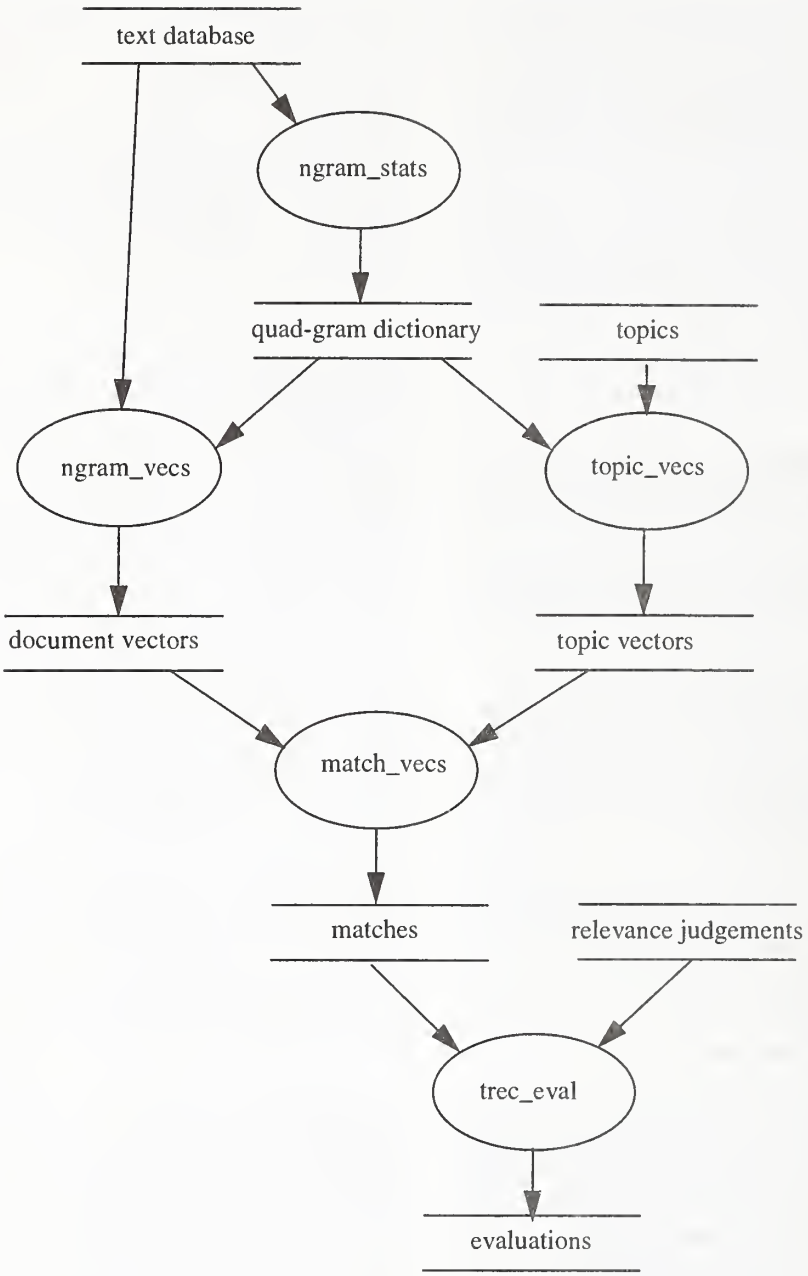


Figure 1: System Dataflow Diagram

words or word parts that word-stemming and stopword elimination would have removed.

3.3 *topic_vecs* And The Topic Vectors

The bubble labeled *topic_vecs* represents a process that is very similar to the *ngram_vecs* process. It reads the topic files and produces for each topic a vector giving the appropriate weights for each quad-gram with respect to that topic. The only significant difference between *topic_vecs* and *ngram_vecs* is that they have to parse two different kinds of SGML structures, and there are some differences in the details of term weight calculations. The format of the actual vectors produced is the same. Figure 2 gives a sample topic and Figure 3 a portion of the corresponding vector for the topic.

3.4 *match_vecs* And Document Retrieval

Once we have a set of document vectors available for every document and topic, we can use them to retrieve documents that are relevant to a particular topic. Following the vector processing model, the process represented by the *match_vecs* bubble performs this retrieval. This process

<top>

<head> Tipster Topic Description

<num> Number: 001

<dom> Domain: International Economics

<title> Topic: Antitrust Cases Pending

<desc> Description:

Document discusses a pending antitrust case.

<narr> Narrative:

To be relevant, a document will discuss a pending antitrust case and will identify the alleged violation as well as the government entity investigating the case. Identification of the industry and the companies involved is optional. The antitrust investigation must be a result of a complaint, NOT as part of a routine review.

...

<\top>

Figure 2: Topic 001 From TREC-1

```
=001
' ANT' 14.851851
'NTIT' 19.151485
'TITR' 32.506756
'ITRU' 31.479759
'TRUS' 20.348030
'RUST' 18.863096
'CASE' 9.288112
'ASES' 3.004144
' PEN' 9.087247
'ENDI' 7.693622
'ISCU' 6.482330
'SCUS' 6.481742
'CUSS' 6.467620
'USSE' 3.795147
'USS ' 5.027708
' IDE' 8.047592
'NTIF' 6.976310
'TIFY' 4.401021
'IFY ' 3.791184
'LLEG' 7.611647
'LEGE' 4.174561
...
```

Figure 3: Portion Of Vector For Topic 001

consists of computing the similarity between the topic's vector and all of the document vectors, and then taking the top 1000 documents by this similarity measure.

For the similarity measure, we are again taking the lead from the classic vector processing model described by [Salton94], which uses the cosine of the angle between the vectors. Since we also normalized all vectors to unit length, this similarity measure thus amounts to computing the dot product between the topic vector and each document vector. This is simple and reasonably efficient given the sparse vector representation. For the full dataset consisting of disks 1 and 2, *match_vecs* typically takes around 20 minutes on a Sun SPARCstation 2 to pass all of the 741856 document vectors and compute the top 1000 scores against a single topic vector. While this is fine for our research task, it is too slow for any realistic use as an interactive tool. However, there are several possible ways to significantly improve search times which we hope to investigate in future versions.

3.5 *trec_eval* And Performance Evaluation

The final bubble labeled *trec_eval* is a program supplied by Cornell University as part of their SMART system, but specifically modified by them to serve as the standard evaluation tool for the TREC task. This program produces some standard reports showing precision and recall values, and is the basis for the comparative reports and graphs in all of the TREC proceedings.

4.0 Term Weighting

In her closing talk at TREC-2, Donna Harman observed that by making simple changes in the term weighting one could drive the performance curve of a good retrieval system down to completely unacceptable levels. Also Salton has explored numerous term weighting schemes, and found that they make very large differences in performance [Salton88]. As we observed earlier, our TREC-2 system used a completely adhoc term weighting scheme that doubtless was a major contributor to its weak performance. For our initial work in TREC-3, we decided to use a simplified version of one of Salton's weighting methods. For documents, our system uses term weights of the form:

$$w_j = \frac{(\log 2\,(tf_j) + 1)}{\sqrt{\sum_i w_i}}$$

where tf_j is the term frequency of the jth quad-gram in the document, and the denominator is a normalization for the document vector for unit length. (log2 refers to the log function in base 2.) For queries, our system uses similar term

weights, but which include the IDF (inverse document frequency):

$$w_j = \frac{(\log 2\,(tf_j) + 1) \cdot IDF_j}{\sqrt{\sum_i w_i}}$$

The precise formula we used for the IDF of a quad-gram was:

$$IDF_j = \log 2\left(\frac{N}{n_j}\right)$$

where N is the number of the documents in the collection, and n_j is the number of documents in the collection containing at least one occurrence of the jth quad-gram. This is about the simplest IDF formulation possible, and we have not had a chance to pursue the effects of using one of the more sophisticated versions. See [Harman92] for a more detailed discussion of the IDF, and alternative formulations other researchers have used.

Following a suggestion by Salton, we use the IDF factor only for the query vector so that the IDF factor appears only once for each term in the final dot product between the document vector and the query vector.

5.0 Assessing Performance

The beauty of the TREC task is that there already is a defined performance criterion in the relevance assessments provided by NIST and the standard *trec_eval* tool for actually calculating the performance results. According to *trec_eval* measures, our system appears to be functioning reasonably well. We ran a number of tests using the earlier TREC data sets and query sets. Results for tests on the dataset from Disks 1 & 2 appear in Table 1. 'Average Precision', 'Precision @ 100 docs', and 'R-precision' are some of the most popular and best understood of the performance measures used by other TREC participants. In the Description field of the table, NG93 and NG94 refer to last year's and this year's N-gram-based retrieval systems, respectively. Q1, Q2, Q3, and Q4 refer to the topic query sets 1, 2, 3, and 4. The shaded rows are the results from this year's (NG94) system. The unshaded rows provide some basis for comparing the performance of NG94 on query set 3 against last year's system and against the median performance of all system in last year's TREC evaluation. NG93 was at the top of the bottom quartile of last year's TREC competitors, but has now moved very close to the median performance of those systems. This is very encouraging, especially in view of the fact that we have not had a chance to further tune the system with regards to term weighting or query enhancements. The row labelled NG94/Q4 gives our results for this year's adhoc test.

TABLE 1. Test Results For Disks 1&2

Description	Average Precision	Precision @ 100 docs	R-precision
NG94/Q1	0.1997	0.3934	0.2874
NG94/Q2	0.2172	0.3988	0.3002
NG94/Q3	0.2614	0.4068	0.3236
NG93/Q3	0.1885	0.3426	0.2494
Median93/Q3	0.2804	0.4354	0.3388
NG94/Q4	0.2061	0.2884	0.2668

Table 2 shows similar results for the dataset on Disk 3.

TABLE 2. Test Results For Disk 3

Description	Average Precision	Precision @ 100 docs	R-precision
NG94/Q2	0.1877	0.2938	0.2502
NG93/Q2	0.1415	0.2524	0.2031
NG94/Q3	0.2528	0.3360	0.3101

Again, comparing the 'NG94/Q2' and 'NG93/Q2' rows shows a significant performance improvement from last year's system to this one.

The row labelled 'NG94/Q3' gives our results for this year's routing test.

Table 3 gives our results from the Spanish test.

TABLE 3. Test Results For Spanish Test

Description	Average Precision	Precision @ 100 docs	R-precision
NG94/SP1	0.4275	0.5258	0.4522

6.0 Conclusions And Future Directions

We have successfully implemented an N-gram-based information retrieval system using the vector processing model. This system uses the frequency of N-gram occurrence in queries, documents, and the entire document collection as a whole to drive its processing. This approach has many advantages, including:

- It provides a robust retrieval system that can tolerate spelling errors in both documents and queries.

- It requires no linguistic pre-processing of documents or queries to perform word-stemming or stopword removal. Thus it is also inherently language independent, as indicated by our early results in processing Spanish documents.

- It allows the system to accrue all of the benefits of the vector processing model, including being able to manipulate documents and queries in a uniform way. For example, it easy to use a retrieved document as a query for a more refined search, such as is necessary for relevance feedback systems.

Using the various TREC datasets, query sets, and relevance assessments, we determined that our new system performed substantially better than last year's adhoc N-gram-based system. Indeed, even in its current simple form, it is competitive with over half of the systems fielded last year.

Besides performing better, the new system, unlike its predecessor, has more obvious directions for improvements. For example, we used only the simplest of term weighting schemes that Salton suggested. It would be straightforward to try some of the more elaborate weighting schemes mentioned in [Harman92] and [Salton88]. Also, all of the high-performing systems from last year included some sort of auxiliary processing using thesauri or semantic networks to enhance or enlarge the queries, and improve their retrieval performance. We have not yet done any of those things, but it seems clear that they would help. Furthermore, as we noted in Section 3.2, one of the real limitations we found was that we had to trade off disk space and performance. It would be very worthwhile to investigate other low-level representation techniques that would allow us to keep more quad-grams with lower IDFs to allow a bit more selectivity.

Finally, although this system is far faster than last year's entry, it still is not usable for true interactive use. We will pursue various approaches for significantly improving the system's retrieval times, including the use of intermediate indexes to speed the selection of relevant vectors to evaluate.

Acknowledgments

We would like to thank the Environmental Research Institute of Michigan for its support during this research, and especially for the use of its computing resources.

References

[Cavnar93a] Cavnar, William B., and Vayda, Alan J., "N-Gram-Based Matching For Multi-Field Database Access In Postal Applications," Proceedings of the Second Symposium on Document Analysis and Information Retrieval, University of Nevada Las Vegas, 1994, pp. 287-297.

[Cavnar93b] Cavnar, William B., "N-Gram-Based Text Filtering for TREC-2," Proceedings of The Second Text REtrieval Conference (TREC-2), D. K. Harman, Editor, National Institute of Standards and Technology, March 1994, pp. 171-179.

[Cavnar94] Cavnar, William B., "N-Gram-Based Text Categorization," Proceedings of the Third Symposium on Document Analysis and Information Retrieval, University of Nevada Las Vegas, 1994, pp. 161-176.

[Harman92] Harman, Donna, "Ranking Algorithms," in Information Retrieval: Data Structures & Algorithms, edited by William B. Frakes and Ricardo Baeza-Yates, Prentice Hall, 1992, pp. 366-376.

[Harman94] Harman, Donna, "Overview of the Second Text REtrieval Conference (TREC-2)," Proceedings of The Second Text REtrieval Conference (TREC-2), D. K. Harman, Editor, National Institute of Standards and Technology, March 1994, pp. 1-20.

[Salton88] Salton, Gerard, and Buckley, Chris, "Parallel Text Search Methods," CACM, Feb. 1988, pp. 202-215.

[Salton94] Salton, Gerard, and Allan, James, "Text Retrieval Using the Vector Processing Model," Proceedings of the Third Symposium on Document Analysis and Information Retrieval, University of Nevada Las Vegas, 1994, pp. 9-22.

[Thomas] Thomas, Timothy, "Document Retrieval From a Large Dataset of Free Text Descriptions of Physician-Patient Encounters via N-Gram Analysis," Los Alamos National Laboratory.

Appendix I: Quad-Gram Frequency Tables

TABLE 1. Top 20 Quad-Grams By Frequency In Data Disks 1 & 2 (English)

Quad-Gram	Total Occurrences	Number of Documents Containing Quad-Gram	Inverse Document Frequency
_THE	20563024	731778	0.019733
THE_	18302404	730260	0.022729
ING_	8240660	690620	0.103247
TION	7751182	687725	0.109307
AND_	7631370	706559	0.070329
_AND	7179946	703706	0.076166
ION_	7125066	686460	0.111964
ATIO	4532871	619104	0.260957
_FOR	4177190	624514	0.248405
ENT_	3985281	592830	0.323521
_PRO	3534605	551068	0.428909
FOR_	3344583	592207	0.325038
MENT	3307425	530937	0.482598
_THA	3293673	510667	0.538756
_COM	3093855	547052	0.439461
HAT_	2981242	486217	0.609539
TED_	2976304	587264	0.337130
_CON	2963004	566401	0.389315
THAT	2775365	480230	0.627414
ONS_	2239260	495977	0.580866

Appendix I Continued

TABLE 2. Top Quad-Grams By Frequency In Spanish Data

Quad-gram	Total Occurrences	Number of Documents Containing Quad-Gram	Inverse Document Frequency
QUE_	1136311	57071	0.011732
_QUE	1084538	57043	0.012440
_CON	651580	56787	0.018929
LOS_	650327	56509	0.026009
_LOS	567903	56147	0.035281
_EST	471028	55709	0.046580
_PAR	428472	55453	0.053224
_DEL	424458	55696	0.046916
NTE_	409750	55510	0.051742
DEL_	406817	55496	0.052106
_POR	387101	55370	0.055385
ENTE	382908	55007	0.064875
LAS_	370691	53887	0.094553
_COM	353891	53944	0.093027
ADO_	346527	54791	0.070551
_LAS	326221	53274	0.111058
POR_	324827	54548	0.076964
DOS_	319534	53112	0.115452
ARA_	310465	53803	0.096803
PARA	308931	53718	0.099084

A Parallel DBMS Approach to IR in TREC-3

David A. Grossman
Office of Information Technology
3E09 Plaza B
Washington, DC 20505
dgrossm1@mason1.gmu.edu

David O. Holmes
AT&T Global Information Solutions
2 Choke Cherry Road
Rockville, MD 20850

Ophir Frieder*
Department of Computer Science
George Mason University
Fairfax, VA
ophir@cs.gmu.edu

Abstract

In this our first year of TREC participation, we implemented an IR system using an AT&T DBC-1012 Model 4 parallel relational database machine. We started with the premise that a relational system could be used to implement an IR system. After implementing a prototype to verify that premise, we then began to investigate the performance of a parallel relational database system for this application. We only used the category B data, but our initial results are encouraging as processing load was balanced across the processors for a variety of different queries. We also tested the effect of query reduction on accuracy and found that queries can be reduced prior to their implementation without incurring a significant loss in precision/recall. This reduction also serves to improve run-time performance.

Finally, in a separate set of work, we implemented Damashek's n-gram algorithm for n=3 and were able to show similar results as found when n=5.

1 Introduction

For TREC-3, we implemented relevance ranking queries using unchanged SQL on an AT&T DBC-1012 (formerly Teradata) parallel database machine [4]. The purpose of this implementation was to test the following hypotheses:

- A relational system that implements standard SQL, may be used as the search engine for an information retrieval application.
- A parallel relational database machine will use an optimizer to balance the workload across multiple processors. The result will be a scalable system such that required levels of performance may be achieved with the purchase of additional hardware.
- Query reduction based on term frequency counts will dramatically improve performance without a significant degradation in accuracy.

*This work supported in part by the National Science Foundation under contract number IRI-9357785.

We have found that each of these hypothesis are true as our relational implementation provides scalable performance. Additionally, query reduction improved run time performance without a significant degradation in accuracy.

Section 2 describes related prior work that serves as the foundation behind our hypothesis. The use of the relational DBMS to model an inverted index is described in Section 3. Our means of computing a measure of relevance is described in in Section 3.3. Section 4 and Section 5 describe our runtime and accuracy results. Conclusions and suggestions for future work are given in Section 6.

2 Prior Work

We briefly describe the prior work that provides the motivation behind these hypotheses. The use of a relational system that would serve as a search engine for an IR application was first proposed by Blair [5]. In this work, it was mentioned that SEQUEL (a precursor to SQL) could be used to perform boolean keyword retrievals such as "Find all documents that contain the word 'terrorist'. " Later, Macleod presented several SEQUEL queries that performed keyword searches using unchanged SEQUEL. Additional operators were then described that could achieve relevance ranking of a set of documents to a query. Research continued in the use of the relational model as a means of providing a more robust form of information retrieval.

Most research in the area stopped when user-defined operators were defined to address certain "application specific" operations [3]. Any function required by an application that does not exist in the database system may be incorporated via a user-defined operator. Stonebraker, et al, examined the usefulness of user-defined operators in assisting a text editing application [10]. A more recent thesis was devoted to the use of user-defined operators to provide typical information retrieval functionality such as keyword searches and proximity searches [8]. Additionally, enhancements to the database optimizer were analyzed that would allow for query optimization of these user-defined operators.

We have developed algorithms using unchanged SQL that implement vector space relevance ranking, proximity searches, and Boolean retrieval. Additionally, we computed worst case disk I/O estimates for a relational implementation and a traditional IR implementation [7]. It was this work that led us to the realization that query reduction based on term selectivity would dramatically affect I/O. For TREC-3, we were able to test different levels of query reduction based on term frequency and verified the impact on disk I/O and accuracy.

3 Implementation Details

We now discuss the approach used to migrate the category B portion of the TIPSTER collection to a set of relations. The relations effectively model an inverted index which may be used to efficiently query the database. The steps used to move text to the relational model are preprocess, load, and index.

3.1 Preprocess

We have developed a text preprocessor that accepts SGML marked text as input and produces three files as output. The preprocessor applies text formatting rules for special characters as described in [1, 2]. Subsequently, for each document, the document frequency for each distinct term is computed and written to a flat file. After all of the documents have been processed, a list of each distinct

term is identified and the *inverse document frequency* for each term is computed. Implementation details of the preprocessor are found in [9].

3.2 Create

Relations are created on the DBC-1012. A clustered primary key ensures that the data are stored in a fashion such that all tuples for a distinct term are on contiguous data pages. Some of our experiments suggest that this approach seemed to best minimize I/O. Other research continues to investigate the best placement strategy for these data [11].

Additionally, FALLBACK mode is used for each of the relations. This results in a full replica of all the data so that the system has resilience to a disk failure.

3.2.1 Load

The DBC-1012 FASTLOAD utility is used to move the data from the flat files (residing on an Intel 80486 based machine) to the DBC-1012. The utility makes use of all of the processors and ensures that data are distributed to each processor in the fashion prescribed by the clustered index. We typically found that the FASTLOAD was able to load our largest relation at a speed of 857 rows per second. By comparison, work we have done on an Intel Pentium based processor running Microsoft SQL Server on Windows NT typically loads data (using the Bulk Copy facility) at a rate of 381 rows per second.

3.2.2 Query processing

For TREC-3, we only addressed the ad-hoc queries (Topics 151-200). The queries were preprocessed and loaded into corresponding relations using a similar method as done for the data.

3.2.3 Example

The following example illustrates our process of starting with an input document in a TIPSTER file and completing with the properly loaded relations.

Consider the following document from the TIPSTER data (a final sentence has been added to assist our presentation):

<DOC>
<DOCNO> WSJ870323-0180 <DOCNO>
<HL> Italy's Commercial Vehicle Sales <HL>
<DD> 03/23/87 <DD>
<DATELINE> TURIN, Italy <DATELINE>
<TEXT>
Commercial-vehicle sales in Italy rose 11.4% in February from a year earlier, to 8,848 units, according to provisional figures from the Italian Association of Auto Makers.
Sales for the Association are expected to rise an additional 2% in July. <TEXT>
<DOC>

The following relations will be built to model this document:

DOC:

doc_id	*doc_name*	*date*	*dateline*
1	WSJ870323-0180	3/23/87	Turin, Italy

DOC_TERM:

doc_id	*term*	itterm_freq
1	commercial	1
1	vehicle	1
1	sales	2
1	italy	1
1	rose	1
1	11.4%	1
1	february	1
1	year	1
1	earlier	1
1	8,848	1
1	according	1
1	provisional	1
1	figures	1
1	italian	1
1	association	2
1	auto	1
1	makers	1
1	expected	1
1	additional	1
1	2%	1
1	July	1

IDF

term	idf
11.4%	2.9595
2%	1.5911
8848	4.3936
according	0.7782
additional	1.0792
association	1.0792
auto	1.2788
commercial	1.0000
earlier	0.6021
expected	0.6990
february	1.3222
figures	1.2553
italian	1.8451
italy	1.6721
July	1.0414
makers	1.3010
provisional	2.5172
rose	0.7782
sales	0.6990
vehicle	1.7709
year	0.0000

The first relation, *doc* contains a tuple for each document that occurs in the text. The internal document identifier and official TIPSTER document name are stored in this tuple along with any other structured data that has a 1-1 relationship with the document. For TREC-3, we only used the *date* and *dateline* SGML marked fields, but other fields such as *source* could easily be placed into this relation.

The *doc_term* relation models a typical inverted index. An attribute *doc_freq* is used to store the number of occurrences for the term in the document. A compressed internal document identifier is used here, while the official TREC-3 name may be obtained from the *doc* relation.

Finally, the *idf* relation stores the inverse document frequency for each distinct term in the entire document collection. The *idf* is computed as:

$$idf(term) = \log_{10} df$$

where *df* is the number of distinct documents in which the term appears.

3.2.4 Overhead

Typically, storage overhead has been a justification used against relational IR implementations. Given that the document identifier and the term must be replicated numerous times in the *doc_term*

relation, it would appear that storage requirements would be substantial. The following table indicates the storage requirements for each of the three relations. Since the DBC-1012 uses hash-based indices, there is no extra storage required for each index; rather, a fixed 13 byte overhead is assigned for each table to maintain an internal hash identifier.

Storage Overhead

Name	Tuples	Megabytes
Doc	173,252	29.6
Idf	389,797	32.7
Doc_Term	33,497,912	2,037.5

For the 534 megabytes found in Category B data the relational structures required to implement it required 2.1 Gigabytes of storage. However, the DBC-1012 replicates all data to provide real time protection for a disk failure. Without this replication, only 1.05 Gigabytes would be required. The overhead ratio is 1.97:1.00.

3.3 Query Processing

Having constructed the relations, we were able to implement a variety of experiments to learn more about performance of the DBC-1012 for this application. To learn more about accuracy, we reduced the size of TIPSTER queries based on the precomputed *idf* values. The premise was that a very frequently occurring term increases the I/O but decreases the accuracy of the query. Hence, this form of query reduction is based on the same premise as a stop word list and can be viewed as a tailored stop word list.

3.4 Building the Query Threshold Relation

A new query relation is generated called *query_threshold*. The original *query* relation is joined with the *idf* relation such that the *idf* is obtained for each term in the query. The terms in the query are then sorted in decreasing order of their *idf*.

The relation is then exported to a flat file and a simple utility is invoked to determine the number of terms in the query and develop new queries based on a given threshold. The threshold indicates the percentage of terms that are found in the new query. A threshold of ten indicates that the new query contains ten percent of the terms in an unmodified query. For a one hundred term query, a threshold of ten would result in a query composed of those terms that are ranked (by *idf*) one through ten. We developed a *query_threshold* relation that contains tuples that represent queries for thresholds of .1, .2, .25, .3, .33, .5, and 1.0.

3.5 Implement the Query with Unchanged SQL

We have previously identified queries that use standard SQL to implement both the inner product and cosine measures of relevance [7]. These queries used a *query* relation as described in Section 3. A slight modification is required to implement different query thresholds. Additionally, we found performance improved when we denormalized the *query_threshold* table to contain the *idf* as well. The *doc* table is used to obtain the official doc name after the internal document identifier is matched. The query used to compute a vector inner product between a query and all document vectors for the TREC-3 data is:

```
SELECT c.doc_name, sum(q.cntidf*b.term_freq)
  FROM QUERY_THRESHOLD a, DOC_TERM b, DOC c
 WHERE a.term = b.term AND
 b.docid = c.docid AND
         THRESHOLD = ? AND
 QUERY_NUM = ?
 GROUP BY b.doc_id
 ORDER BY 2 DESC
```

4 Run Time Performance

We developed macros to execute queries 151-200 for threshold of .10, .20, .25, .30, .33, .50, and 1.0. Each query was run with one and three concurrent sessions. The reason for this is that the processors were only being used at a capacity of between twenty and thirty percent with a single session. Increasing the number of concurrent sessions gives the machine more work to do and results in increased throughput.

Figure 1 gives the average response time for all fifty queries for each of the query thresholds. Separate lines depict results for one and three sessions. Figures 2 and 3 provide CPU and disk I/O results. It can be seen that for thresholds less than .5, workload increases linearly, and over .5, we experience an exponential change. This is consistent with Zipf's law [12].

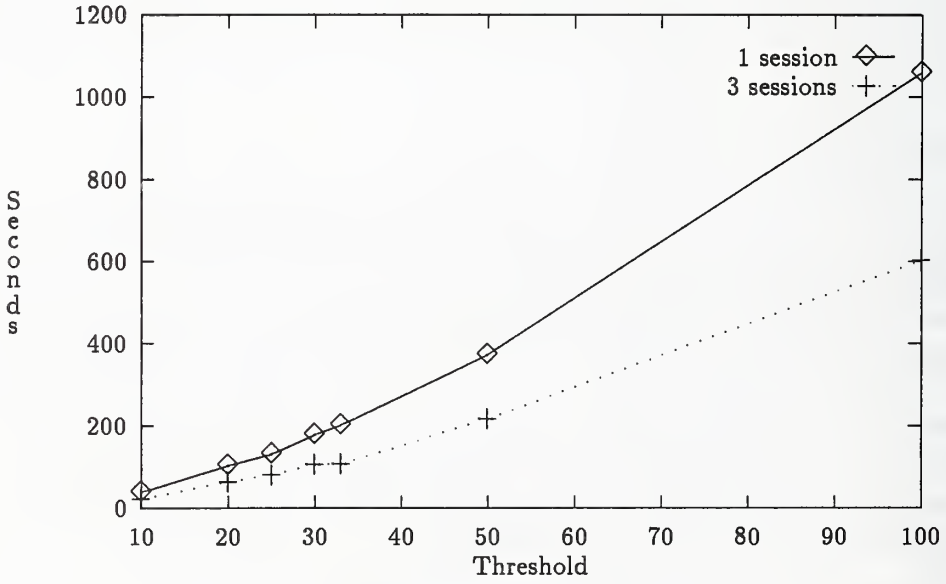


Figure 1: Avg. Response Time for Varying Query Thresholds

To further illustrate the cause of the exponential increase in run time performance, Figure 4 provides the number of tuples found in the *doc_term* relation that match a term in the query. Again, the behavior is the same, and we can see that thresholds of below .5 runs substantially faster than over .5.

Finally, we measured the amount of load balancing that takes place in using the four pro-

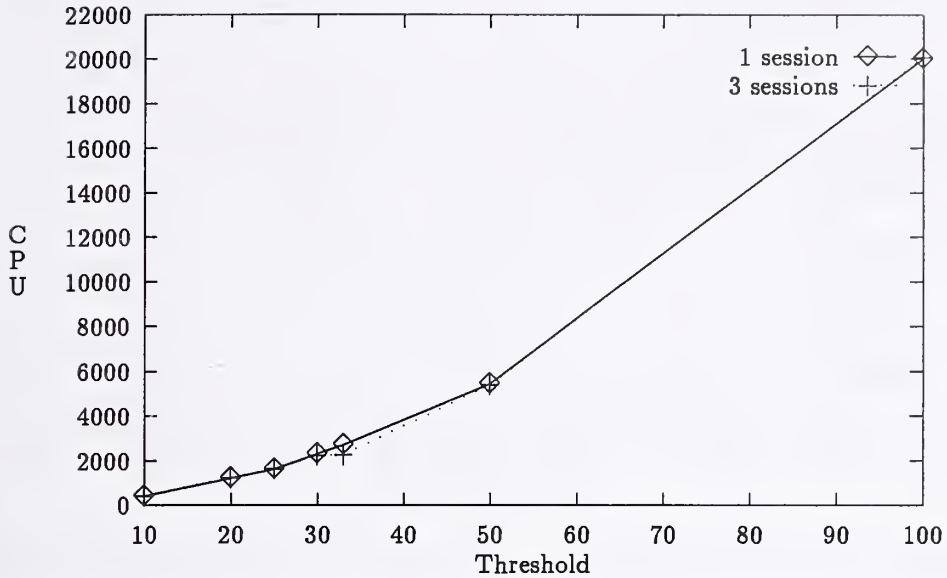


Figure 2: Avg. CPU Time for Varying Query Thresholds

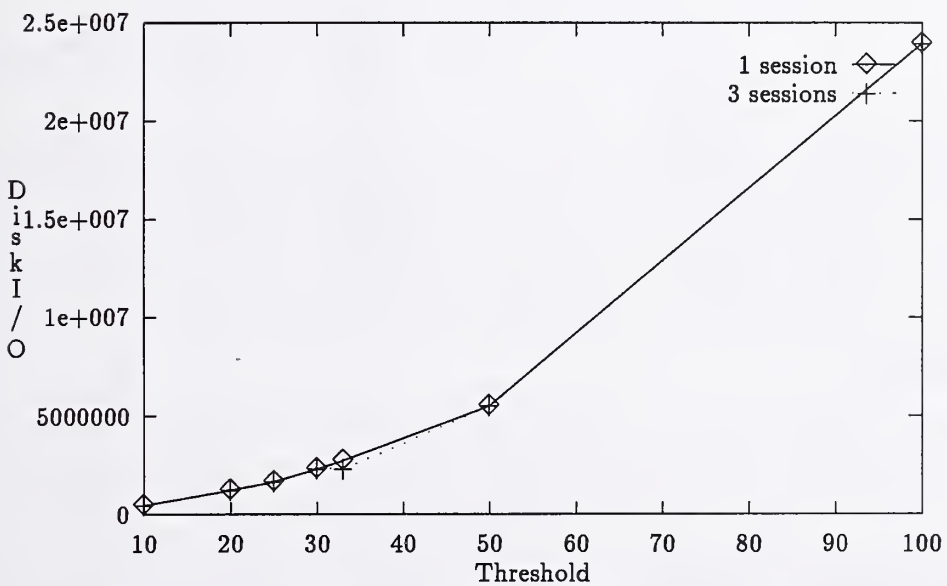


Figure 3: Total Disk I/O for Varying Query Thresholds

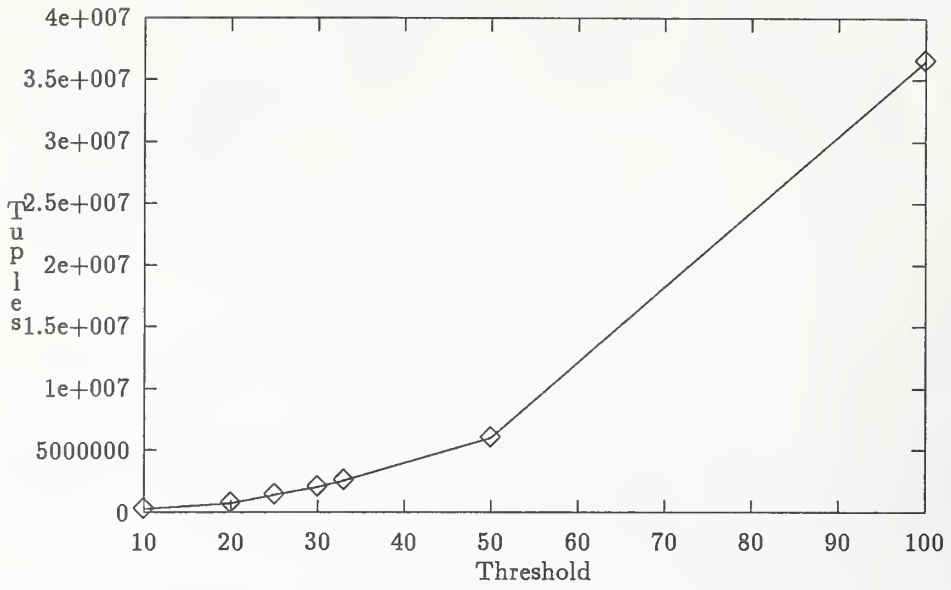


Figure 4: Total tuples in DOC_TERM accessed for Varying Query Thresholds

cessors on the DBC-1012. The table below indicates the amount of processor imbalance for CPU time: $\frac{maxcpu-mincpu}{avgcpu}$ measured at each query threshold. It can be seen that for all workloads, the processors are less then ten percent out of balance. Given this high degree of load balancing, it is reasonable to suspect that additional processors may be added to achieve required response time.

Percent of Processor Imbalance :

threshold	*CPU Time (1 session)*	*CPU Time (3 sessions)*
10	7.36	7.88
20	8.39	7.29
25	9.38	4.81
30	4.79	2.02
33	5.02	3.02
50	8.19	2.38
100	5.13	4.51

5 Accuracy

We measured precision/recall for each of the queries 151-200 for thresholds of .10, .25, .33, .50, and 1.0. Our hypothesis was that higher thresholds would result in reduced precision/recall as the query would be searching for terms that were very common across the corpus and would do little for differentiating a relevant document from an irrelevant document.

Figure 5 illustrates the number of relevant documents that were retrieved for all fifty queries using varying thresholds. Separate lines for result sets of size 100 and 200 are presented. It can be seen that as the threshold increases from ten to twenty-five, more relevant documents are found. This is reasonable to expect as a threshold of ten or twenty may omit many query terms that assist in identifying relevant documents. As the threshold increases beyond twenty-five the number of

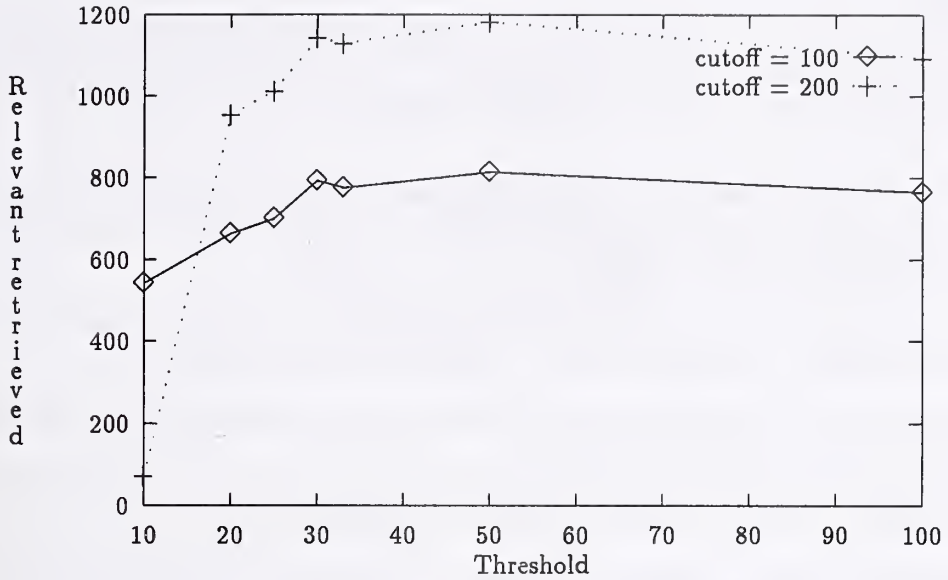


Figure 5: Effect on Query Reduction on Number Docs Retrieved that are Relevant

relevant documents retrieved drops. This is also understandable as a higher threshold will result in increased noise within the query. Hence, we have found that a query threshold of between twenty-five to thirty-three may yield good accuracy as well as dramatically improved run time performance over a query that is not filtered at all (one hundred percent threshold). These results are unofficial as they were determined after the TREC-3 submission deadline.

6 Conclusions and Future Work

Our TREC-3 effort has served as a proof-of-concept of our prior ideas that an unchanged relational DBMS may be used to serve as an engine for IR. Our experiments found that overhead was somewhat high, but tolerable for a large scale machine. An overhead of 1.97:1.00 may be reasonable for applications that have performance requirements large enough to justify a large scale parallel architecture. Given the lack of scalable parallel algorithms for parallel information retrieval, our work may be one means of easily spreading workload across large numbers of processors.

Another advantage of our approach is that structured data and text may be easily integrated. Although we only used *date* and *dateline* in our prototype, other structured fields could have easily been used. Queries that integrate both structured data and text are relatively straightforward extensions to the queries we have discussed.

We have also shown that query reduction using term frequencies may be a viable means of reducing disk I/O without significantly affecting accuracy.

Finally, it should be noted that the results given here are unofficial. Our official results came as a result of implementing Marc Damashek's approach using n-grams of size three instead of size five [6]. Interestingly, our results for the ad-hoc collection were markedly similar to those submitted by Damashek's group. We plan to incorporate this work into our relational implementation by developing algorithms that implement this work using unchanged SQL.

References

[1] E. Adams. *A Study of Trigrams and Their Feasibility as Index Terms in a Full Text Information Retrieval System.* PhD thesis, George Washington University, Department of Computer Science, 1991.

[2] E. Adams. Trigrams as index elements in full text retrieval observations and experimental results. *Proceedings of the First Annual Conference on Information and Knowledge Managementnt (CIKM '92)*, October 1992.

[3] M. Stonebraker Jeff Anton and Eric Hanson. Extending a database system with procedures. *ACM Transactions on Database Systems*, 12(3):350–376, September 1987.

[4] AT&T Global Information Systems. *Teradata DBC-1012 Concepts and Facilities*, March 1992.

[5] D. Blair. Square (specifying queries as relational expressions) as a document retrieval language). Written while working on the System R project., 1974.

[6] M. Damashek. Gauging similarity via n-grams: Text sorting, categorization, and retrieval in any language. Submitted to Science, 1994.

[7] D. Grossman. Using the relational model and part-of-speech tagging to implement text relevance. *Proceedings of the First Annual Conference on Information and Knowledge Managementnt (CIKM '92)*, October 1992.

[8] C. Lynch and M. Stonebraker. Extended user-defined indexing with application to textual databases. *Proceedings of the 14th VLDB Conference*, pages 306–317, 1988.

[9] E. Pulley. A preprocessor for integrating structured data and text. Technical Report CfIA-94-003, Center for Image Analysis, George Mason University, 1994.

[10] M. Stonebraker, H. Stettner, N. Lynn, J. Kalash, and Antonin Guttman. Document processing in a relational database system. *ACM Transactions on Office Information Systems*, 1(2):143–158, April 1983.

[11] A. Tomasic and Hector Garcia-Molina. Performance of inverted indices in shared-nothing distributed text document information retreival systems. In *Proceedings of the 2nd International Conference on Parallel and Distributed Information*, pages 8–17, 1993.

[12] G.K. Zipf. *Human Behaviour and the Principle of Least Effort.* Addison-Wessley, 1949.

Research in Automatic Profile Creation and Relevance Ranking with LMDS

Julian A. Yochum

Logicon, Inc.
2100 Washington Blvd.
Arlington, VA 22204-5703
jyochum@logicon.com

Abstract

This paper describes the development of a prototype system to generate routing profiles automatically from sets of relevant documents provided by a user, and to assign relevance scores to the documents selected by these profiles. The prototype was developed with the Logicon Message Dissemination System (LMDS) for participation in the Third Text REtrieval Conference (TREC-3).

Each generated profile contains two sets of terms: a very small set to select documents, and a much larger set to assign a relevance score to each document selected. The profile generator chooses each term and assigns a weight to it, based on its frequency of occurrence in the set of documents provided by the user, and on its frequency of occurrence in a large representative corpus of documents. The LMDS search engine uses the resulting profiles to select documents, and then passes the documents to the scoring prototype for ranking. The score assigned is a function of the weights of all profile terms found in the document.

Performance figures and TREC-3 results are included.

1. Introduction

LMDS is a commercial off-the-shelf (COTS) product, designed specifically for high-speed document routing on a wide range of hardware and operating system platforms. LMDS users create interest profiles to specify the types of documents that they wish to receive. Each profile contains a list of tokens (or "search terms") which may appear in such a document, together with a Boolean expression indicating the logical combination in which the tokens must occur for LMDS to select the document. In addition, the user may optionally attach a weight to each search term.

When LMDS selects a document, it passes the document to a scoring routine, together with pointers to the set of search terms discovered in the document, and to their associated weights. In addition, LMDS provides an API which allows any installation to customize the scoring routine for its own purposes. The resulting document scores are then used to order the documents for display.

A feature not yet available in LMDS is the ability to generate profiles automatically, based on a sample of documents which the user deems relevant to a specific topic. Since the relevance judgments associated with the TREC-3 training documents provide 50 such sample sets, our objective for TREC-3 was to use these sample sets to develop a prototype system for automatic profile generation and relevance ranking.

LMDS is designed to run thousands of profiles against incoming documents in a minimum amount of time and with minimum hardware requirements. To qualify as workable enhancements to LMDS, therefore, any algorithms for automatic profile generation and relevance ranking must be not only effective, but also compact and fast. The prototype system was designed with these goals in mind, and a hardware configuration was used which reflected typical real-world resource constraints:

Processor:	Dedicated SPARCstation IPC
CPU Clock:	25 MHz
RAM:	24 MB
Hard Disk:	4 GB

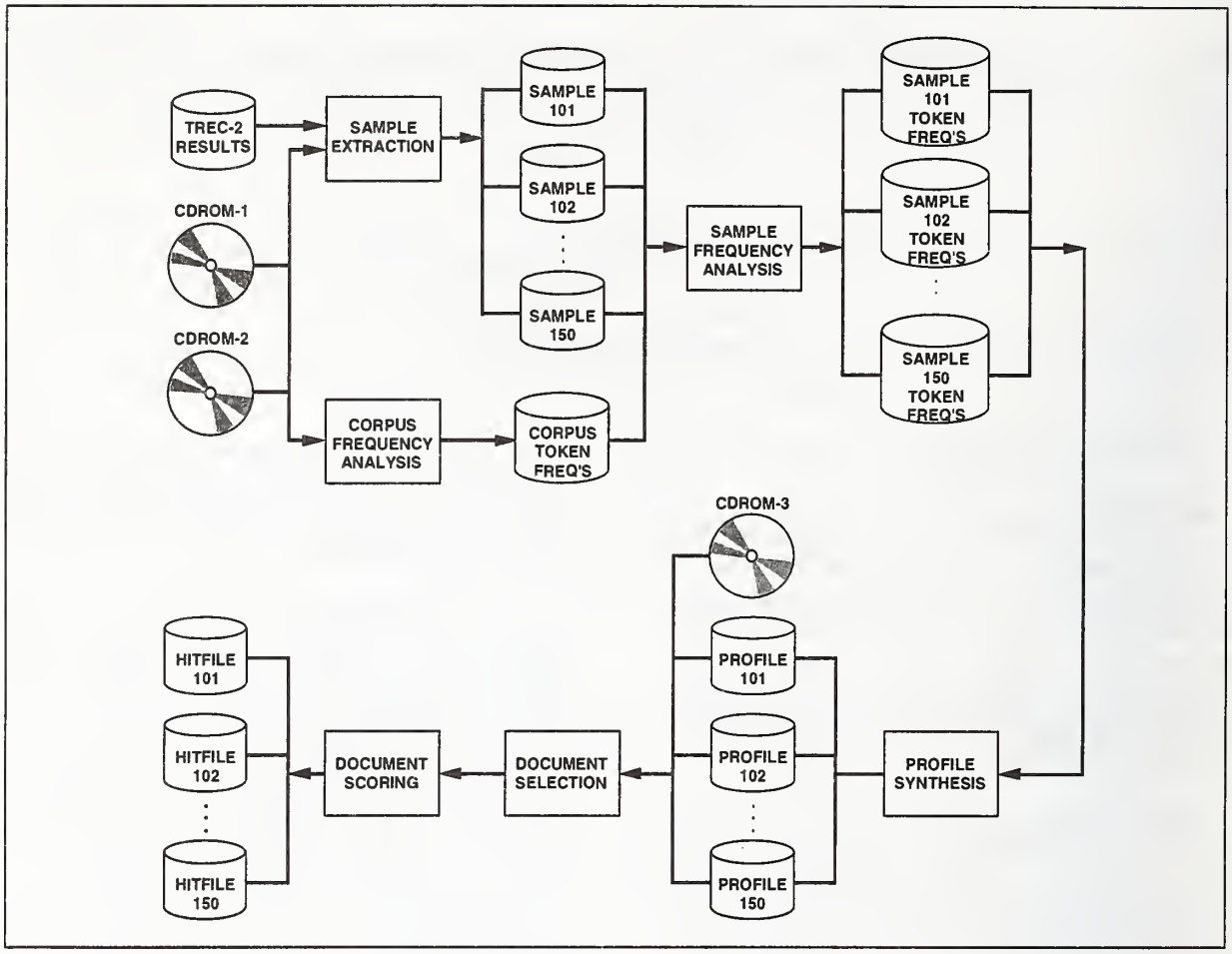


Figure 1: System Data Flow

2. Problem Description

Routing participants in TREC-3 were provided with 742,000 training documents, 2.1 gigabytes on two CDROM disks. Documents were selected from the *Wall Street Journal*, *AP Newswire*, *Computer Select* disks, *Federal Register*, and DOE abstracts, and were stored with embedded SGML formatting tags. Participants also received 50 routing topics, each a full-page description of a user's information needs, together with 50 sets of document numbers (TREC-2 retrieval results) identifying the training documents which satisfied each topic.

Participants were to use the above data to generate routing profiles and to develop relevance scoring algorithms. They were then provided with 336,000 test documents, 1.1 gigabytes on another CDROM. They were to run their routing profiles against the test documents to identify the 1,000 highest-scoring documents for each topic. They were then were to sort the set of hits for each topic in descending order of relevance, and were to return the 50 sets of 1,000 hits for evaluation by TREC-3 judges.

3. System Overview

As shown in Figure 1, the prototype system consists of six separate processes. For TREC-3, each had its own specific purpose:

1. Corpus Frequency Analysis analyzed a large representative corpus of documents to create a database of token-frequency data.

2. Sample Extraction used the TREC-2 results to extract all documents relevant to each routing topic from CDROM-1 and CDROM-2, and created a database of topic samples.

3. Sample Frequency Analysis analyzed the sample set of documents associated with each topic to determine the set of tokens that were

statistically most descriptive of the sample, and assigned a weight to each token based on its observed frequencies of occurrence in the sample and in the corpus.

4. Profile Synthesis incorporated the most descriptive tokens into a LMDS profile, appended a Boolean logic statement to indicate which tokens would be used for document selection, and sent the completed profile to LMDS for activation.

5. Document Selection used the LMDS routing engine to dispatch each active profile against each document on CDROM-3. Whenever the Boolean logic statement in a profile was satisfied by the document, LMDS sent the document to the Document Scoring process, together with the token and weighting data necessary to score the document.

6. Document Scoring calculated a document score for each profile which selected the document, based on the weights of all profile terms that were found in the document. The scoring routine then stored the document and its score in a profile-designated hitfile for TREC-3 evaluation.

The details of each of the above processes are discussed below in the context of TREC-3. Performance figures for each process are provided.

3.1. Corpus Frequency Analysis

This preliminary process analyzed all training documents on CDROM-1 and all training documents with relevance judgments on CDROM-2. (This total set of documents will be referred to hereafter as the "corpus.") For each document, the process tokenized all alphanumeric strings, and then eliminated those tokens which:

1. Were only one character long.
2. Contained one or more numeric characters.
3. Were in the stopword list.
4. Served as SGML tags.
5. Occurred only once in the corpus.
6. Were in an area of the document deemed non-searchable by TREC-3 rules.

The process next counted the number of documents in which each token occurred, then sorted the tokens alphabetically and stored them in a file with their associated document counts. A portion of this file is shown in Figure 2. The following figures summarize the performance of the process:

487,000	Documents analyzed
171	Stopwords used
241,000	Tokens identified
0.79	Wall-clock seconds/document, average

3.2. Sample Extraction

This preliminary process used the TREC-2 results to extract all relevant documents for each topic from CDROM-1 and CDROM-2, and to store these sets of documents in 50 separate directories.

```
CORPUS SIZE: 487013

CORPUS
 TOKEN  TOKEN
  DOCS  STRING

   506  aa
   126  aaa
     2  aaaa
     2  aaac
        .
        .
        .
    99  privatisation
    14  privatise
    38  privatised
    16  privatising
   920  privatization
    93  privatizations
   239  privatize
   300  privatized
     2  privatizer
     7  privatizes
   141  privatizing
        .
        .
        .
    57  zz
     5  zzz
    60  zzzz
     7  zzzzbest
```

Figure 2: Corpus Analysis Results

TOPIC ID: 128
SAMPLE SIZE: 405
CORPUS SIZE: 487013

TOKEN NUMBER	BINOMIAL PROBABIL PERCNT	TOKEN RECALL PERCNT	TOKEN PRECIS PERCNT	SAMPLE TOKEN DOCS	CORPUS TOKEN DOCS	TOKEN STRING
1	0.00000000	70.1	30.870	284	920	privatization
2	0.00000000	58.8	1.125	238	21152	sale
3	0.00000000	53.6	1.386	217	15656	owned
4	0.00000000	91.1	0.556	369	66405	government
5	0.00000000	21.0	65.385	85	130	denationalizat
6	0.00000000	52.3	1.010	212	20983	tnm
7	0.00000000	19.0	25.667	77	300	privatized
8	0.00000000	44.2	1.242	179	14414	mergers
9	0.00000000	43.7	1.107	177	15995	tender
10	0.00000000	44.0	1.082	178	16445	acquisitions
.						
.						
.						
998	0.00094249	11.4	0.161	46	28588	best
999	0.00094302	2.0	0.657	8	1217	indefinitely
1000	0.00095270	15.1	0.144	61	42319	cost
.						
.						
.						
3287	17.64646237	1.2	0.084	5	5925	donald
3288	17.65115527	1.2	0.084	5	5950	replaced
3289	17.65598058	1.2	0.083	5	6007	fundamental

Figure 3: Sample Analysis Results for Topic 128

Each directory thus contained the sample for a given topic. No figures were kept on the time required to perform this extraction, but sample sizes varied from a low of 28 documents to a high of 792 documents.

3.3. Sample Analysis

This process was responsible for producing the sample statistics necessary for automatic profile generation. For each topic sample, the process first tokenized each document in the sample, using the same tokenizing rules as Corpus Frequency Analysis. The process then counted the number of sample documents in which each token occurred. The process then combined the sample count for each token with the corpus count for that token to calculate the binomial probability distribution *P(r)* for the token, as shown in the following formula:

$$P(r) = \frac{n!}{r!(n-r)!}\left(\frac{p}{q}\right)^r\left(1-\frac{p}{q}\right)^{n-r}$$

where:

n = documents in sample
r = sample documents containing token
p = corpus documents containing token
q = documents in corpus

Calculated in this way, the value *P(r)* can be used as a measure of how "descriptive" each token is with regard to a given sample of documents, with lower values indicating greater descriptive power.

The process next calculated the weight *w* for each token, as shown in the following formula:

$$w = \frac{r}{p}$$

```
! Profile:  P_128X_999.DIS
! Topic:    128

! This is a machine-generated profile, based on an analysis
! of the word frequencies in a sample of relevant documents,
! and of the word frequencies in the corpus as a whole.

DISSEM:  jy_1/C-128X-999
SOURCE:  all

TERMS:

1     DOC CONTAINS      privatization     (H 30870)
2     DOC CONTAINS      sale              (H 1125)
3     DOC CONTAINS      owned             (H 1386)
4     DOC CONTAINS      government        (H 0556)
5     DOC CONTAINS      denationalizat    (H 65385)
6     DOC CONTAINS      tnm               (H 1010)
7     DOC CONTAINS      privatized        (H 25667)
8     DOC CONTAINS      mergers           (H 1242)
9     DOC CONTAINS      tender            (H 1107)
10    DOC CONTAINS      acquisitions      (H 1082)
          .
          .
          .
998   DOC CONTAINS      best              (H 0161)
999   DOC CONTAINS      indefinitely      (H 0657)
1000  DOC CONTAINS      cost              (H 0144)

LOGIC:  ANY OF 1 THRU 10
```

Figure 4: Automatically Generated Profile for Topic 128

The process then created a table containing each unique token in the topic sample, together with its associated values for $P(r)$ and w. Finally, the process sorted this table on $P(r)$, the measure of descriptiveness.

A portion of one such table is shown is Figure 3. Its rows contain the statistics for each unique token in the topic sample. Its columns contain the information described below:

1. Token Number is a one-up token identification number.

2. Binomial Probability Percent is the value $P(r)$ expressed as a percent -- the probability that this token has occurred in this sample as often as it has, purely by chance.

3. Token Recall Percent is the value:

$$\frac{r}{n}$$

expressed as a percent -- the probability that this token will occur in a relevant document.

4. Token Precision Percent is the value w expressed as a percent -- the probability a document will be relevant if it contains this token.

5. Sample Token Documents is the value r -- the total sample documents containing the token.

6. Corpus Token Documents is the value p -- the total corpus documents containing the token.

7. Token String is the first 14 characters of the token.

3.4. Profile Synthesis

To create the actual LMDS profile, this process concatenated the top 1000 tokens with their respective weights from the sorted table, and placed them in a file. It then appended a Boolean logic statement containing the top 10 tokens in an OR condition, indicating that a document was to be selected if it contained any one of the top 10 most descriptive tokens. (Once a document was selected, however, the scoring algorithm would assign a document score based on the weights of all profile tokens that appeared in that document.) The process then sent the new profile to LMDS to be activated. A portion of one such profile is shown in Figure 4.

The following figures summarize the combined performance of the Sample Analysis and Profile Synthesis processes:

50 Profiles generated
6.1 Wall-clock minutes/profile, average

3.5. Document Selection

Document selection was performed with the LMDS 2.13 routing engine, which enabled each profile to perform a free-text scan of each document on CDROM-3.

Whenever the Boolean logic statement in any profile was satisfied by a document, LMDS sent the document to the Document Scoring process, together with the token and weighting information needed by that process to score the document.

The LMDS program product itself was not modified in any way for TREC-3. All prototype functionality was added via externally-generated LMDS profiles and via the LMDS API.

3.6. Document Scoring

For each document and set of profile tokens and weights received from LMDS, this process first tokenized the document, using the same tokenizing rules as Corpus Frequency Analysis. The process then counted the number of unique tokens in the document, and calculated the score s for the document, according to the following formula:

$$s = (1-(1-w_1)(1-w_2)...(1-w_n))\frac{u}{t}$$

where:

w_i = *weight of each unique profile token in doc*
u = *unique profile tokens in doc*
t = *unique tokens in doc*

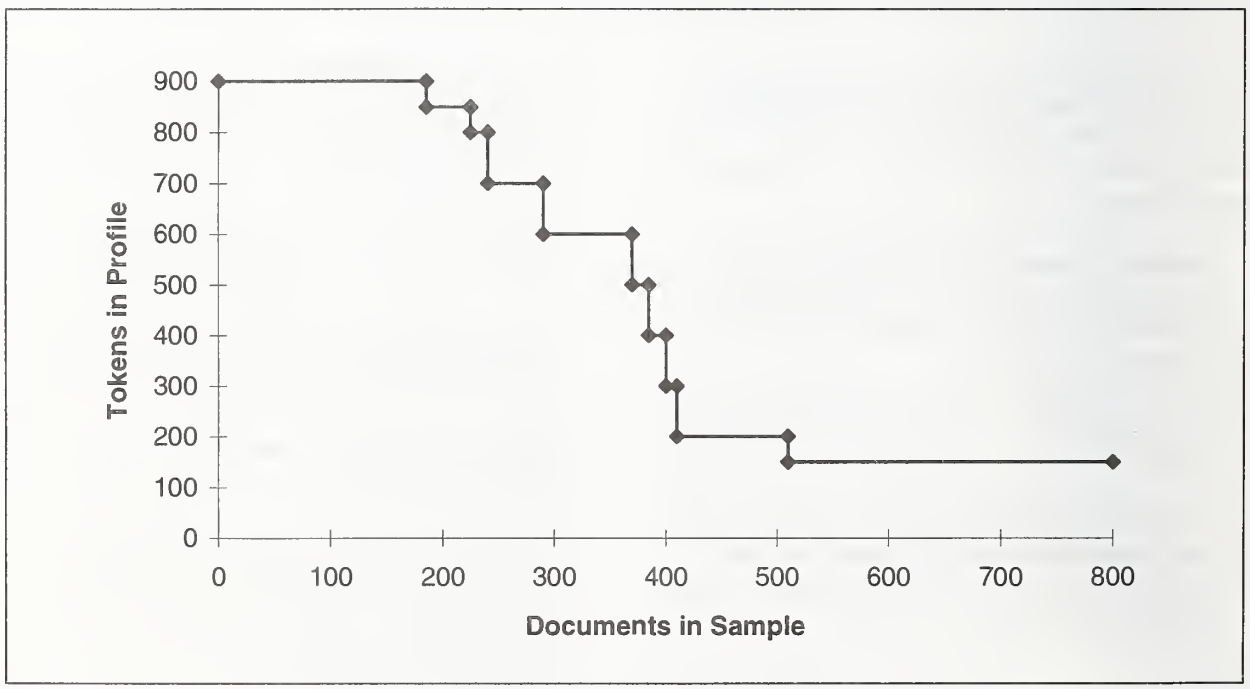


Figure 5: Observed Optimum Relationship of Sample Size to Profile Size

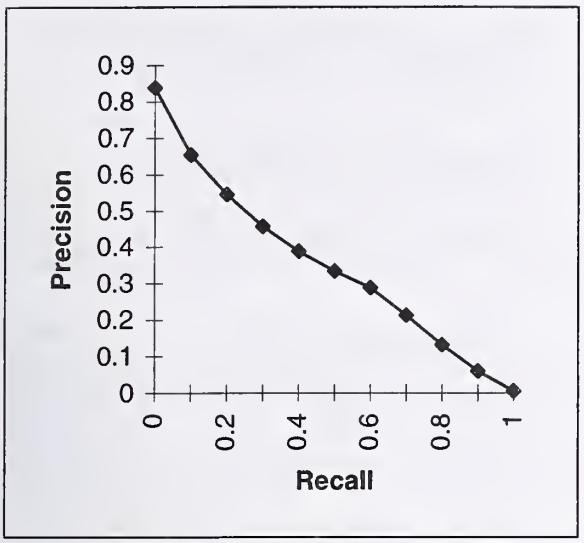


Figure 6: Recall-Precision Curve

The first part of this formula calculates the probability that this document will be relevant to a topic, given that the document contains a particular subset of the topic's most descriptive tokens. Multiplication by the ratio of *u/t* then corrects for the tendency of larger documents to achieve higher scores simply by having more tokens.

Although each profile contained 1,000 tokens, it was discovered during testing with trial documents that as the sample size increased, the accuracy of the profile usually decreased. Further testing suggested that the optimum number of profile tokens indeed was inversely proportional to the size of the sample. Accordingly, the Document Scoring process was modified to calculate *s* with only the top *x* tokens in the profile, where *x* was the value of a step function on the sample size, as shown in Figure 5.

When scoring was complete for the profile, the process stored the document number and its score into the hitfile designated by the profile. When all documents on CDROM-3 had been processed in this fashion, the 50 hitfiles were sorted on document score, and the top 1,000 entries in each hitfile were sent to the TREC-3 judges.

The following figures summarize the combined performance of the Document Selection and Document Scoring processes:

336,000	Documents processed
0,99	Wall-clock seconds/document, average

```
Total number of documents
over all queries:
   Retrieved:    50000
   Relevant:      9353
   Ret AND Rel:   6756

Interpolated recall-precision:
   At 0.00:  0.8387
   At 0.10:  0.6539
   At 0.20:  0.5450
   At 0.30:  0.4576
   At 0.40:  0.3885
   At 0.50:  0.3344
   At 0.60:  0.2877
   At 0.70:  0.2122
   At 0.80:  0.1316
   At 0.90:  0.0606
   At 1.00:  0.0055

Non-interpolated precision
over all relevant documents:
   0.3373

Precision:
   At    5 docs:  0.5840
   At   10 docs:  0.5760
   At   15 docs:  0.5747
   At   20 docs:  0.5560
   At   30 docs:  0.5253
   At  100 docs:  0.4288
   At  200 docs:  0.3540
   At  500 docs:  0.2168
   At 1000 docs:  0.1351

R-Precision:
   0.3725
```

Figure 7: Routing Results for Prototype

4. TREC-3 Results

The recall-precision scores for the prototype are summarized in the graph in Figure 6, and a table of detailed scores is provided in Figure 7.

Except for the first section of Figure 7, all results are averages of the corresponding scores for each of the 50 topics. While most sections of Figure 7 are self-explanatory, several concepts may require additional explanation:

1. Interpolated precision is the maximum precision over a range of recall points. Thus, the interpolated precision at recall 0.10 (i.e., after 10% of relevant documents have been retrieved

for a query) is the maximum precision at all recall points >= 0.10.

2. Precision at X docs is the precision after X documents have been retrieved, whether or not the documents are relevant.

3. R-Precision is the precision after R documents have been retrieved, where R is the number of relevant documents possible for a topic. Thus, if a topic has 50 possible relevant documents, precision is measured for that topic only at 50 documents.

5. Analysis

As stated in the Introduction, to qualify as workable enhancements to LMDS, any algorithms for automatic profile generation and relevance ranking must be not only effective, but also compact and fast. The prototype is evaluated below with regard to those criteria:

1. Effectiveness -- The prototype achieved aggregate scores (Figures 6 and 7) slightly above average across all measures, a strong showing for a first-time participant in TREC-3. These scores can now be used a baseline for developing improved versions of the prototype.

2. Compactness -- The only element of the prototype of any notable size is the 5-megabyte file containing the results of Corpus Analysis. Even allowing for some growth in a production environment, its costs in disk storage would remain insignificant. Moreover, the entire file does not need to be in RAM at any one time.

3. Speed -- Average wall-clock times are recapped below for those processes which could be ongoing in a production environment:

Corpus Frequency Analysis:	0.79 seconds/doc
Sample Analysis plus Profile Synthesis:	6.1 minutes/prof
Document Selection plus Document Scoring:	0.99 seconds/doc

Of these, only the figure of 6.1 minutes for profile generation is unacceptable. The profile generation software, however, is quite inefficient, and a few simple code modifications should produce a substantial increase in speed.

6. Anticipated Improvements

Substantial opportunities exist for improving the prototype. A number of these are discussed below:

1. Word-Stemming Algorithm -- Since the weight of a token is directly proportional to the number of times it occurs in a sample, use of a word-stemming algorithm should allow Sample Analysis to more accurately calculate the weight of a word which might appear many times in a sample, but in slightly different forms (e.g., "privatize" in Figures 2, 3 and 4).

2. Larger Stopword List -- Additional stopwords should help exclude from profiles those tokens with no predictive value, and should also improve the *u/t* calculation used by Document Scoring to correct for the tendency of larger documents to score higher simply by having more tokens.

3. AND Logic -- The current approach of selecting documents via an OR condition on the top 10 most descriptive tokens virtually guarantees that the set of documents passed to Document Scoring will contain all relevant documents. It also guarantees the set will contain a very large number that are totally unrelated. Examination of Sample Analysis output, such as that shown in Figure 3, strongly suggests that selecting documents via an AND condition on any 2 of the top 10 tokens would substantially reduce the number of unrelated documents selected without significantly affecting the number of relevant ones.

4. Mathematically Optimized Profile Sizes -- The optimization step function graphed in Figure 5 was derived by analyzing the results of a subset of training profiles and averaging the observed optimum relationships between sample sizes and profile sizes. While a useful tool, it cannot accurately predict the optimum size for every profile. Preliminary evidence suggests that optimum profile size actually is a cumulative function of Token Recall and Token Precision values over the set of most descriptive tokens.

5. Reduced Profile Generation Time -- Sample Analysis and Profile Synthesis are the two most inefficient processes in the prototype. Speed improvements of at least an order of magnitude should result from using C code in place of UNIX shell scripts for Profile Synthesis, and from streamlining the ways in which Sample Analysis uses the database of corpus token frequencies.

The above set of improvements should result in smaller, more accurate profiles, and in reduced throughput time for Document Selection and Document Scoring. More accurate profiles, in turn, should provide noticeable improvements in both precision and recall.

7. Summary

The prototype described in this paper is a hybrid routing system, successfully coupling the LMDS Boolean search engine with a probabilistic scoring algorithm, and using the speed of LMDS to quickly reduce the set of documents that must be evaluated by the more computationally-intensive scoring algorithm. As implemented, the prototype is simple, compact and fast, and requires no special hardware.

TREC-3 scores for the prototype are highly encouraging, and will be used as a baseline against which to measure future improvements. A set of such improvements has been identified, and will be implemented and evaluated for TREC-4.

TREC-3 Retrieval Evaluation Using Expert Network

Yiming Yang
Christopher G. Chute
Geoffrey E. Atkin
Andrew Anda

Section of Medical Information Resources
Mayo Clinic/Foundation
Harwick 615, 200 First Street, S.W.
Rochester, Minnesota 55905 USA

Abstract

In Mayo Clinic's first year of participation at the Text Retrieval Conference (TREC-3, Category B), our system takes a completely automatic approach to both routing and ad-hoc retrieval, using a combination of a statistical learner (Expert Network or ExpNet) and a shared-word-based matcher (STR). Our focus is to examine how much a Nearest Neighbor approach to query expansion can improve retrieval performance, given the kind of relevance information available in TREC. We found ExpNet effective in the routing test because large amounts of relevant documents are available for each query. In contrast, we found ExpNet less effective in the ad-hoc test because only a small number of training queries are available, and they are often not representative of the testing queries. Therefore, relevance information about such training queries is not very useful for statistical learning about query expansion in general. A realistic strategy for the TREC collections then is to use shared-word-based matching as a basic approach to relevance judgment, and use statistical learning about human judgments for additional evidence. Our experiments show that combining ExpNet and STR leads to better results than using either alone.

1. Method and System Description

Our retrieval system consists of two components, STR and ExpNet. Given a query and a collection of documents, each component system contributes a partial relevance score of a document. The weighted sum of these partial relevance scores are used to rank documents with respect to the query.

STR (string matcher) is our implementation of a vector space model for shared-word-based matching between queries and documents. STR has the same basic features as the SMART system (developed by the Salton group at Cornell) [1], and it allows the use of different word weights such as binary, term frequency (TF), Inverse Document Frequency (IDF), and combinations of them (TFIDF, etc.); however, it does not provide mechanisms for stemming, phrase identification, and relevance feedback. Ideally, we would like to use the SMART system as the string matcher, but we did not have a working interface between SMART and the other components of our system by the deadline for the TREC-3 submission, so we used STR instead.

ExpNet is a statistical learning method for document retrieval and document indexing based on human relevance judgments [2] [3] [4]. In document retrieval, it applies a Nearest Neighbor (NN) approach to query formulation using a set of training queries and their related documents. Given an arbitrary query, ExpNet compares this query to training queries, finds its NNs according to a cosine-similarity measure, and then uses terms (words and/or categories) of the NN-related documents to formulate the "translation" of the original query. ExpNet is similar to relevance feedback in the sense of using training documents to expand a query. The fundamental difference is that ExpNet can handle queries which are not included in the training set, while relevance feedback can only handle queries which are included in the training set. ExpNet is applicable, in theory, to both routing and ad-hoc tasks, however, its effectiveness is dependent on the availability of training queries which are representative of testing queries. The training data in the TREC collections for routing tests and adhoc tests are obviously unbalanced. In the routing case, each testing query itself is a training query, and has a few hundred related documents assigned by humans, providing rich information for the expansion of this query. In the ad-hoc case, on the other hand, none of the training queries are

the same as a testing query, and the total number of training queries is rather small (150 at most). This means that a testing query may not have any "near" neighbors in training queries, and that using related documents of remote training queries to expand this testing query can be misleading. Due to the nature of training data, we expect the relative performance of ExpNet to be much better in routing tests than in ad-hoc tests. We use a weighting factor to adjust the effect of ExpNet when combining it with STR in the retrieval process. That is, we make ExpNet more influential in relevance judgment when the training data are relatively reliable; otherwise, we make ExpNet less influential. The weighting factor can be experimentally determined, as described in the following sections.

ExpSTR is the combined system using both ExpNet and STR. There are two ways to combine the two component systems. One way, as shown in Figure 1, is to combine them at the stage of query formulation, i.e.

$$Q_{expstr} = Q_{str} \cup \alpha \times Q_{expnet} = \{t_1, ..., t_m\} \cup \alpha \times \{t_{m+1}, ..., t_{m+n}\},$$

where Q_{str} is the original query, Q_{expnet} is the expanded portion by ExpNet, and Q_{expstr} is the combined query. The weighting factor, α, adjusts the influence of Q_{expnet} in Q_{expstr}. We use conventional term weights (Section 2 and 3) in query formulation; in the above formula, t_i for $i = 1, \ldots, m$ are the weights of the terms in the original query, and t_i for $i = m+1, \ldots, m+n$ are the weights of the terms in the NN-related documents. In document retrieval, the cosine-similarity between Q_{expstr} and each document is used as the relevance score of the document.

An alternative way to combine ExpNet and STR is to run each component system separately to obtain partial relevance scores of documents, and then merge the relevance scores of these component systems, i.e.

$$\text{relevance-score}(D) = \text{cosine-similarity}(Q_{str}, D) + \beta \times \text{cosine-similarity}(Q_{expnet}, D)$$

where D is a document, and β is the factor adjusting the influence of ExpNet in ExpSTR and is experimentally chosen. Figure 2 illustrates this approach.

There is no fundamental difference between the two ways of merging. The former makes the combined query Q_{expstr} explicit, which is required by the TREC committee for management purposes. The latter makes the experiments for finding the optimal merging factor more efficient, i.e. we only need to run each component system once to search, and repeat the merge with different values of β.

Our retrieval system is implemented as a combination of C++, Perl and UNIX shell programming, running on a SUN SPARCstation 10 with 32 Mbytes of memory and 3 Gbytes of local disk.

2. Document Indexing

Our experiments use the category B documents including the Wall Street Journal (WSJ) documents in disks 1 and 2, and the San Jose Mercury News (SJMN) documents in disk 3. Each document is represented using a vector whose dimensions are words or identifiers of categories. For convenience, we call a word or a category identifier a term. Each vector of a document consists of subvectors, each of which corresponds to a field of the document. For WSJ documents, we use the words in the fields of TEXT, HL and LP, and the categories in the fields of CO, GV and IN. We use the phrase of a category name as an atomic token. For SJMN documents, we use the words in the fields of TEXT, HEADLINE, LEADPARA and DESCRIPT. All terms have a TFIDF weight which is the product of the within-document term frequency and the Inverse Document Frequency (IDF) of the term. The document collection of category B is used to compute IDF values of terms. All categories have an equal weight of one. The system allows using different weighting factors for subvectors; however, due to the time limit we had for the deadline of TREC-3, we only tested an equal weight for all subvectors. We use the stop list of SMART to remove non-informative words. No stemming or phrasing is used.

Document indexing is the major computational bottleneck in our experiments. As this is our first year experience with TREC, our current programs are relatively inefficient to handle large amounts of documents. It takes about one hour on our system to index 10 Mbytes of documents (counting the size of the raw data on TREC disks), which is significantly slower than the SMART system which reported an indexing performance

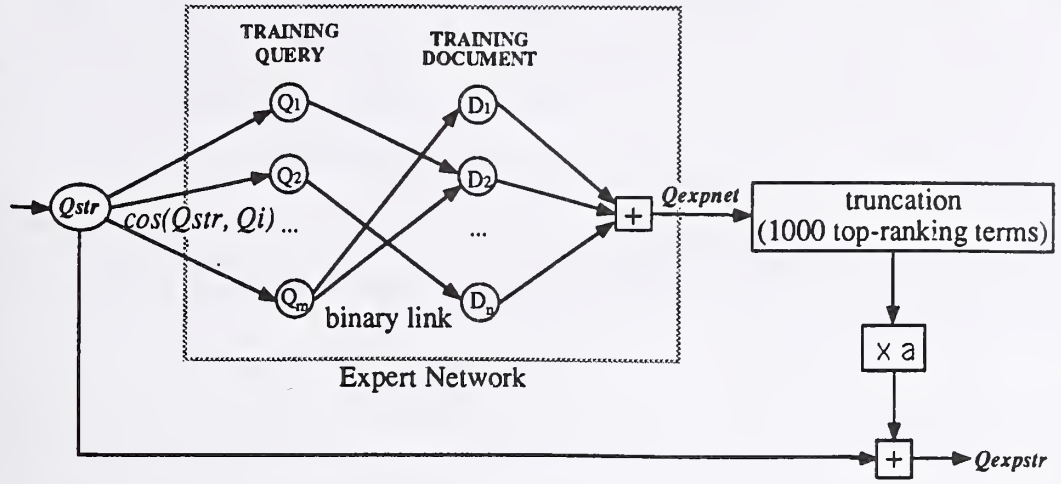


Figure 1. Query formulation using Expert Network where *Qstr* is the original query, *Qexpnet* is the transformed query by Expert Network, and *Qexpstr* is the expanded query by merging *Qstr* and *Qexpnet*.

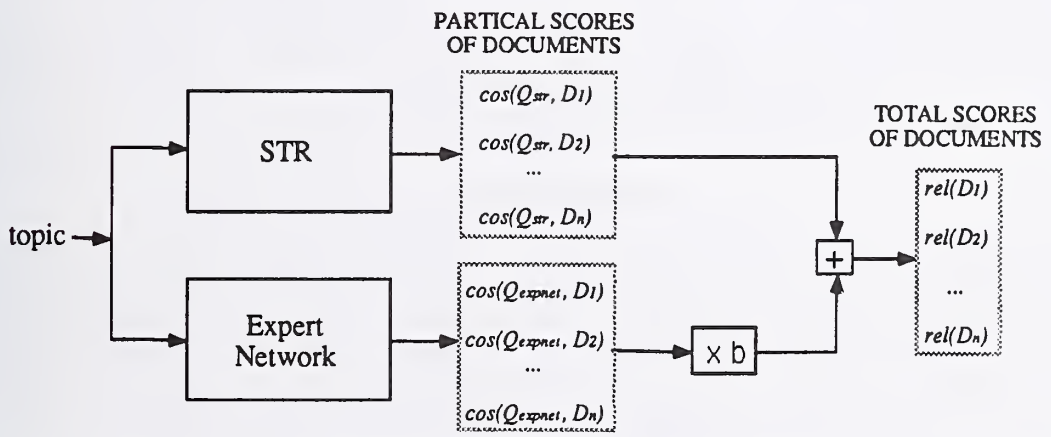


Figure 2. Merging document relevance scores obtained by STR and ExpNet; the truncated version of *Qexpnet* is used, i.e. the maximum length of *Qexpnet* is 1000 terms.

of about 240 Mbytes/hour. Due to the inefficient indexing performance, we have not been able to try a variety of options in term weighting and vector normalization, so the current setting of document vectorization may not be optimal.

3. Query Processing

A query is represented using a vector whose dimensions are topic words. We use the fields of *title*, *desc*, *narr* and *con* in a topic to form the subvectors of a query. Different weighting factors are used for subvectors. We give a weight of one to the field *narr*, and a weight of 10 to the other three fields. These weights were determined empirically, i.e. we tried three combinations of weights, and chose the best combination among these three; this choice is not necessarily optimal. All query words have a TFIDF weight which is a product

of within-query term frequency and the IDF value of the term. The IDF values are the same as those used in document indexing.

Both routing queries and ad-hoc queries are expanded. We use ExpNet to identify K Nearest Neighbors (NNs) among training queries using the cosine-similarity scores of training queries with respect to a given query. The documents related to these K training queries are collected and weighted using corresponding cosine-similarity scores, and the terms in these documents are added together to form a new query Q_{expnet}:

$$Q_{expnet} = \cup_{Q_i \in \{K \text{ top-ranking NNs}\}} \text{cosine-similarity}(Q_{str}, Q_i) \times \{D_{i1} \cup D_{i2} \cup \ldots\}$$

where Q_i for $i = 1, \ldots, K$ is the ith-ranking nearest neighbor of the given query, K is a predetermined parameter, and $D_{i1}, D_{i2} \ldots$ are the documents related to Q_i. From the terms in Q_{expnet}, N top-ranking terms are further selected, weighted, and added to the original query as below:

$$Q_{expstr} = Q_{str} \cup \alpha \times \{N \text{ top-ranking terms in } Q_{expnet}\}$$

where α is the weighting factor which we use to adjust the influence of ExpNet in ExpSTR, as mentioned earlier.

The three parameters, K, N and α are empirically chosen. The choices are dependent on the nature of tasks and statistical features of training data. For routing we set K to 1 because each testing query is a training query. That is, a query is the NN of itself, and its related documents are the most ideal references for query expansion; referring to more NNs would only increase the noise in query expansion and decrease the retrieval performance [2]. In the ad-hoc case, on the other hand, all the training queries are different from the testing queries, i.e. none of the training queries are an ideal NN of a testing query, so it is better to refer to multiple NNs instead of a single NN [2]. Experimentally, we found that $K = 13$ is about optimal for the ad-hoc test. We also found that $N = 1000, \alpha = 0.45$ for routing, and $\alpha = 0.10$ for ad-hoc are about optimal settings. The α value in routing is much higher than its value in ad-hoc because the training data in the former case are much more reliable than those in the latter case.

4. Tests and Results

The routing test: For training, we used topics 101-150 and their related WSJ documents in disks 1 and 2; for testing we used the same topics as queries, and SJMN documents in disk 3 as the search space. Due to a communication problem between our group and NIST, we used some wrong files in the SJMN collection, which caused an SGML parsing error, and as a result, only 40% of the SJMN documents were indexed. Our official submission of the routing results (*expst1*) was effected by this error, whose average precision was 0.1838. After correcting this indexing error, the average precision is 0.2446 (the SMART evaluation packages were used for the statistics).

We also used SMART to replace STR for a comparison. We used the *ltc* query weighting and the *lnc* document weighting schemes in SMART; we did not use phrase processing or relevance feedback. No claim is made that these are the best possible settings of SMART. Our interest is to use SMART as a better string matcher, and to observe whether the effect of ExpNet on SMART is similar to its effect on STR. The system combining ExpNet and SMART is referred to as ExpSMART.

Figure 3 shows the recall-precision curves of ExpSTR with the processing error (the official routing result), ExpSTR after the error was fixed, and ExpSMART. The 11-point average precision of ExpSMART is 0.3509, a 43% improvement over ExpSTR (the correct result) whose 11-point average precision is 0.2446. This comparison indicates that SMART is significantly better than STR, i.e. term weights used in STR and ExpNet are far less than optimal, and that an improvement in term weighting would significantly improve the results of STR and ExpNet, and consequently improve the results of ExpSTR and ExpSMART.

Figure 4 compares the curves of STR, ExpNet and ExpSTR. The 11-point average precisions are 0.1667 of ExpNet, 0.2007 of STR, and 0.2446 of ExpSTR which has a 22% improvement over STR. We also observed a 5% improvement of ExpSMART over SMART whose 11-point average precision is 0.3332 (not shown in this figure). It is evident that ExpNet can improve the results of both STR and SMART.

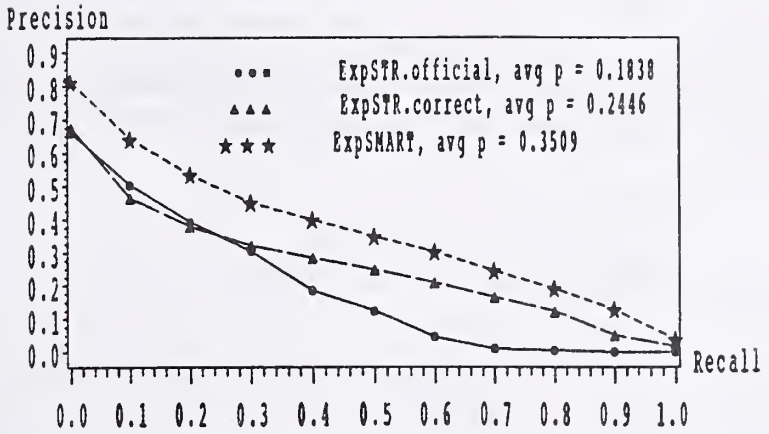


Figure 3. Routing Results

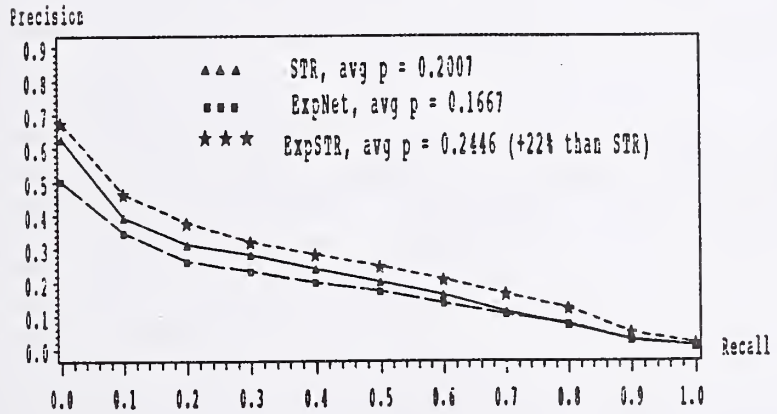


Figure 4. STR, ExpNet and ExpSTR in Routing Tests

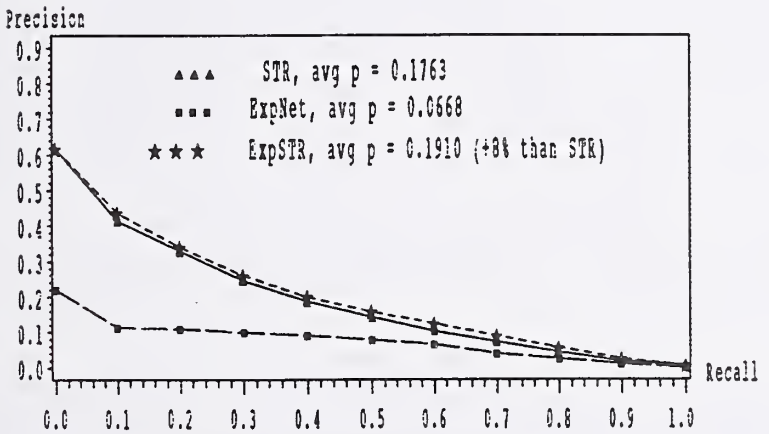


Figure 5. AdHoc Results

The ad-hoc test: For training, we used topics 1-150 and their related WSJ documents in disks 1 and 2; for testing we used topics 151-200 as queries, and the same WSJ documents as the search space. We tested both ExpSTR (as the official submission to TREC-3, labeled as *expst2*) and ExpSMART. Figure 5 shows the curves of STR, ExpNet and ExpSTR. The 11-point average precisions are are 0.0668 of ExpNet, 0.1763 of STR, and 0.1910 of ExpSTR (8% improvement over STR). The improvement of ExpSMART over SMART was 2% (0.3307 verses 0.3231, not shown in this figure).

ExpNet had a much poorer performance in the ad-hoc test than its result in the routing test, reflecting the sensitivity of ExpNet to the quality of training data. Nevertheless, using ExpNet in combination with a string matcher, STR or SMART, still improved the results of the string matcher. ExpSMART is again significantly better than ExpSTR (0.3307 verses 0.1910, 73% improvement), indicating the potential improvements by using better term weights in STR and ExpNet, and consequently the improvements in ExpSTR and ExpSMART.

5. Discussion

In the TREC experiments, we have focused on whether a statistical learning method is useful in situations where training data may or may not be sufficient. Our Nearest Neighbor approach to query expansion has shown improvements in both routing and ad-hoc tests, compared to the results when using shared-word-based matching alone. The improvements by using ExpNet were much more significant in the routing tests than those in the ad-hoc tests, reflecting that ExpNet is sensitive to the quality of training data. Interestingly, despite the poor performance of ExpNet in the ad-hoc tests, it still improved the retrieval results when it is used in combination with STR or SMART. This means that the statistical evidence about relevance captured by ExpNet is consistent and complementary with the evidence captured by word-based matching. It also indicates that better retrieval performance of ExpNet can be expected when more training data are available in TREC.

Further questions can be raised about our Nearest Neighbor approach. That is, can we use this approach without requiring human relevance judgments? Would we get better results if we use automatically estimated relevance judgments instead of using human relevance judgments, given that the training data are insufficient? The answer to both questions is yes. Many TREC participating systems applied a modified version of relevance feedback to the ad-hoc task. In those systems, instead of requiring human relevance judgments among K top-ranking documents in an initial retrieval for an ad-hoc query, all the K top-ranking documents are treated as relevant documents, and used for expanding this query. This is equivalent to our Nearest Neighbor approach in ExpNet, if we remove the nodes of training queries, and directly search for NNs among training documents instead of training queries. The reported improvement of SMART in the tests of the Cornell's group (TREC-3 Nootbook) is about 20% when using the modified relevance feedback, compared to the baseline search (no relevance feedback) of SMART. This improvement is larger than the improvement (2%) of ExpSMART over the baseline SMART in our tests, indicating that the NNs found in a document space are much more representative of an ad-hoc query, compared to the NNs found in training queries. This is not surprising, considering the density in the document space is much higher than the density of the query space. We do not think the above observation is generalizable when a better collection of training queries is available; however, it seems a better strategy to use training documents instead of training queries (and their related documents) in the NN approach, given the current data of TREC collections.

In our first year of experience with TREC, we have faced major challenges in system efficiency issues. The performance level of our current system is not satisfactory, and this has limited our experiments to find optimal parameters of our method. Further study includes optimization of term weights and vector normalization in document and query representations, and improvements in data structures and algorithms for system efficiency.

References

[1]. Salton G. Development in Automatic Text Retrieval. *Science* 1991; 253:974-980

[2] Yang Y. Expert Network: Effective and Efficient Learning from Human Decisions in Text Categorization and Retrieval, 17th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 94) 1994, 11-21.

[3] Yang Y, Chute CG. An application of Expert Network to Clinical Classification and MEDLINE indexing. Proceedings of 18th Ann Symp Comp Applic Med Care (SCAMC 94) JAMIA 1994; 18 (Symp Suppl):157-61. [Best Theoretical Paper Award].

[4] Chute C, Yang Y, Buntrock J. An evaluation of computer-assisted clinical classification algorithms. Proceedings of 18th Ann Symp Comp Applic Med Care (SCAMC 94) JAMIA 1994; 18 (Symp Suppl):162-6.

Acquaintance: A Novel Vector-Space N-Gram Technique for Document Categorization

Stephen Huffman and Marc Damashek
Department of Defense
Ft. George G. Meade, MD 20755-6000

Acquaintance is the name of a novel vector-space n-gram technique for categorizing documents. The technique is completely language independent, highly garble resistant, and computationally simple. An unoptimized version of the algorithm was used to process the TREC database in a very short time.

Acquaintance is the name of a new technique for information processing that combines the robustness of an n-gram based algorithm with a novel vector-space model. Acquaintance gauges similarity among documents on the basis of common features, permitting document categorization based on a common language, a common topic, or common subtopics. The algorithm is completely language and topic independent, and is resistant to garbling even at the 10% to 15% (character) level. Acquaintance is fully described in (Damashek, 1995). The TREC-3 conference provided the first public demonstration and evaluation of this new technique

The Acquaintance algorithm can be used for processing the information in a database of documents in at least two distinct ways. One method explores the conceptual space of a set of documents by determining the degree of similarity among all the documents in that set. When the documents are then viewed with a tool that displays the strengths of the connections among documents, and arranges them so that the distance between them corresponds with their putative degree of similarity, the conceptual space defined by those documents will be apparent. That is, those documents which are similar, and thus related by language or topic, will cluster together. Furthermore, documents that relate to two or more different topics will be clearly visible, both in terms of their positions and strength of their connection to more than one cluster of documents. Those documents which are not clearly similar (that is, related by topic) to any others in the set will stand alone and unconnected to other documents. This mode of using Acquaintance is very useful when exploring the contents of a large and unknown database.

Acquaintance can also be used in the more traditional task of retrieving documents from a database based on their similarity to one or more example, or reference, documents. When used in this manner, reference documents are compared to the documents in the database. Those documents in the database which are similar to the reference documents can be quickly and easily identified. Clearly, using Acquaintance in this fashion most closely approximates the tasks performed by systems participating in TREC, and so two variations on this latter method were used to process the data for TREC-3.

Methodology

N-Gram Processing

The Acquaintance algorithm starts by processing texts in a manner very similar to traditional n-gram based techniques. An n-wide window is stepped through text, moving one character at a time. From each n-gram lying within the window, a hash function generates a value that is treated as an address in a vector characterizing that document, and the contents of that vector address are incremented by one. When all of the n-grams in the document have been processed, the document vector is normalized by dividing the frequency count of n-grams at each vector address by the total number of n-grams in the document.

Centroid Subtraction

A distinctive and crucial aspect of Acquaintance is the subtraction of a centroid vector from document vectors when gauging similarity among documents. The centroid in Acquaintance defines a context within which a set of documents can be usefully compared. The method of subtracting a centroid stands in contrast to more traditional vector-space models which frequently use some form of multiplicative weighting, leading to a rescaling of the axes in the vector space.

Another advantage of using a centroid vector is that it characterizes those features of a set of documents that are more or less common to all the documents, and are therefore of little use in distinguishing among the documents. The Acquaintance centroid thus automatically captures, and mitigates the effect of, those features traditionally contained in stop lists.

The creation of the centroid vector for a set of documents is straightforward and language independent. After each separate document vector is created, the normalized frequency for each n-gram in that document is added to the corresponding address in a centroid vector. When all documents have been processed, the centroid vector is normalized by dividing the contents of each vector address by the number of documents that the centroid characterizes. A centroid thus represents the "center of mass" of all the document vectors in the set.

Computing Similarity Scores

Once documents are characterized by normalized document vectors, the resulting vector-space model permits the use of geometric techniques to gauge similarity among the documents. When comparing a set of document vectors to a set of reference vectors, the cosine of the angle between each document vector and each reference vector, as viewed from the centroid, is computed using the equation:

$$S_{mn} = \frac{\sum_{j=1}^{J}(x_{mj}-\mu_j)(y_{nj}-\mu_j)}{\left[\sum_{j=1}^{J}(x_{mj}-\mu_j)^2\sum_{j=1}^{J}(y_{nj}-\mu_j)^2\right]^{1/2}} = \cos\theta_{mn}, \quad m,=1,\ldots,M, \quad n=1,\ldots N \quad (1)$$

where the vectors x_m, $m \in 1,\ldots,M$ are the M document vectors, the vectors y_n, $n \in 1,\ldots,N$ are the N reference vectors in a J-dimensional space, and μ is the centroid vector.

A cosine value of 1.0 indicates that the document and reference vectors are perfectly correlated (or identical), a value of 1.0 that they are perfectly anticorrelated (or antithetical), and a measure of 0.0, that they are uncorrelated (or orthogonal). We have done a great deal of experimentation using this scoring method for gauging topic similarity, and we have a clear idea of how the measure behaves as features such as n-gram length and garbling are varied (Huffman, 1995).

Acquaintance at TREC-3

System Parameters and Text Processing Procedures

We made the decision to participate in this year's TREC conference at a very late date. Consequently, we did not have the opportunity to modify the algorithm or tune it to the data; we used a generic, unoptimized version of Acquaintance written in ANSI C. The TREC data was processed on a heavily time-shared Cray YMP. Both the routing and ad hoc tasks were run as overnight background jobs, and each took less than 8 hours clock time to finish. We used an n-gram length of 5, and our vector length was 262144.

Acquaintance requires almost no preprocessing of the documents. To prepare the TREC database, we merely stripped away the SGML tags and headers from the data, and then processed only the characters between the TEXT tags. Acquaintance ignored all non-alphabetic characters in the text and translated all lowercase alphabetic characters to uppercase characters.

Query generation was, of course, completely automatic. The "queries" consisted of reference vectors generated from example documents or topic descriptions.

Routing

It is clear that Acquaintance, at least in the form discussed here, is best used for example-based document retrieval. Therefore, in TREC-3 we were particularly interested in how the technique performed on the routing task, which permits systems to operate in an exemplar-based fashion. To perform the routing task, we took the documents from TREC-2 which were defined to be relevant to each of the routing topics. We concatenated all the relevant documents for a topic into a single large document, and from that created a reference vector for that topic. From all the reference vectors, we created a centroid vector, in the manner described above. Thus, we had robust reference vectors and a centroid vector based on a large set of typical, relevant documents.

To score the documents in the routing database, we created a document vector from each document and computed the cosine of the angle between that document vector and each of the topic reference vectors, according to Eq. (1). If a document scored above a certain threshold when compared to a reference vector, we stored that document's number and score. To ensure that we would report at least 1000 documents for each reference vector, we kept the highest scoring document from every file for each topic, whether that document exceeded the threshold or not. After all documents were compared to all reference vectors, we sorted the documents by score within topic, creating a ranked list of documents gauged similar to each topic.

According to a preliminary analysis of the TREC-3 results, among the 34 systems participating in this task, Acquaintance scored at least as well as half in 18 out of the 50 topics, and it scored better than two-thirds of the systems in 10 of the topics. A summary of Acquaintance's performance on the routing task is shown in figure 1.

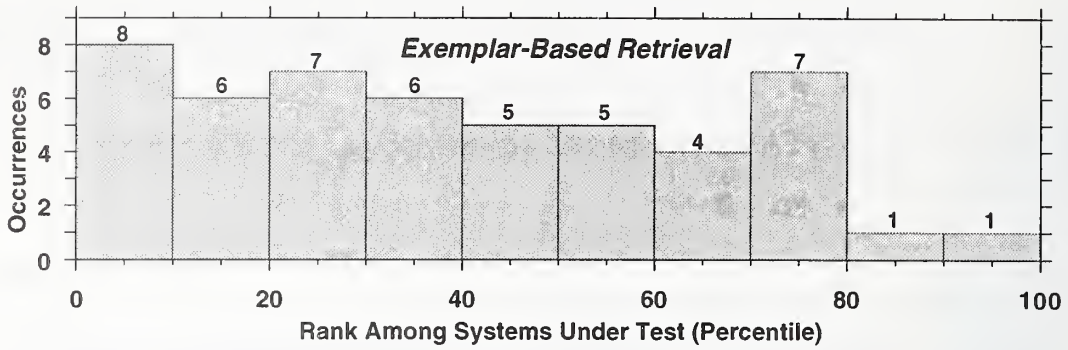


Figure 1

Ad Hoc

For the ad hoc task we had no example documents for the new topics. Furthermore, the topic descriptions were not particularly good exemplars of the documents in the database, since they were short, generally contained a fairly restricted vocabulary, and contained boilerplate language irrelevant to the topic (i.e., " .. to be relevant, a document ...") as well as language expressly unrelated to the topic ("..not relevant would be...").

In a effort to minimize the problem of irrelevant, boilerplate language in the topic descriptions, we scored the document vectors against the reference vectors in a different manner than in the routing task. As before, we created a reference vector from each of the topic descriptions, and a reference centroid vector from all of the topic vectors. This reference centroid served to minimize the effect of the stereotyped phrases peculiar to the ad hoc topic descriptions. Then, however, we read in one file of documents from the database at a time, and created not just document vectors, but also a centroid vector for that set of documents to capture the commonality among the database documents. Finally, when comparing a document vector to a reference vector, the appropriate centroid was subtracted from the corresponding vectors, as shown in Eq. (2):

$$S_{mn} = \frac{\sum_{j=1}^{J}(x_{mj}-\mu_j)(y_{nj}-\nu_j)}{\left[\sum_{j=1}^{J}(x_{mj}-\mu_j)^2\sum_{j=1}^{J}(y_{nj}-\nu_j)^2\right]^{1/2}} = \cos\theta_{mn}, \quad m=1,\ldots,M, \;\; n=1,\ldots N \qquad (2)$$

where the vectors x_m, $m \in 1,\ldots,M$ are the M document vectors, the vectors y_n, $n \in 1,\ldots,N$ are the N reference vectors in a J-dimensional space, μ is the centroid vector for the current set of documents, and ν is the centroid for the set of reference documents.

While this double-centroid measure seemed to eliminate the problems caused by stereotyped language in the topic descriptions, the lack of good example documents for

the topics caused Acquaintance's performance in the ad-hoc portion of TREC to be considerably below that of its performance in the routing task. In fact, Acquaintance exceeded the median score of the other systems on only two topics, though it performed better than 10% of the other systems for more than half of the topics. A summary of the performance of Acquaintance in the ad hoc task is shown in figure 2.

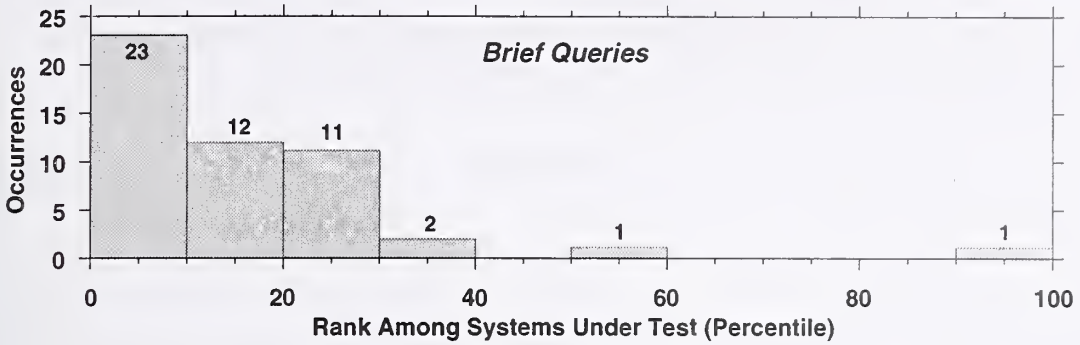


Figure 2

Summary

Acquaintance's performance in the TREC routing task was on par with most of the systems presented at the conference. Given the fact that the algorithm combines such performance with complete language independence, strong tolerance for garbles, and ease of implementation, we believe that it warrants serious study and further development.

Future Directions

Acquaintance has often been used in conjunction with a knowledge visualization tool called Parentage (Cohen, 1995). This powerful tool permits a user to view the relationships among documents in a variety of ways. It allows, for instance, the exploration of a database's conceptual space. For the next TREC conference, we hope to utilize Parentage to improve Acquaintance's performance, particularly on the ad hoc task. We intend to begin by processing the data with essentially the same basic Acquaintance algorithm, and determine the 1000 or so documents which score most highly when compared to a particular topic. These high scoring documents and the relationships among them can be examined using the Parentage tool, and those documents that form the best cluster of documents which are highly related to the topic can rapidly identified. Those documents will be used to create a new reference vector for that topic, now based on highly relevant example documents. We believe that this method will recover a much better sample of relevant documents from the database.

As part of our poster session, we demonstrated that Acquaintance was able to reliably categorize documents which were garbled at the 15% character level. We believe that the ability of information retrieval systems to recover relevant documents regardless of a moderate degree of garbling is very important when considering the real-world applications of these systems. Therefore, we intend to take part in the new track at TREC-4 which will investigate systems' performance on such garbled data.

References

[Cohen 1995] Jon Cohen: "Drawing Graphs to Convey Proximity: an Incremental Arrangement Method," submitted to ACM Transactions on Computer-Human Interaction.

[Damashek 1995] Marc Damashek: "Gauging Similarity via N-Grams: Language-Independent Categorization of Text," Science **246** (1995, in press).

[Huffman 1995] Stephen Huffman, in preparation.

Information Retrieval System for TREC3

Kenji SATOH, Akitoshi OKUMURA, Kiyoshi YAMABANA
Information Technology Research Labs.
NEC Corp.

1. Introduction

This is our first participation with TREC. Our team researches natural language processing, and we have developed English-Japanese and Japanese-English machine translation system. (The code name of machine translation system is VENUS.) We are now researching a new natural language processing environment, including information retrieval and text understanding. (The environment name is VIRTUE: VENUS for Information Retrieval and Text Understanding.)

Last year, our team participated with MUC5, and we got promising results[1]. This year,

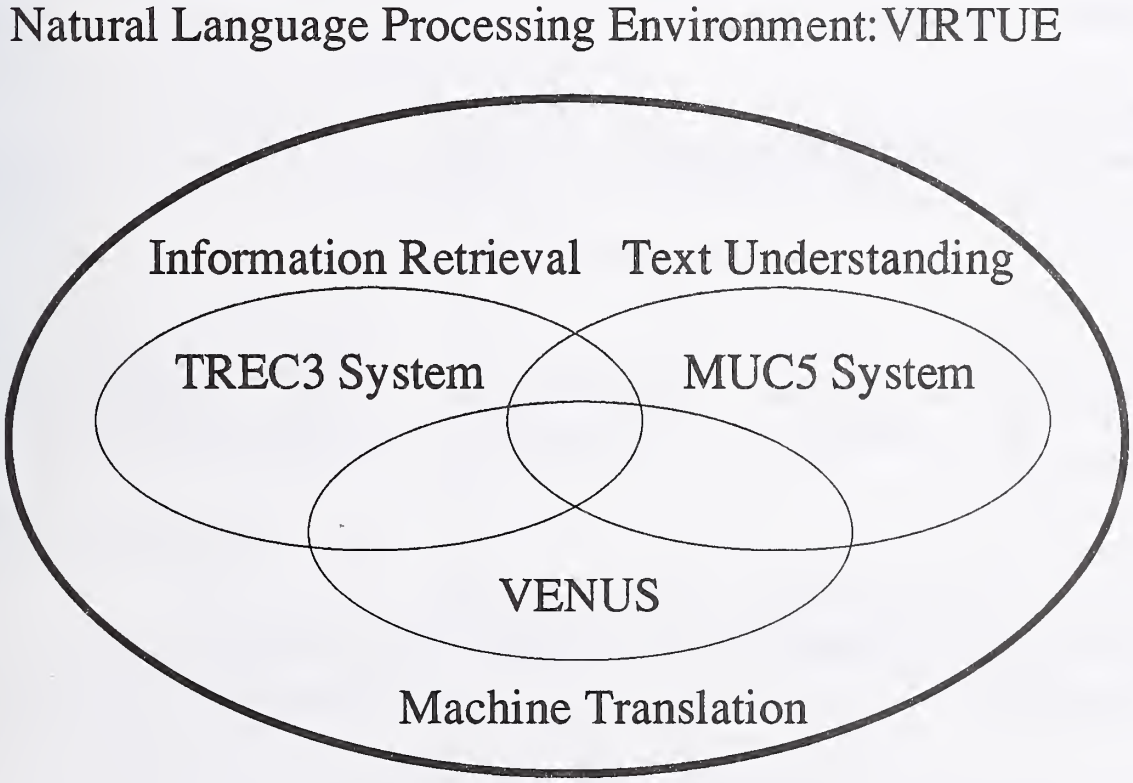


Figure 1. Our Goal

our team has participated with the TREC3 for researching and developing and information retrieval system. Figure 1 shows our goal of researching natural language environment.

Because of our past development of machine translation systems, we have on hand translation system dictionaries (140,000-word), a conjugation table of English words, and morphological and syntax parsers. The purpose of participating with TREC3 focuses on the feasibility of using this data and applying English language analysis technology. We developed a system very quickly using them.

2. System Modules

The overall outline of the system concerns firstly, the generation of an index from NIST document collection as inverted files. We generated inverted files for each CD-ROMs. Query term generation from topic with routing task phase is generated manually, and with ad-hoc task phase is processed automatically.

Query weights are added by weight calculation modules with referring inverted files. Finally, matching between query and document as well as ranking documents used the same program with both tasks.

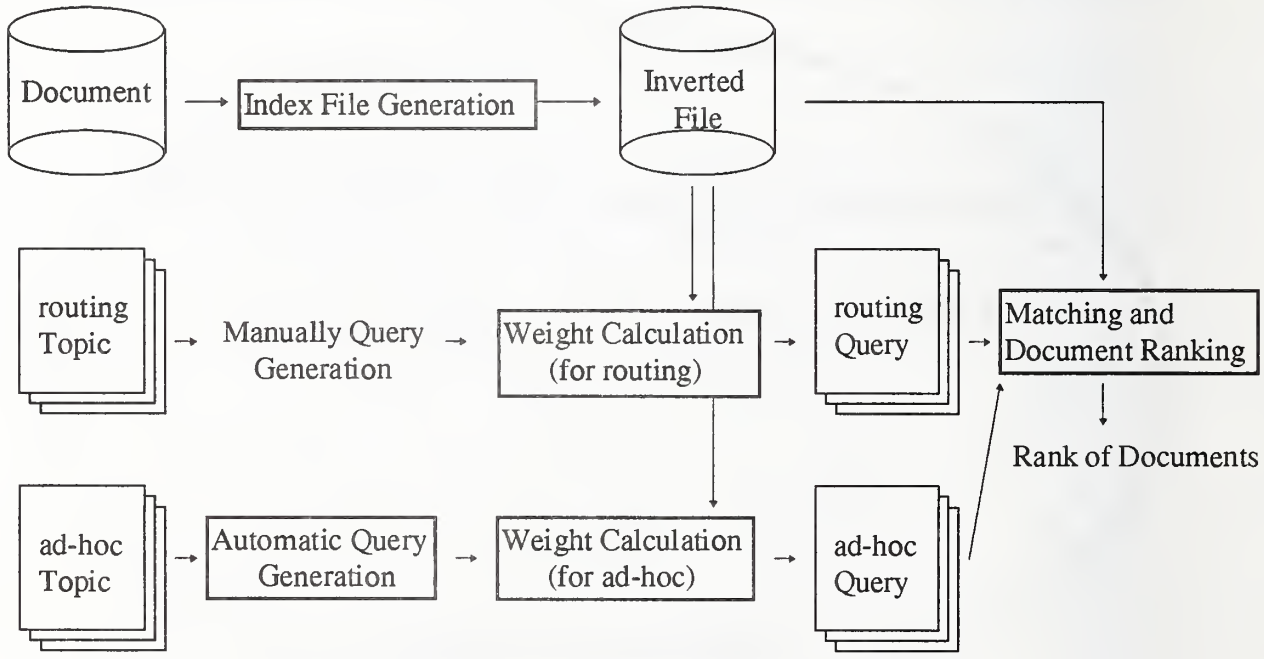


Figure 2. System modules

I will explain this system in order; Index File Generation, Query Generation, and Matching and Document Ranking. Figure 2. shows modules of our system.

3. Index File Generation

The index generation method from a document collection will be explained in this section. The index is created by an inverted file in which a single word (not including spaces) as a key. Words in the conjugated form and/or the plural form may be used as a key.

The reason that general stemming method dose not used is this method can not handle irregular conjugations, our system has an English-language conjugation table, and it is thought that with this table, the conjugated form sufficiently handled during matching phase.

The word recognition of each document is carried out by matching with a 140,000-word English dictionary, the 430 stop words are dropped. Furthermore, symbolic strings (repetition of upper case and hyphens, etc.) are recognized as a single word.

After the word recognition phase is carried out, the DOCNO of documents in which the word has appeared, as well as the frequency and position of the word in each document are saved. Although word weight is not recorded at this point in time, word weight calculates for the words appearing in the query when matching with document.

During indexing generation, listing the number of words in the document is simultaneously generated as a separate file. This information is used in normalizing the word weight during matching.

The size and generating time (with Sun SPARCstation 10) of the inverted files are:

Disk 1 --- 880 MBytes, 93 Hours
Disk 2 --- 610 MBytes, 53 Hours
Disk 3 --- 660 MBytes, 67 Hours

4. Query Generation

A sample of query is shown in Figure 3. The query format is a compound format consisting of the boolean operator and word weight. The one key of the query does not necessarily have to be a singular word; if it is, the word sequence in the text is checked.

As boolean operator, AND condition as simple listing of keys in query, OR condition (enclosed by brackets) in which the appearance of multiple keys regarded as a single key. A key

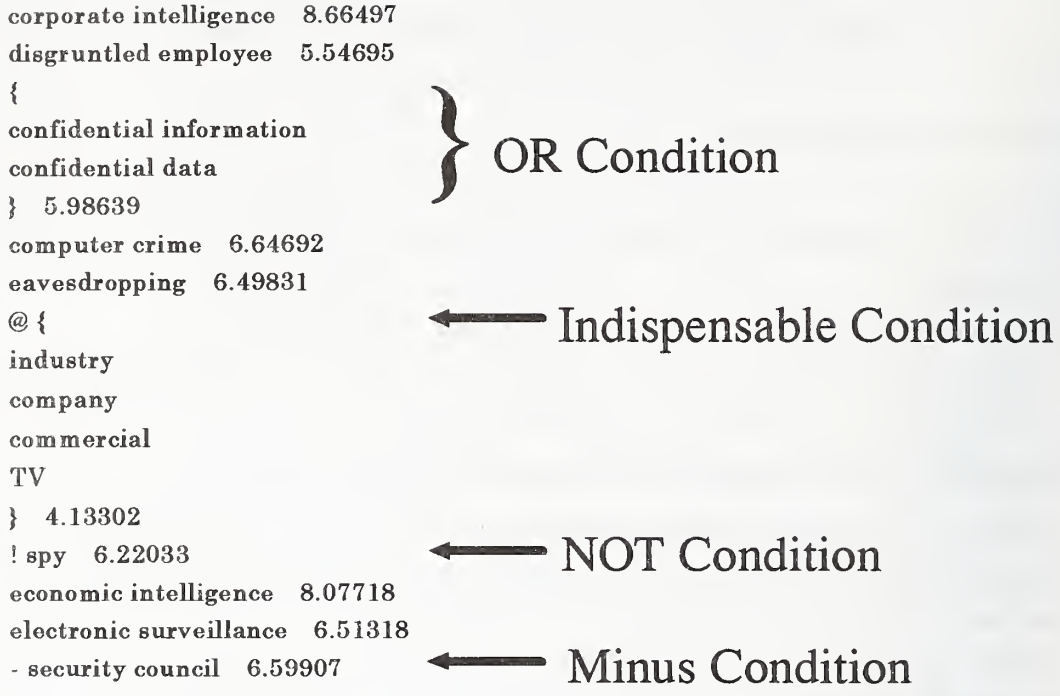


Figure 3. Sample of query

preceded by an exclamation mark indicates a NOT condition, and a key preceded by an @-mark indicates the indispensable condition. A key preceded by a minus-sign makes the word weight value take on a negative.

The query generation with routing task phase is executed manually. At the first stage, initial query matched for training documents. That result is compared with the relevance judgment, upon which the query is modified.

The query generation with ad-hoc task phase processed automatically. For all the files in topic, an English-language analysis is executed from which noun phrases are extracted. In this phase, if there are expressions indicating negation for a noun phrase, the phrase is inserted in a query with a minus operator.

Word weight calculation does not occur in either of the query generation processes. Word weight is assigned by a program after completion of the query keys. A combination method using the operator and word weight will be explained as follows.

5. Matching and Document Ranking

The way in which the query operator and word weight combinations are carried out during matching is as follows. Keys connected by an OR condition calculate the frequency and list up the documents in which a word appears as a single key. A document with the NOT condition key in a matched document on another key is excluded from the ranking calculation. Conversely, a document without the indispensable key conditions in a matched document on another key is excluded from the ranking calculation.

Next, when an inverted file is referenced by a key in the query, the key returns once to its original form by using the conjugation table and it is unfolded into the all the possible conjugated form. If a key 'wrote' exists in the query, it returns to 'write' and it is unfolded into 'write', 'wrote', 'written' and 'writing.'

Also, if the key in the query is consist of multiple words, the unfolding phase and index search are enabled for each word in the key. If a key 'wrote file' exists in the query, it is unfolded into 'write file', 'wrote file', 'written file', 'writing file', 'write files', 'wrote files', 'written files' and 'writing files.' After each word is searched, the sequence check is executed according to their positions in the document which saved in the index file.

The weight calculation during the routing task phase is uses the modified form of traditional relevance weight model[2]:

$$w_i = \log \frac{(r_i + 0.5)/(R - r_i + 0.5)}{(n_i - r_i + 0.5)/(N - n_i - R + r_i + 0.5)}$$

where

N : the number of documents
n_i : the number of documents containing the term i
R : the number of relevant documents
r_i : the number of relevant documents containing the term i

Also, the weight calculation during the ad-hoc task employs the modified form of the standard deviation of word appearance frequency:

$$w_i = \log\left(\frac{N}{n_i}\sqrt{\frac{1}{L}\sum_{j=1}^{L}\left(\frac{m_{ij}}{M_j} - \frac{n_i}{N}\right)^2}\right)$$

where

L : the number of documents
N : the total number of terms in all documents
n_i : the number of term i in all documents
M_j : the total number of terms in document j
m_{ij} : the number of term i in document j

For ranking, the internal product is taken between the weight of words in a query and word frequency in a document, and the log factor is imported by the results of multiplying each internal product [3].

$$S_i = \frac{1}{\log D_i}\sum_{j=1}^{N}\log\left(\left(d_{ij}+1\right)* w_i\right)$$

where

S_i : the score of document i
D_i : the total number of terms in document i
N : the total number of terms
d_{ij} : the number of term j in document i
w_j : the weight of term j

Total time of weight calculation, matching with document and document ranking is about 20 Minutes (average per query.)

6. Discussion

The results of our system are shown in Table 1.

Table 1. Results of our systems

	Routing (manually)	Ad-hoc (automatically)
Average precision	0.2717	0.0624
R-precision	0.3204	0.1170

Because the calculation of the word weight uses the standard deviation of word appearance frequency during ad-hoc task phase, the deviation value is often very large for particularly long documents. This results in such documents being placed in a high ranking. Another problem is that noun phrases which automatically extracted from topic, sometimes retrieves insufficient documents. Table 2 shows the summary of our system.

Concerning points of improvement from now, we are thinking of another weight calculation method rather than the use of deviation. Also, if an adjacent operator were introduced, we believe that even if a topic changes within a particularly long document, more precise matching can be carried out. For the retrieval of more sufficient documents, other sources, such as thesaurus or Word Net, should be used so that the number of keywords is increased.

Table 2. Summary of our system

	routing	ad-hoc
Index File Generation	Conjugated form used as a key. Position of word is saved.	(same as routing)
Query Format	Boolean and weight compound format	(same as routing)
Query Generation		Noun phrase extraction
Term Weighting	(traditional relevance weight model)	Standard deviation
Matching	Keys are unfolded into all conjugated form. Multiple words key is checked the sequence.	(same as routing)
Ranking	(internal product)	(internal product)

7. Conclusion

Because this is the first such presentation of TREC3, we started from estimation of the amount of time and memory necessary for the creation of an inverted file, and unfortunately, did not have enough time to repeatedly matching nor modify of the weight.

But using our original dictionary and conjugation table, as well as adopting natural language processing technology for developing this system, we got promising results when applying these technologies to information retrieval systems, while at the same time, we decreased developing and maintenance costs. (We have developed this system in a month with a 3-person team.)

References

[1] Muraki, K. Doi, S. and Ando, S. "NEC: Description of the VENIEX System as Used for MUC-5", proceedings of Fifth Message Understanding Conference (MUC-5), pp.147-159, Aug. 1993

[2] Robertson, S.E. and Sparck Jones, K. "Relevance weighting of search terms", Journal of the American Society for Information Science, 27, pp. 129-146, 1976

[3] William B. Frakes and Ricardo Baeza-Yates edit, "Information Retrieval - Data Structure & Algorithms"

Decision Level Data Fusion for Routing of Documents in the TREC3 Context: A Best Case Analysis of Worst Case Results.

Paul B. Kantor
Department of Library and Information Studies and
Alexandria Project Laboratory (APLab), SCILS, Rutgers
kantor@kantor.rutgers.edu

ABSTRACT

The performance of a simulated test of decision level data fusion in the routing (filtering) task of the Text Retrieval Conference is summarized and analyzed. The relatively poor results of an approach in which a specific fusion rule was selected for each retrieval task are analyzed in terms of a best possible fusion scenario based on a given scheme for quantizing the messages from the systems to be combined. The limitations of that scenario are in turn explored, and possible ways to improve upon it are outlined.

I. INTRODUCTION

In the TREC3 setting, each participant submits two proposed ranked sets of retrieved documents for each of two sets of 50 problem or "topic" statements. The top portions of each retrieved set are pooled and evaluated for relevance (binary score) to the corresponding topic by trained evaluators. [See Harman (1995) for further details]. One set of topics, called the "Adhoc topics" are provided without any training information, and are to be run against a specified set of retrievable documents. The second set of topics, called the "Routing topics" a provided together with a set of relevance judgements for (selected items) from a training set of documents. This permits participants to "tune" the query formulations, retrieval systems, or fusion rules, before applying them to a "test" set of data. One expects that the performance of such tuned rules would be, in general, better than the performance of adhoc retrieval, and that the performance on the test set should be comparable to (but probably somewhat lower than) the performance on the training set.

The purpose of the present work is to simulate the situation in which two or more distinct systems are available, each providing evaluations of the "similarity score" (or "retrieval status value") for the same set of documents, to a given topic. In addition, we seek to simulate the situation in which the internal workings of the several systems and, in particular, the "scores" are hidden from view (either because they are proprietary, or because the working of the system e.g. a neural network, does not produce a numerical score). In this case each system provides only its ranked list of documents, for each topic. This is an instance of what is called "decision level" fusion. [Hull, Esp Ch 6]. That is, the ranked lists represent the decisions, made by the several systems, in response to the presented topic. In the work described here, a single very effective retrieval system, the University of Massachusetts Inquery system [Turtle and Croft] was used to perform all indexing, stemming and retrieval.

In an effort to simulate the case of several systems, three different retrieval modes of the Inquery system were used, and their results were treated as if they had come from three different systems. Two of these modes accept a Boolean formulation, and the third, called "natural language processing (nlp)" accepts strings of terms, with possible repetitions. Queries were formed by a "nearly algorithmic" process, carried out by one person, working without aid of thesaurus. The details of the fusion rules are set out below.

After the training set had been run, the apparently most effective fusion rule was selected for each routing topic.

Table 1. Overall Results of Data Fusion Trials. The measure used throughout is precision at 100 documents retrieved, averaged over 50 topics

Method	Precision at 100 Documents
rutfua1	.371
rutfua2	.359
rutfur1	.266
rutfur2	.267

File rutfur1 contains, for each topic, the results from that fusion scheme which did best on the training set for that topic. Best is determined by comparing precision at 100 documents. File rutfur2 contains, for each topic, the results from that fusion scheme which had second best performance on the training set for that topic. Best is determined as above.

The procedures for rutfur[1,2].test are exactly as described for the training set. The weights used are the ones determined from the training set. The searches were run at the University of Massachusetts, using Inquery, with (inverse) document frequencies determined on the training set only.

Since there were no training data for adhoc runs, the method which appeared to be most effective most often was applied to the adhoc topics. The adhoc runs, rutfua[1,2].test were run using the best data fusion schemes, based on overall performance on the training set. Specifically rutfua1.test uses a sum of ranks provided by the three component schemes, to determine an effective rank. rutfua2 uses the minimum of the ranks assigned by the three component schemes to determine an effective rank.

For the adhoc runs, rutfua1 and rutfua2 our scores fall near the median whichever measure of performance is used. As shown by Tague-Sutcliffe [Workshop presentation at TREC3], posthoc Scheffe tests can be applied to these data, treating the precision scores as the sum of an effect due to the topic and an effect due to the system. Under this analysis, using the precision at 100 documents as a score, the adhoc fusion results are not significantly worse [at 95% confidence] than the results obtained by using the full power of the Inquery system, which

provided top performance among all the systems in TREC3. This lack of difference is presumably a reflection of the low power of the TREC setting to discriminate among systems, rather than any suggestion of parity between the two sets of results. A discussion of a somewhat less restrictive nonparametric approach to the comparison of systems, when their rank in the TREC setting is known, is given in an unpublished note [Kantor 1994].

For the routing case, as noted, a "tuned" choice of fusion rule was made for each topic. As a second set, the second best fusion rule was assigned for each topic. These results ranked dead last by several measures of system performance! The purpose of the present note is to lay out in more detail what was done, and to explore some details of the decision-level fusion process, in an effort to understand why the results were not better.

II. QUERY FORMULATION AND THE THREE SIMULATED SYSTEMS

The three simulated systems, which are to be combined in a variety of ways, begin with the reduction of the topic text to a Boolean expression as a single conjunction of disjuncts. This corresponds to the basic notion of "combination of concepts" as it is used in commercial Boolean set retrieval systems. The Boolean forms were constructed by a graduate student who added no vocabulary, and only made one correction to an error in the topic text. Proximity operators were used, but no weights were provided, as they would not be interpretable by all modes of the Inquery system. An example query is, for Topic #125:

```
Inference Form: #q125 =
#and(#or(government authority court)
    #or(law regulation control limit warning discourage funding research action)
    #or(#1(anti smoking) smoking tobacco)
    #not(#or(#2(price support) #2(export encouragement)))
);
```

This was submitted to the Inquery "inference mode" (inf), using the default settings for belief levels. [Turtle and Croft] This produces a ranked set output. It was also submitted to the "hard Boolean mode" (hbl) which produces a set retrieval without ranking.

The operators were removed to provide the query formulation for the "natural language processing mode" (nlp), using the default values for belief settings. This produces a ranked set. The Unstructured or "natural language processing" form for Topic 125 is:

```
#q125 =
government authority court law regulation control limit warning discourage funding research
action anti smoking smoking tobacco ;
```

III. DECISION LEVEL DATA FUSION FOR INFORMATION RETRIEVAL.

Given the output of two or more systems or modes of a single system we combine them using several possible fusion rules. In order to illustrate the action of these rules we introduce a binning procedure which will be used for the remainder of this analysis. Such a binning procedure is also called a "quantization" of the decision-level signals issued by the systems. The set of ranks assigned by any mode is broken into 25 bins, each containing 40 consecutive ranks. These are labelled by the numbers 0,1, ... 24, which we call "bin ranks". A given document, if it is retrieved in the top 1000 by two different modes, will have two different bin ranks, (b_1,b_2). Hence it can be represented as appearing in the location indexed by that pair of coordinates.

We will also refer to those products of bins as bins, in the plane. In general there will be some number of documents, in each bin, which have been judged and judged relevant. We denote this by $g(b_1,b_2)$. Similarly, there will be some number that have been judged not relevant, which we denote by $b(b_1,b_2)$. The goal of any retrieval system faced with this information is to sweep across the plane of bins, in such a way that the accumulation of relevant documents is as rapid as possible, while the accumulation of not relevant documents is as slow as possible. This will be made more precise below.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
2	2	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
3	3	3	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
4	4	4	4	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
5	5	5	5	5	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
6	6	6	6	6	6	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
7	7	7	7	7	7	7	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
8	8	8	8	8	8	8	8	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
9	9	9	9	9	9	9	9	9	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
10	10	10	10	10	10	10	10	10	10	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
11	11	11	11	11	11	11	11	11	11	11	11	12	13	14	15	16	17	18	19	20	21	22	23	24
12	12	12	12	12	12	12	12	12	12	12	12	12	13	14	15	16	17	18	19	20	21	22	23	24
13	13	13	13	13	13	13	13	13	13	13	13	13	13	14	15	16	17	18	19	20	21	22	23	24
14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	15	16	17	18	19	20	21	22	23	24
15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	16	17	18	19	20	21	22	23	24
16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	17	18	19	20	21	22	23	24
17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	18	19	20	21	22	23	24
18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	19	20	21	22	23	24
19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	20	21	22	23	24
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	21	22	23	24
21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	22	23	24
22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	23	24
23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	24
24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24

Exhibit 1. Maximum logic for fusion of ranks. The combined rank of each bin is shown.

There are a number of a priori schemes which may be applied to pursue this goal. Three of the simplest are the symmetric rules: MAX, MIN and SUM. These may be defined formally by the rule which gives the combined bin rank b_{com} as a function of the values (b_1,b_2). These

rules are shown in Table 2.

Table 2. Mathematical form of the rules of combination.

Rule	Formula for combined rank
MIN	$\min(b_1,b_2)$
MAX	$\max(b_1,b_2)$
SUM	b_1+b_2

```
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
0  1  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
0  1  2  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
0  1  2  3  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
0  1  2  3  4  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
0  1  2  3  4  5  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
0  1  2  3  4  5  6  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
0  1  2  3  4  5  6  7  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
0  1  2  3  4  5  6  7  8  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
0  1  2  3  4  5  6  7  8  9 10 10 10 10 10 10 10 10 10 10 10 10 10 10
0  1  2  3  4  5  6  7  8  9 10 11 11 11 11 11 11 11 11 11 11 11 11 11
0  1  2  3  4  5  6  7  8  9 10 11 12 12 12 12 12 12 12 12 12 12 12 12
0  1  2  3  4  5  6  7  8  9 10 11 12 13 13 13 13 13 13 13 13 13 13 13
0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 14 14 14 14 14 14 14 14 14
0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 15 15 15 15 15 15 15 15
0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 16 16 16 16 16 16 16
0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 17 17 17 17 17 17
0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 18 18 18 18 18
0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 19 19 19 19
0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 20 20 20
0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 21 21
0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 22
0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
```

Exhibit **2**. Minimum logic for fusion of ranks. The combined rank of each bin is shown.

Each document is located in one of the bins, identified by the rank assigned to that bin by the inf system (reading down the page) and by the rank assigned to that bin by the nlp system (reading across the page). The number at the position of each bin indicates its rank under the fusion rule.

In Exhibits 1,2, and 3 we show the combined rank assigned to each bin in the plane, under each of the three rules of combination. In practice ties must be broken in some way, such as using one of the two bin coordinates as a second sort key. This is not material for the remainder of this analysis, and will not be pursued here.

It is clear from examination of Exhibit 1, that the MAXimum rule corresponds to the

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48

Exhibit 3. Sum of ranks rule for combination of ranks. The combined rank of each bin is shown.

logical combination AND. That is, a bin will have combined rank less that, say, 5, if and only if the ranks assigned to it by both modes of retrieval are less than or equal to 5. Similarly, the MINimum rule corresponds to a logical OR. That is, a bin in the plane will have a rank less than or equal to 5 if either of the two modes assigns it a rank less than or equal to 5.

Finally, the SUM rule, while still treating the two coordinates symmetrically, does not correspond to a specific logic. Rather, it permits a tradeoff of high rank in one mode and low rank in another. There are many other possible symmetric rules of combination, which can be generated from basic symmetric functions of two or more variables as discussed by Kantor [1981]. The enumeration of symmetric logical rules for combining three inputs is given by Cherikh.

In our treatment of the routing problem we asked which of several rules provided the best performance, for a given topic, and then applied the same rule (using of course the same query formulations) to retrieval from the test set. In fact, since our experiments included fusion of decision level data (that is, ranked lists) from each of three modes of the Inquery system, we considered a fourth possible rule, MED which set the combined rank equal to the median of the three separate ranks. Our analysis, in this note, of the problems with tuning decision level fusion will be carried out using only two modes (the nlp and the inf).

IV. EXAMINING THE TEST AND TRAINING DATA TOGETHER

To get a better understanding of the problems that have arisen, we present the bin arrays for the test and training data on a single topic from the routing set, Topic 125. The data of interest are the numbers $g(b_1,b_2)$ and $b(b_1,b_2)$. However, the retrieval performance, by any standard measure, will be optimized if we sort the bins in decreasing value of the ratio:

$$r(b_1,b_2)=g(b_1,b_2)/b(b_1,b_2)$$

Since this may be undefined (if $b(b_1,b_2)=0$), we work instead with:

$$p(b_1,b_2)=g(b_1,b_2)/[g(b_1,b_2)+b(b_1,b_2)].$$

This is undefined only if there are no evaluated documents in the bin. In that case we represent the bin by a ".". To compress the ratio into a single digit we use a formula defining the graphic character, $c(b_1,b_2)$:

Table 3. Definition of Graphic Characters for Exhibit 4.	
$c(b_1,b_2)$	"." if g+b=0
	"!" if g>0 and b=0
	$Int(Log_2(1000*g/(g+b)))$

The largest value of the numerical character is $Int(Log_2(1000))=9$.

Table 4. Training data, for three modes of retrieval, for Topic 125.	
Inquery retrieval mode	Precision at 100 Documents
Hard Boolean	.02
Inferential	.46
Natural Language	.45

Examining the patterns in Exhibit 4, we see that the patterns look generally similar. This means, from the perspective of data fusion, that this should be a fairly good test case on which to explore the potential of data fusion. To do so we consider several possible ways in which to order the bins, in the test array.

One possibility is to pay no attention to the training set, and adopt an order such as one of the ones shown in Exhibits 1,2,3. A second possibility is to "learn" as much as possible from the training set, and to use the results of that learning to perform the data fusion on the test set. We first summarize the information that was available after the training run. Recall that, under

the data fusion rules, the queries themselves are not to be modified on the basis of the retrieved set in the training run. All that can be changed is the rule of combination.

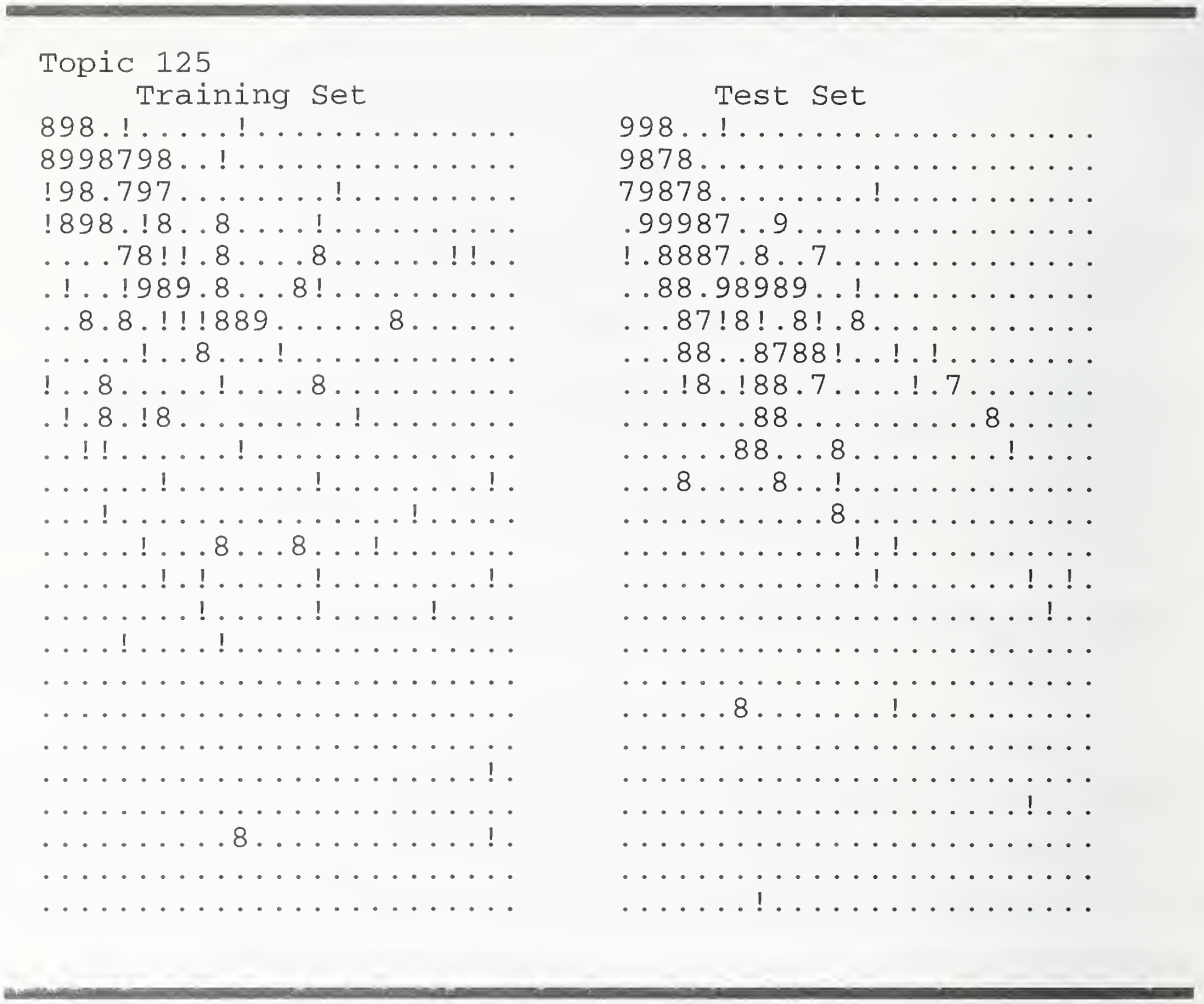


Exhibit 4. Relative distributions of relevant and non-relevant documents in the training and test sets for Topic 125. Codes are defined in Table 3.

The precision, at 100 documents, for the original three runs of Topic 125 are shown in Table 4. These data show that our concentration on just two of the modes is certainly justified here, as the third (the hard Boolean form) seems to have very little to contribute. We note that this disparity might suggest that the problem with our performance is due to the insistence on symmetry among the three modes, which seems unreasonable in situations like the one shown in Table 4. However, the detailed results to follow show that the problems are deeper than that.

V. TUNING THE DATA FUSION RULE.

In the TREC context we explored the possibility of tuning, Topic by Topic, by considering each of several data fusion rules, and selecting, for each topic, the rule which performed best. We did not consider the question of whether differences among rules were statistically significant in making these choices. Four rules were considered: the maximum, the minimum, the equally weighted sum, and a weighted sum. Under the weighted sum rule the rank assigned to a document by each mode is divided by 0.02+p(100, Mode). The 0.02 offset is intended to prevent division by zero. p(100,Mode) is the precision of that Mode, at 100 documents, in retrieval from the test set. In the present case the quantized version of this formula becomes:

$$b_c=b_{hbl}/.02+b_{nlp}/.45+b_{inf}/.46.$$

Effectively, this gives the inf and nlp modes 22 or 23 times as much "weight", in the sense that after the first bin of the hbl retrieval is included, some 22 bins of each of the other two will be brought into the fusion result before the second bin from the bhl mode enters. The precise meaning of this is a somewhat unclear as the boolean system does not produce a ranked retrieval. Thus, at some point there are no more documents to be considered from that mode.

For the present, then, we consider just the inf and nlp modes, as shown in Exhibit 4. However, the performance of four possible rules of combination [for all three inputs] is shown in Table 5.

Table 5. Performance of rules of combination on the training set. Topic 125

MAX	MIN	MED	Sum(Wtd)	Sum(Equal)
.28	.30	.31	.05	.45

SUM(Equal) means the sum with equal weights. Based on these results, our first choice for the test situation should be the sum with equal weights. In our TREC runs, however, this alternative was not included. We do not know why the weighted sum performs so poorly for this topic or, for that matter, in general.

Given the disappointing results of the fusion rules selected here, we have explored the possibility of learning in much finer detail. That is, rather than sweeping across the array of bins in one of the three patterns shown in Exhibits 1,2,3, we consider picking and choosing among the specific bins in the plane, so as to produce the best possible results on the training set.

The most aggressive approach to this is to rank the bins in the plane in decreasing order of g/b. This produces the results shown in Figure 1. This figure contains a great deal of information on both the training and test performance. The horizontal axis records the number of non-relevant documents collected while sweeping across the bins in the indicated order.

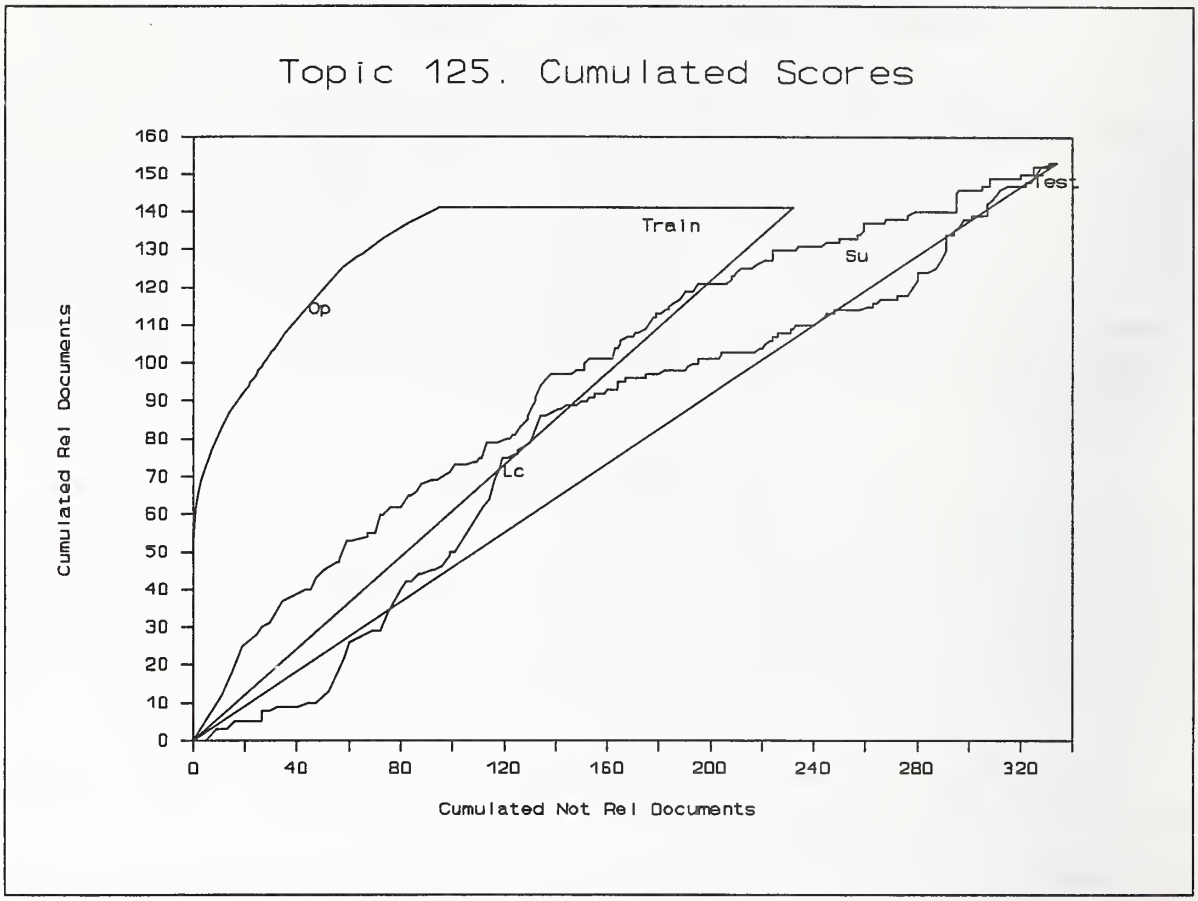


Figure 1. Cumulated Number of Good Documents. Bins sorted by g/(g+b).

The left-most curve , labelled "Op" is the cumulated curve for the training data if bins are collected in the optimal order (corresponding to first taking all bins marked "!", and then taking up the numbered bins in decreasing order, as shown in the left portion of Exhibit 4). The performance in this case, on the training set, is very good. To normalize this cumulated performance curve we have shown the straight line from the origin to the endpoint. This straight line represents the performance that would be achieved, on the average, if bins were simply taken in random order.

In the curves extending to the right-most portion of Figure 1, we show the corresponding structure for the test data. Again the straight line represents expected random performance. The jagged line labelled "Lc" (for "Local" tuning) shows the performance if the order of bin selection is exactly the same as the one used to generate the training curve. In the early portions it tracks the straight line, beginning below it, and rising above it. It lies above the random performance for an interval, and then falls below it again. Thus this highly detailed choice of tuning performs, overall, not very differently from a random selection rule. To calibrate the notion of "not very differently" we show, in the curve labelled "Su" the performance, on the training set, that is achieved by using the sum of ranks rule of Exhibit 3. This clearly everywhere dominates the performance of detailed local tuning. In other words, knowledge of the training set has not

helped the data fusion at all!

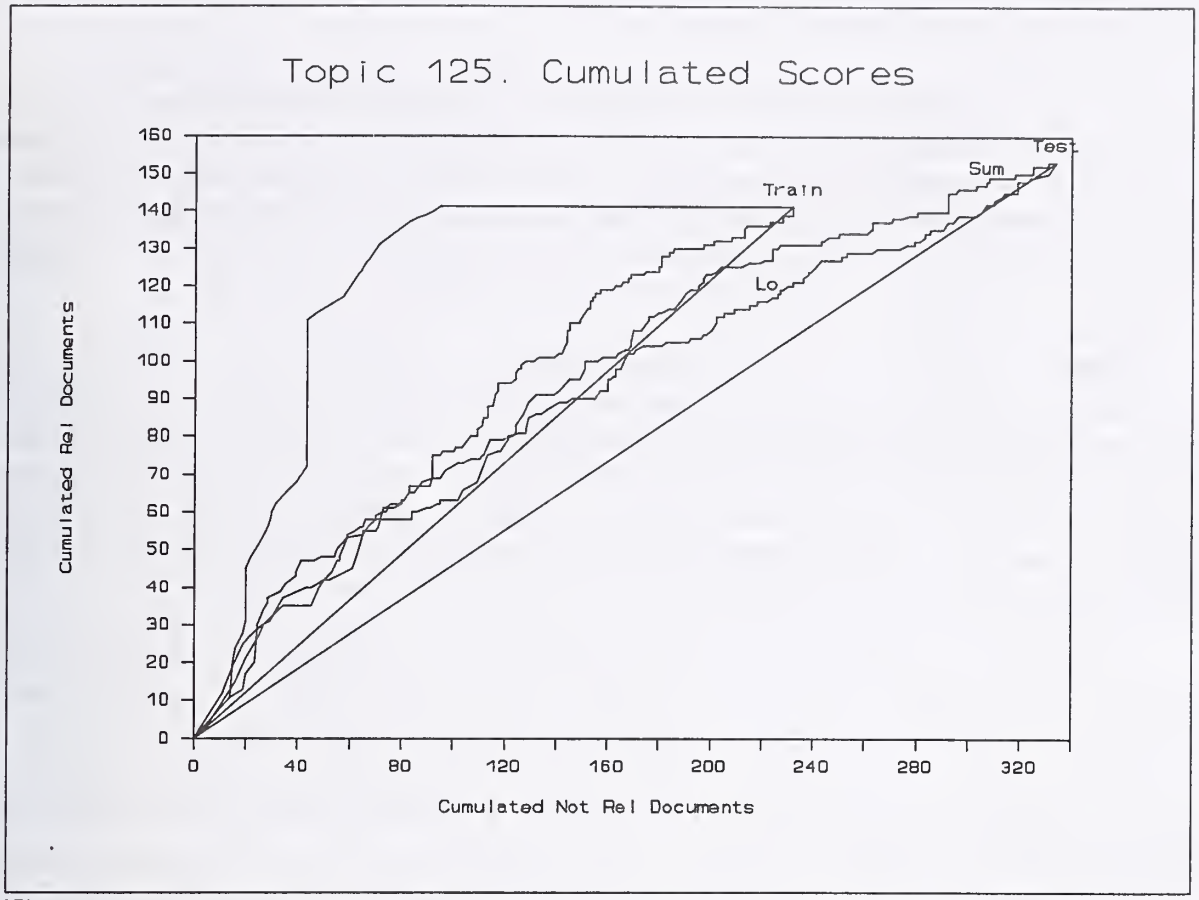


Figure 2. Cumluated relevant documents, using a rule which assigns greater weights to bins with more judged documents.

Table 6. A modified rule for determining combined rank of bins. "sqrt" represents the square root function.

Situation	Combined Rank
b=0, g=0	-1
b=0, g=0	sqrt(g)
b>0, g=0	0
b>0, g>0	g/sqrt(g+b)

The obvious explanation for this is that we have "over trained". That is, we are using distinctions among bins which are due essentially to random fluctuations in the training data. Presumably these fluctuations are most prominent in bins containing small numbers of relevant (or non-relevant) documents. To test this, we have used another ranking scheme, which weights

bins according to the square root of the total number of judged documents that they contain. The modified ranking rule is shown in Table 6.

The resulting cumulated curves of relevant versus non-relevant documents are shown in Figure 2. On the training set this does not perform as well as the optimal rule shown in Figure 1. But it does very well in comparison to the sum rule, shown as a jagged line rising from the origin to the endpoint of the training set ("Train"). On the test set, this less aggressive rule, labelled "Lo" is consistently better than random. But, except for a small region near the origin (which may well be due to fluctuations) it is still not better than the sum rule.

These results move us towards understanding why our application of data fusion to the routing situation did not work well. Disappointingly, they also suggest that even tuning that makes use of all the routing information will not do better than a symmetric fusion rule chosen a priori. However, the details of plots such as Exhibit 4 suggest that there may be other "generic" fusion rules which are more effective that the rules shown in Exhibits 1,2, and 3. In that plot we see that relevant documents are nicely clustered into the upper left corner, but, then, they continue across the plot in patterns that stay close to the main diagonal.

This suggests a rule for combination that does not correspond well to a simple logical expression. For example, it could be represented as $b_{com}=|b_{nlp}-b_{inf}|$. The vertical bars represent absolute value, and ensure that the combined rank is positive. The meaning of such rules is not yet clear. One possible interpretation is that when the two schemes agree on the rank that they assign, it is more likely that the document is relevant than that it is not. But why this should itself be true is not yet clear.

VI. DISCUSSION AND CONCLUSIONS

The clear conclusion of our efforts to select a "best symmetric tuning rule" for each of the routing topics is that, as implemented here, it does very poorly in the TREC setting. The first observation is that (Harman 1995) most TREC groups used results of the training data to improve their query formulations, eliminating terms that led to irrelevant documents, and adding terms that retrieved relevant documents. The evidence shows that this level of tuning is clearly superior to an approach that treats the query formulations as part of an impenetrable black box.

However, motivated by our observation that not all systems will permit their interiors to be manipulated in this way, we ask (perhaps over-optimistically) whether there are directions in which the present work might be extended to bring its performance closer to that of other training schemes.

We can represent the situation here using Q,R,S to represent systems, a prime (') to represent the training of systems, F to represent fusion rules, and F' to represent a trained fusion rule. Using < to mean "performs more poorly than" our present result is that F'(Q,R)<Q'. However, there is a body of evidence suggesting that F(Q,R)>Q, even if Q>R [Belkin, Kantor,

Fox and Shaw].

One line of approach is to seek better formulations of the training rule F-->F'. This will require detailed exploration of schemes for defining and ordering the bins in the space of combined ranks. We have used prior fixed bins. Better results might be obtained, for example, by using a nearest neighbor scheme, or some other pattern analysis method [Fukunaga] to determine the ordering of the bins. The bins will have to be weighted along the lines used in the work shown in Figure 2, to avoid the effects of overtraining.

A second line of approach, to be investigated in future work, considers that perhaps our effort to simulate the case of different systems is not realistic enough. While we use different modes of the Inquery system, those modes still draw upon the same underlying stemming and indexing algorithms. Thus our "systems" Q,R,S are not very different from each other. [This is reflected, in Exhibit 4, by the tendency of the bins with high weight to lie close to the diagonal. This is a pattern which is not consistent with any of the rules shown in Exhibits 1,2,3.]. A very different approach is given by the so-called n-gram schemes [Cavnar, Damashek]. In direct evaluation [TREC-3] these schemes have not done as well as term-based schemes. But they are likely to be more nearly independent from those schemes, and hence to provide a more powerful basis for trainable data fusion schemes in information retrieval.

VII. ACKNOWLEDGEMENTS

Mr. Won-Sik Shim, of the Rutgers program in Library and Information Studies produced the "nearly automatic" but concept-structured query formulations. Mr James Callan processed them in several ways, using the Inquery software at the University of Massachusetts Laboratory for Information Retrieval Studies. We are indebted to Prof. W. Bruce Croft for this support as well. Prof. Nicholas Belkin at Rutgers installed and maintained the Inquery file file structure on the Rutgers computers, and provided the support of Mr. Richard Quatrain, of Electricite de France, who coded some of the post-processing data fusion scripts. The Alexandria Project Laboratory is supported by research funds from the US Department of Education, the Council on Library Resources, and other agencies. Additional support is provided by Dean Richard W. Budd of the School of Communication Information and Library Studies.

VIII. REFERENCES AND LITERATURE CITED

Belkin, NJ, Kantor, PB, Fox, EA, Shaw, JA. (to be published) Combining the Evidence of Multiple Query Representations for Information Retrieval. Information Processing and Management. [To be published].

Cavnar, WB. Using an N-Gram-Based Document Representation with a Vector Processing Model. In Harman 1995 op cit.

Cherikh, M. (1989) Optimal Decision and Detection in the Decentralized Case. Cleveland Ohio. Department of Operations Research, Case-Western Reserve University. PhD. Dissertation.

Damashek, M. Gauging Similarity via N-Grams: Language Independent Sorting, Characterization and Retrieval of Text. Prepritn: Fort George Meade, MD.

Fukunaga, K. Introduction to Statistical Pattern Recognition. Second Ed. Academic Press. 1990. 591pp.

Hall, DL. Mathematical Techniques in Multi-Sensor Data Fusion. Artech House. (1992).

Harmon, DK.[1995] The Third Text REtrieval Conference (TREC-3). NAtional Institutes of Standards and Technology. (This volume).

Kantor PB [1981]. The Logic of Weighted Queries. IEEE Transactions on Systems, Man and Cybernetics. v11(12)816-821. (1981).

Kantor PB [1984e]. A Simulation Method for Assessing the Significance of Non-Paramteric Comparisons in the TREC setting. Unpublished. Available by ftp from kantor.rutgers.edu.

Saracevic, T; Kantor PB. (1988) A Study of Information Seeking and Retrieving. III. Searchers, searches, overlap. Journal of the American Society for Information Science v39(3)197-216.

Turtle, HR; Croft, WB. (1991) Evaluation of an Inference network-based model. ACM Transactions on Information Systems. 9(3)187-222.

TREC-3: Experience With Conceptual Relations in Information Retrieval

David Gardiner, John Riedl, and James Slagle
University of Minnesota
Computer Science Department
(gardiner | riedl | slagle)@cs.umn.edu

Abstract

This report describes an experiment evaluating the performance gains that can be achieved by using high-level conceptual relations in information retrieval. The objective of the experiment is to determine if conceptual relations can improve overall retrieval performance and, if so, under what conditions using relations is likely to be justified. We represent five TREC topics each as two concepts linked by a single relation, where a concept corresponds to a noun phrase and a relation corresponds to a verb phrase or a noun phrase that describes an action, activity, or relationship. A Boolean search (with proximity) is associated with each concept and a parameterized search with each relation. We then compare the performance of the expanded concept-relation-concept representation with the searches for the two concepts linked by each of several proximity operators. Our results show that use of relations can provide significant performance improvements but that the improvements are dependent on the nature of the two concepts and the relation with respect to the text collection being searched.

1. Introduction

The primary objective of the field of information retrieval is to provide technology allowing people to find sufficient information to answer questions. Historically, the problem of information retrieval was finding *any* relevant information. Now, with the wide-spread availability of enormous electronic databases, being overwhelmed with *too much* information is common. The challenge of information retrieval is therefore to provide relevant and *only* relevant information. Unfortunately, providing more relevant information tends to also result in more non-relevant information, just as eliminating non-relevant information tends to eliminate information that is relevant.

A key to improving information retrieval performance is for the retrieval system to have a better representation of the user's information need. Information need is often expressed as a set of concepts. The focus of this research project is to determine whether specifying and searching for specific relations between concepts — one way to more completely specify the information need — can improve retrieval performance. For the purpose of this project, a concept corresponds to a noun phrase and a relation corresponds to a verb phrase or a noun phrase that describes an action, activity, or relationship.

Ideas that form information needs are comprised of both concepts and relations. If there are multiple relations that can occur between two concepts, specification of the relation becomes critical to properly specifying the query. Cimino & Barnett note the importance of relations in their analysis of questions generated by physicians (Cimino & Barnett 1992). They cite an example of the concepts *methotrexate* and *psoriasis* and several questions that might be posed by these two concepts: *"Does methotrexate cause psoriasis?", "Does methotrexate treat psoriasis?",* and *"Is the use of methotrexate contraindicated in*

someone with psoriasis?" They advocate development of a vocabulary of "Relation Concepts" to allow more precise specification of medical questions.

While use of relations can allow better representation of information need than concepts alone, there has been limited research evaluating the impact that using relations can have on retrieval performance. The two research questions addressed in this project are:

1. Can retrieval performance be improved by explicitly specifying and searching for relationships between concepts?

2. Under what conditions is the improvement likely to be justified?

Systems such as RUBRIC (McCune et al., 1985) search for concepts. Relationships between concepts are specified implicitly, using Boolean operators and proximity. The idea of using explicit relations to improve retrieval performance has been explored in the context of document indexing. (Farradane, 1981) describes a notation for indexing relations between concepts. Concepts and relations can be specified in a search. Searches then use the index to identify matches. (Farradane & Thompson 1980) describes preliminary experiments indicating that relational indexing can improve retrieval performance. Relational indexing requires manual recognition and indexing of relations in each document in the collection. While manual indexing of concepts has been widely used, it is labor-intensive. Our approach builds on (Farradane, 1981) in that we also explore using relations to improve retrieval performance. However, instead of indexing each relation, we construct a search expression for the existence of the relation in the collection. The manual effort required is in developing the recognition expression. The only indexing required is automated word-based indexing as used in most current information retrieval systems.

2. Experimental Design

To answer these research questions, our experiment compares the performance of searches using a relation between two concepts to searches using a simple proximity operator between the same two concepts. We use concept-relation-concept queries because doing so reduces the issues associated with representing a topic. In many cases, an information need cannot be represented as a single concept-relation-concept (Liddy & Myaeng 1994). A complete representation of an information need often involves multiple concepts and relations, nesting, and functional dependencies. Evaluating the effect of a single relation in a complex representation would be difficult. Furthermore, it is not clear what a complex representation would be compared against. We chose proximity operators as our baseline for comparison because they are commonly used to specify that some relationship exists between two terms without specifying the relationship explicitly.

Since this experiment is done as a part of the Third Text REtrieval Conference (TREC-3), we use the TREC document collection, topics, and relevance judgments. Most TREC topics require more than two concepts and a relation for a complete representation (Liddy & Myaeng 1994). Of the five topics that we use, only two were adequately represented by concept-relation-concept. This means that, for three of the topics used, there is a significant discrepancy between the topic on which the relevance judgments are based and the representation of the topic that we use for searching. We use failure analysis to analyze the effect of this discrepancy on our results.

Figure 1 shows the experimental process used to collect performance results. An information need specification is manually translated into both a concept-relation-concept representation and a concept-

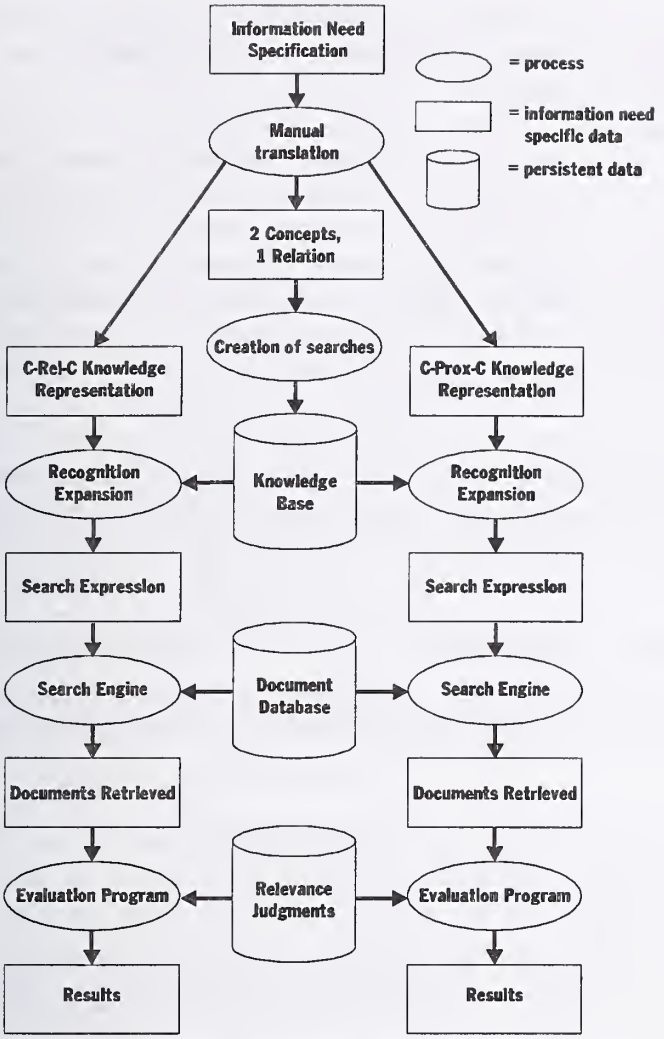


Figure 1. Experimental Process for a Single Information Need. For the experiment, information needs are manually translated into Concept-Relation-Concept representation. The same two concepts are also linked with a proximity operator to produce a Concept-Proximity-Concept representation. The two representations go through the succeeding steps in parallel. First, each is expanded into a search expression using the knowledge base. Next, documents are retrieved by a search engine using the search expression. Then the retrieved documents are evaluated using the relevance judgments to produce performance results.

proximity-concept representation. A search expression is created for each of the concepts and the relation and put into the knowledge base. At this point there are two parallel, identical paths: one for the concept-relation-concept representation and one for the concept-proximity-concept representation. Each representation is expanded using the searches stored in the knowledge base and the expanded search run against a large document collection. The documents retrieved are compared to the pre-existing relevance judgments and an evaluation program outputs performance results. The results for the concept-relation-concept and concept-proximity-concept representations are then compared.

Our planned experimental methodology is shown in Figure 2. We plan to collect results for approximately five information needs. The searches will then be developed and refined using a training collection. When the searches are as good as we can make them, we will collect "after refinement" results. We then plan to load two other collections in turn, collecting "blind" results, refining the searches, and then collecting "after refinement" results on each collection. Finally, we plan to load a final test collection and collect "blind" results on that collection.

Our methodology involves manual effort used to create the concept-relation-concept representations and the concept and relation searches. Since this experiment is intended as a qualitative exploration of the value of relations in information retrieval, the source of the relations and their linked concepts is not significant. Their significance is in what we learn from evaluating their performance. Similarly, the source of the concept and relation search expressions is unimportant. The quality of the concept search expressions has little effect on the results of the experiment as the same concept searches are used in both the concept-relation-concept and concept-proximity-concept searches. For relation searches, our objective is to find whether good relation searches can be defined. If the relation

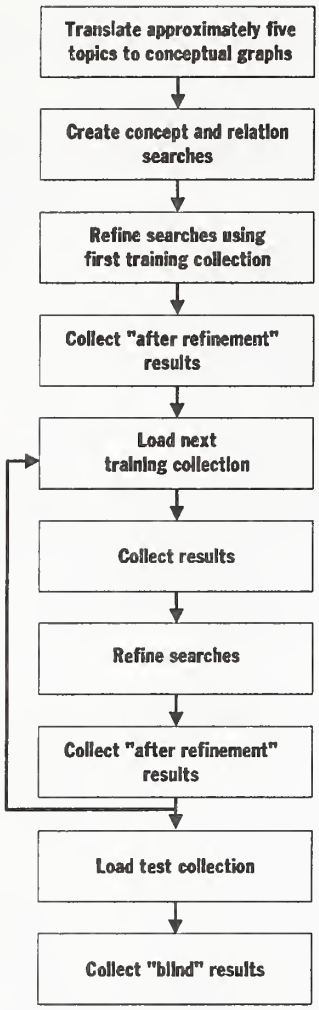


Figure 2. Planned Methodology. TREC topics are translated to concept-relation-concept graphs. Searches for the concepts and relation are created and refined using training collections. A test collection is loaded and final "blind" results collected.

searches are poorly defined, then this experiment might make relations look less effective than they really are.

A danger in using manually created search expressions is that they might be *tuned* to a particular collection so that changes that improve performance in one collection make performance worse in another. We contrast tuning with *refinement*, which improves performance in a new collection without negatively impacting the performance in a previous collection. The use of multiple training collections allows us to determine how much tuning is occurring. The collections used are drawn from two different sources, with all documents in each collection drawn from a single source. This diversity between collections helps ensure that we do not tune for a single source. If tuning occurs, this implies that a relation search may work only with collections very similar to the training collection. If tuning does not occur, it suggests that relations may be defined that can work on a broader set of collections.

The following sections describe the components of the dataflow diagram in detail.

2.1. Information Need Specification and Knowledge Representation

The start of any information retrieval process is some specification of an information need. For this experiment, we use topics and relevance judgments provided as part of the Third Text REtrieval Conference (TREC-3). The narratives of the topics used in this experiment are shown in Figure 3.

We need a notation for the concept-relation-concept representation of an information need. Since our representational need is simple, virtually any knowledge representation has the representational power required. The notation we use is Sowa's conceptual graphs (Sowa 1984). Examples of other representations that would have worked are frames or any Lisp-like notation.

Conceptual graphs are directed graphs containing concepts (denoted with brackets) and relations (denoted with parentheses) connected with arrows. Concept types are organized in a type lattice stored in the knowledge base. While Sowa's notation for concepts allows specification of both a concept type name and a referent, we replace the referent with a *type scope* indicator that is either null, >=, or >. The type scope indicators specify that the concept refers to the specified concept type *only*, to the specified concept type

and all narrower types, or to narrower types *only*, respectively. For example, the concept [politician] represents *only* the specific concept of a "politician" while [politician: >=] represents the concept of a politician or any kind of politician such as a senator, congressperson, or supreme court judge. [politician: >] refers to kinds of politicians, not to the concept of "politician" itself. Use of type scope indicators simplifies representation of information need and appears to simplify knowledge base creation and maintenance.

The relations that we use in this project are high-level, complex relations. Examples are *develops, spies on,* and *finances.* These relations could be defined by conceptual graphs containing concepts and more fundamental relations such as *agent* and *object.* We use complex relations because they tend to be represented in text by words rather than sentence structure and are therefore easier to recognize.

In this project, we use conceptual graphs only as a convenient notation for the concept-relation-concept form of an information need. Unlike previous work in information retrieval that has used conceptual graphs (Dick 1991; Liddy & Myaeng 1994), we do not do direct matching between a conceptual graph and the document collection. Instead, a conceptual graph is converted into a search expression and that search is run on the collection.

2.2. Knowledge Base and Recognition Expansion

Our knowledge base is our repository for concept and relation recognition knowledge. That is, it contains the knowledge used to translate a conceptual graph representation of information need into a form that can be evaluated by the search engine.

Both concept and relation types have *recognition expressions* associated with them. These recognition expressions are written in the Personal Librarian full-text search language with one extension: a concept, which may contain a conceptual graph, may be inserted anywhere in a recognition expression. This capability allows recognition expressions to refer to other concept and relation types rather than repeating the search expression for those types. For example, the concept type *cancer drug* has the recognition expression

[cancer: >=] NEAR/20 [drug]

The recognition expression for a relation type is similar to that of a concept type but must contain two parameters, represented as %1 and %2. These parameters correspond to the recognition expressions of the relation input and output concepts, respectively (we limited our implementation to binary relations). The following is the recognition expression for the relation type *finances:*

%1 NEAR/15 (financ* or fund* or (pay for) or contrib* or donation*) NEAR/15 %2

Recognition expansion is the process of transforming a conceptual graph into a full-text search language expression by replacing each concept and relation with its fully expanded recognition expression.

2.3. Document Collection

As in any information retrieval experiment, we need a large document collection. For this experiment, we need four large document collections: one for testing, two for the knowledge base refinement iterations, and one for the final test. TREC provided three CD-ROMs full of compressed text files for information retrieval research. The data is organized by source. Each of the collections we used was one source from one disk. Our initial training collection was Disk 1 Wall Street Journal (WSJ) data. The other collections,

in the order used, were: Disk 2 WSJ, Disk 3 Associated Press (AP), Disk 1 AP. The Disk 3 AP data was used to produce our official TREC-3 conference results. Due to technical problems, we were able to load only four of twelve months of this data.

2.4. Search Engine

The search engine is the final component in the process of identifying documents that are relevant to the information need. The search engine takes a search expression and returns a set of documents from the document collection. There are two basic kinds of search engines readily available: intelligent search engines that take natural language input and Boolean search engines that require a specific, Boolean-based search language. We chose to use Boolean searching for this experiment because Boolean provides us with greater control over the searching. We did not want the effects of an intelligent search mechanism confounding our results.

The search system used for this experiment is Personal Librarian, created and sold by Personal Library Software. Personal Librarian provides a robust Boolean search mechanism as well as an intelligent search mechanism that we did not use. We stress that we do not test the performance of Personal Librarian's natural language-based search mechanism in this experiment but only its robust Boolean and proximity search capabilities.

The Personal Librarian search language includes the standard Boolean operators AND, OR, and NOT, as well as unidirectional and bidirectional proximity operators. A unidirectional operator requires that the operands be in the specified order and proximity. A bidirectional operator requires only that the proximity criterion be met. In Personal Librarian, the unidirectional and bidirectional operators are W/*n* and NEAR/*n*, respectively, where *n* is the proximity in words. For example, the search "A W/10 B" will return only documents where word A occurs ten or fewer words *before* word B. "A NEAR/10 B" will return all documents where word A occurs within ten words of word B, regardless of the order.

In the experiment, we compare the performance of each relation search expression to ten alternatives: both unidirectional and bidirectional proximity operators for 5, 10, 15, 20, and 30 words. We had hoped to also use *within sentence* and *within paragraph* proximities but these operators were not available in the search engine used. Figure 1 is simplified to show the process for the relation and a single proximity.

Personal Librarian takes the search expression produced by recognition expansion and returns a list of documents. We use a utility program to change this list into a form that can be processed by the TREC evaluation program, discussed below. The results returned by Personal Librarian are ordered by the Personal Librarian search engine's proprietary algorithm.

2.5. Relevance Judgments

Evaluating the performance of a retrieval method requires relevance judgments specifying which documents in the collection are relevant for an information need and therefore should have been retrieved by the system. Relevance judgments were supplied by TREC. Ideally, relevance judgments would exist for all combinations of topics and documents. Since TREC now has 200 topics, the collection contains hundreds of thousands of documents, and relevance judgments must be made by people, only a small fraction of the possible relevance judgments exist. These judgments were created for the top ranked documents returned by systems used in previous TREC conferences.

The limited relevance judgments are issue for our experiment, as for all TREC participants. The TREC evaluation program treats all unjudged documents as not relevant. If a significant portion of the unjudged documents are relevant the results may be skewed.

2.6. Evaluation Program and Metrics

Once we have the list of documents retrieved and the relevance judgments, we can perform evaluation to produce our experimental results. TREC provides an evaluation program that outputs several different measures. The metric that we use in our experiment is non-interpolated average precision. Average precision corresponds to the area under an ideal (non-interpolated) recall/precision curve and is computed as the average of the precision at each relevant document retrieved. We use average precision as our primary metric for three reasons:

1. **It is a single measure of overall system performance.** Any retrieval method will provide some mixture of precision and recall. Since improving precision tends to reduce recall, and vice versa, those cannot be used for comparison. We needed one metric that would provide a measure of *overall* system performance, and average precision is the best metric of that type at this time.
2. **It is a standard metric that is widely used within the information retrieval research community.** Since retrieval performance is subjective, depending on the exact information need of each user, no metric can provide an indisputable measurement of system performance. Average precision is currently the most commonly used metric of overall system performance.
3. **The TREC evaluation program computes it.** This ensures that our computations of average precision would be standard.

Measurement of time for this experiment was "wall clock" time. Wall clock time measures the interval between when a query is submitted and results are displayed, which is the key measure of time relevant to retrieval system users. The standard Windows clock was displayed in digital mode. The start time was recorded when the Personal Librarian "Search" button was pressed, and the finish time was recorded when the search results *began* displaying. The clock display was in one second increments, so the precision of the time results is plus or minus one second.

2.7. System

We developed a system named *Teknos* that is used to build and maintain the knowledge base, maintain a list of conceptual graph queries, and perform recognition expansion on the graphs. The resulting search expression is copied (via the Windows clipboard) to Personal Librarian where the search is run. The results are output to a file and run through a utility program to provide a results file that can be processed by the TREC evaluation program.

Teknos is implemented in Borland C++ using the POET object-oriented database system and runs under Microsoft Windows 3.11. The computer used for this project is a Gateway 2000 486DX2 66 MHz PC with 12 MB of RAM and two hard drives: one has 424 MB of space, the other has 540 MB compressed using Stacker 4.0 to provide over 1 GB of space.

Topic 111

Topic Narrative: *A relevant document will report on efforts by the UN's International Atomic Energy Agency to monitor compliance with the Nuclear Non-proliferation Treaty, or, report on efforts by the United States, Britain, France, USSR, India, or China to control the transfer of technology, equipment, materials, or delivery systems to nations suspected of nuclear weapons development programs, or, a relevant document will report any nuclear activities by Argentina, Brazil, Iraq, Israel, North Korea, Pakistan, South Africa, or Iran (all suspected proliferators).*

Conceptual Graph: [nuclear wannabe: >] → (develops) → [nuclear weapon].

Graph Narrative: *A relevant document will report on nuclear weapons development programs in Argentina, Brazil, Iran, Iraq, Israel, North Korea, Pakistan, or South Africa (all suspected proliferators).*

Topic 122

Topic Narrative: *A relevant document will report on any phase in the worldwide process of bringing new cancer fighting drugs to market, from conceptualization to government marketing approval. The laboratory or company responsible for the drug project, the specific type of cancer(s) which the drug is designed to counter, and the chemical/medical properties of the drug must be identified.*

Conceptual Graph: [company] → (tests) → [cancer drug].

Graph Narrative: *A relevant document will report on any company testing a new cancer fighting drug.*

Topic 129

Topic Narrative: *A relevant document will discuss reported espionage by entities of the Soviet government - KGB, GRU, etc. - conducted within the territory of the United States of America, or against U.S. diplomatic or military facilities overseas. Reported entrapment or involvement of U.S. citizens, residents, or employees in Soviet spying, be it overseas or within U.S. territory, is also relevant. However, espionage cases involving states linked to the USSR - Czechoslovakia, Bulgaria, Cuba, etc. - are NOT relevant, unless linkage to Soviet intelligence can be demonstrated.*

Conceptual Graph: [USSR] → (SpiesOn) → [USA].

Graph Narrative: *A relevant document will discuss any spying conducted by the USSR against the USA.*

Topic 131

Topic Narrative: *A relevant document will provide specified data on contracts awarded to McDonnell Douglas for the production of military aircraft for any nation, any military service, or of any aircraft type (interceptor, fighter-bomber, helicopter, etc.). The contract must be for completed airframes, NOT contracts for aircraft development, factory tooling, components, spare parts, services, etc. To be relevant, the document also must specify the number of aircraft to be delivered, the dollar size of the contract, and the aircraft type sought.*

Conceptual Graph: [McDonnell Douglas] → (WonContractFor) → [military aircraft].

Graph Narrative: *A relevant document will discuss McDonnell Douglas winning a contract for some kind of military aircraft. The contract must be for completed airframes or aircraft development, not factory tooling, components, spare parts, services, etc.*

Topic 150

Topic Narrative: *A relevant document will show how U.S. politicians (federal, state, or local — individually or as a group) pay for their election campaigns, the role played by "special interests" and contributors in the electoral process, allegations or evidence of campaign contributions buying political favors, and/or proposals to limit the cost of campaigns or "reform" electoral finance practices.*

Conceptual Graph: [politician: >=] → (Finances) → [campaign].

Graph Narrative: *A relevant document will discuss a politician financing his or her campaign.*

Figure 3. Narratives for TREC Topics Used in Experiment

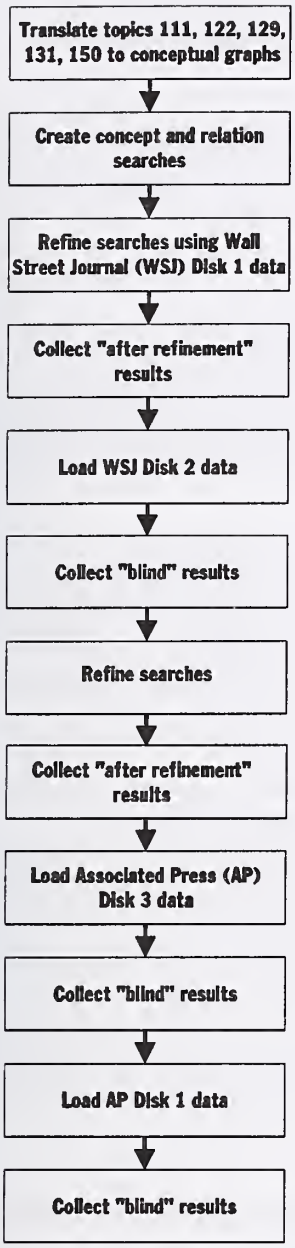


Figure 4. Actual Methodology

3. Experimental Process

This section discusses the details of our data collection process as shown in Figure 4.

To isolate the effects of single relation, a requirement of this experiment is that each query had to be represented by a single concept-relation-concept graph. Most TREC topics require several such graphs for a complete representation (Liddy & Myaeng 1994). We identified five topics that we could "bend" for our use: 111, 122, 129, 131, and 150. Topics 129 and 150 were reasonably well represented by a simple conceptual graph. The substance of Topic 131 was well represented but the topic contained a requirement for more detailed information than the conceptual graph could represent, hence the conceptual graph was more general than the topic. To get two additional topics, we allowed the topic to contain two primary concepts that were linked via multiple relations (Topic 122) or to have a major aspect of the topic representable with a simple conceptual graph (Topic 111).

The topic-graph mismatch for Topics 111, 122, and 131 means that the relevance judgments for these topics are only partially valid. This issue is discussed in more detail in Section 5. The topic narratives, conceptual graphs, and conceptual graph narratives are presented in Figure 3.

Our training collection was Wall Street Journal Disk 1, which was loaded into Personal Librarian. To develop the searches, we reviewed a few of the relevant documents in the test data and used them as guides to develop preliminary concept and relation searches. We then used an iterative process of searching, analysis using the TREC evaluation program, and refinement to improve the individual concept searches. When the concept searches worked reasonably well individually, we began testing the relation using the whole conceptual graph search. All of the relation searches are of the form

%1 *proximity* (*relation search*) *proximity* %2

where %1 and %2 are substituted with the concept searches linked by the in and out arrows, respectively. The two *proximity* operators are the same and the *relation search* is a search for words indicating the relation independent of the concepts. The first focus was refining the relation search using techniques similar to those used for the concepts. Next, we determined the most effective proximity to use. We experimented with same *proximity* operators used later in the experiment —unidirectional and bidirectional operators for 5, 10, 15, 20, and 30 words — collecting average precision results for each. The proximity

operator that produced the best results on the test data was used in the final version of the relation search. One of the benefits of using relations is that recognition criteria such as proximity can be experimentally determined during knowledge engineering.

Once the searches were complete, we collected final results for WSJ Disk 1, which were "after refinement" results. We then loaded the WSJ Disk 2 data and collected "blind" results. Our methodology called for refinement of all searches and subsequent collection of "after refinement" results. We determined that a search needed refinement if the results for a new collection were significantly worse than they had been for an earlier collection and failure analysis indicated that the cause was relation words that occurred in the new collection but not in the earlier one. We actually needed to refine the searches and collect "after refinement" results for Topic 111 only. The performance of the other four topics did not indicate a need for further refinement of the searches, though the performance of Topic 122 indicated other issues. The results from the relation-based search for topic 122 were much worse than for the proximity operators and preliminary analysis indicated that this was largely due to the topic-graph mismatch and the limited number of relevance judgments.

We decided to create custom relevance judgments for Topic 122 and the WSJ Disk 2 data to eliminate these problems. The set of documents judged was the set of 208 documents returned by the broadest proximity operator used in the experiment, NEAR/30. Due to time and funding constraints, judgments were created by one of the authors of this paper rather than an independent judge. We used the conceptual graph narrative for topic 122 as the guide to relevance and focused on being as objective as possible.

Because of the small amount of search refinement that we had to do with the WSJ Disk 2 data, we modified our experimental methodology to use both of the remaining collections for blind results only.

Our third collection was the Associated Press (AP) Disk 3 data used for the official TREC-3 runs. This was the first collection used for blind results only. Our plan was to load all of the AP Disk 3 data. Unfortunately, technical difficulties in indexing caused us to load only one third of that data (four of twelve months). Two of our five relation-based searches — topics 122 and 131 — returned no documents on this collection. We submitted our results to TREC for the three searches that did return documents. Since we needed to have results for eleven searches for each topic (one relation-based search and ten proximity-based searches), our TREC-3 results appear as eleven runs. Within each run are the results for the three topics that worked.

The fourth and final collection used was the AP Disk 1 data. This was the second collection used for blind results only. The relation-based search for topic 131 did not return any documents on this collection. There was one relevant document in the collection, which *was* returned by all of the proximity-based searches. As with the official TREC-3 run, all results from this collection were "blind" and based on the standard TREC relevance judgments (the judgments resulting from TREC-2, not the new TREC-3 judgments).

4. Results

Table 1 is a summary of the results of this experiment, showing the relative improvement of the relation-based search over the best proximity alternative and the relative increase in time required. The detailed performance results are presented graphically in Figure 5 through Figure 9. Performance results are reported as *precision ratios* which allow easy comparison between a relation-based search and its proximity-based alternatives. A precision ratio is computed as

$$\text{precision ratio (operator)} = \frac{\text{average precision (operator)}}{\text{average precision (relation)}}$$

so that the value for the relation-based search is 1.000. Any operator whose performance is greater than 1.000 did better than the relation-based search; any operator whose performance is less than 1.000 did worse.

A few notes on the results:

1. The WSJ Disk 2 results for Topic 111 are the "after refinement" results
2. The WSJ Disk 2 results for Topic 122 for both the standard TREC relevance judgments and our custom judgments are shown.
3. Due to technical difficulties and time constraints, we did not collect time search times for the AP Disk 3 data

Topic	WSJ Disk 1		WSJ Disk 2		AP Disk 3		AP Disk 1	
	Ave Prec	Time	Ave Prec	Time	Ave Prec	Time	Ave Prec	Time
111	+18.7%	+328%	-5.1%	+222%	-13%		+9.7%	+109%
122	-7.3%	+29%	***+55%***	***+26.7%***			+18.6%	+33%
129	***+2073%***	***0%***	***+1200%***	***+5.7%***	+252%		***+676%***	***+3%***
131	***+82.3%***	***+77%***	***+227%***	***+71%***				
150	+8.8%	+164%	+22.2%	+182%	+176%		***+114%***	***+73%***

Table 1: Relative Improvement of Relation vs. Best Alternative
Italics indicate where the percent performance increase is greater than the percent time increase

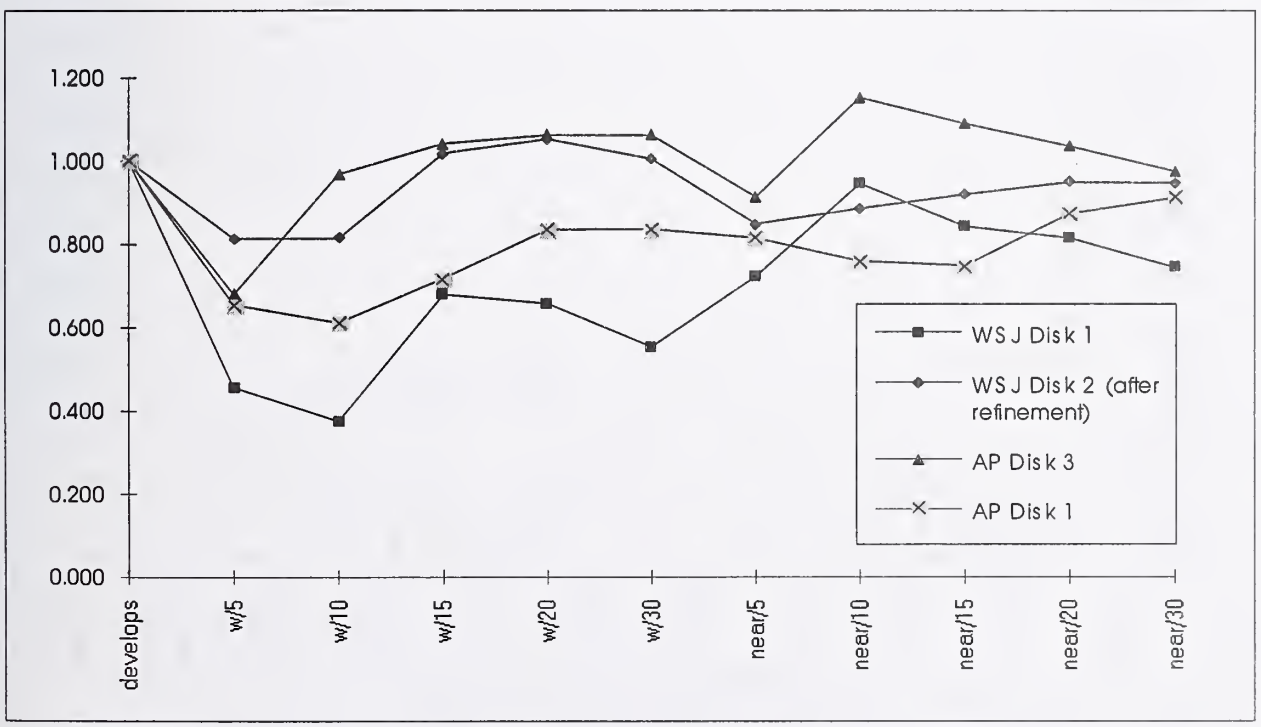


Figure 5. Topic 111 Precision Ratio

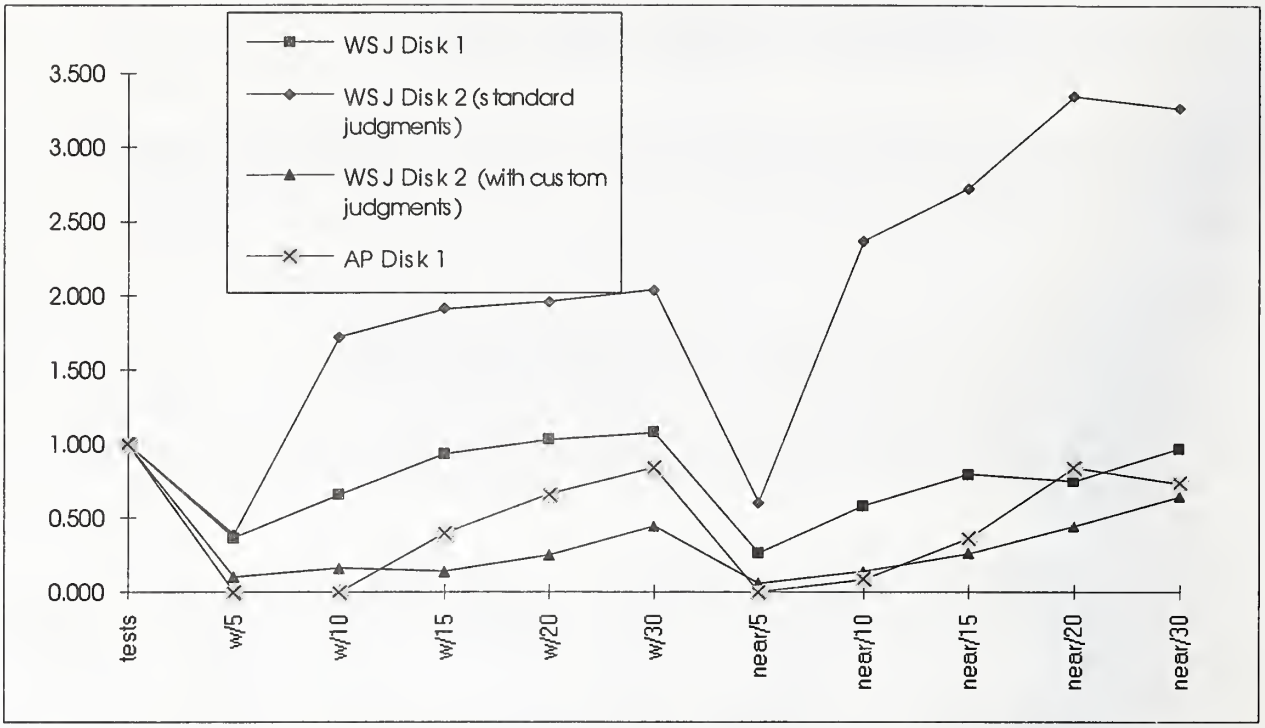


Figure 6: Topic 122 Precision Ratio

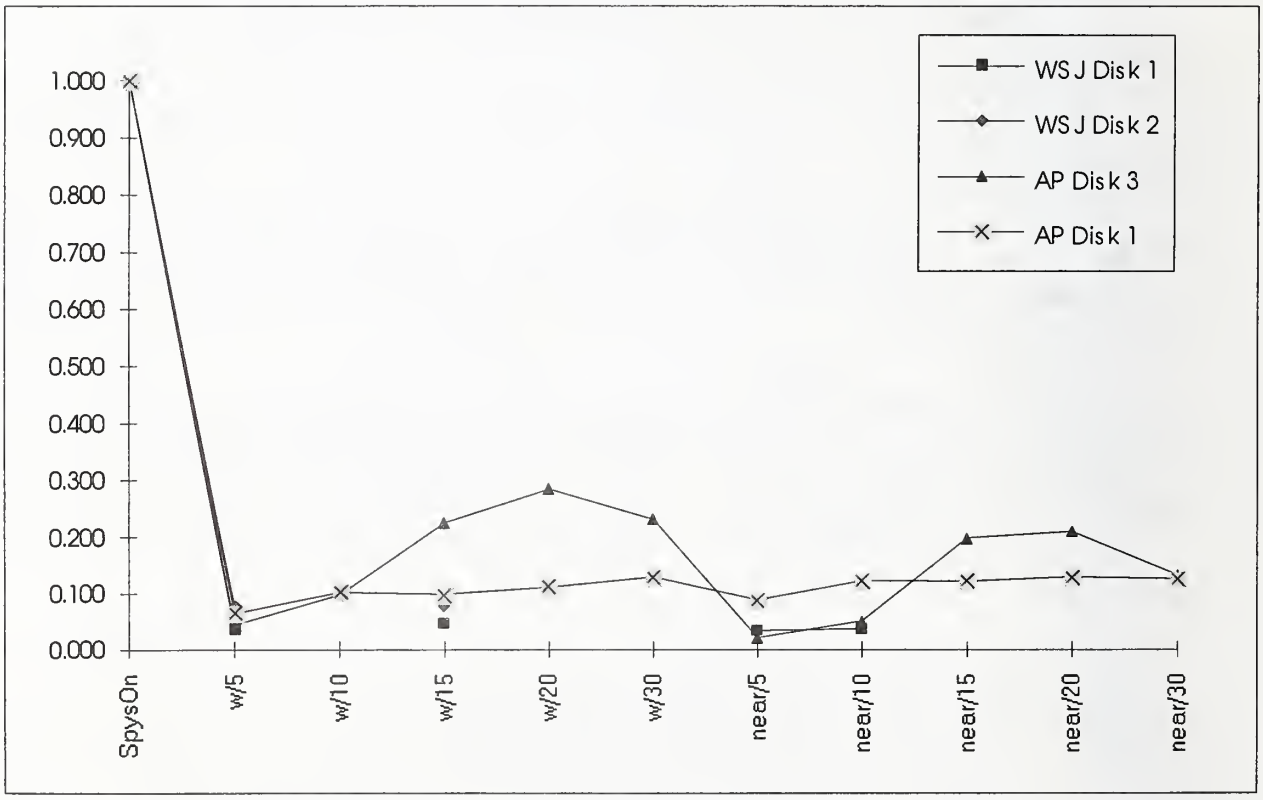


Figure 7: Topic 129 Precision Ratio

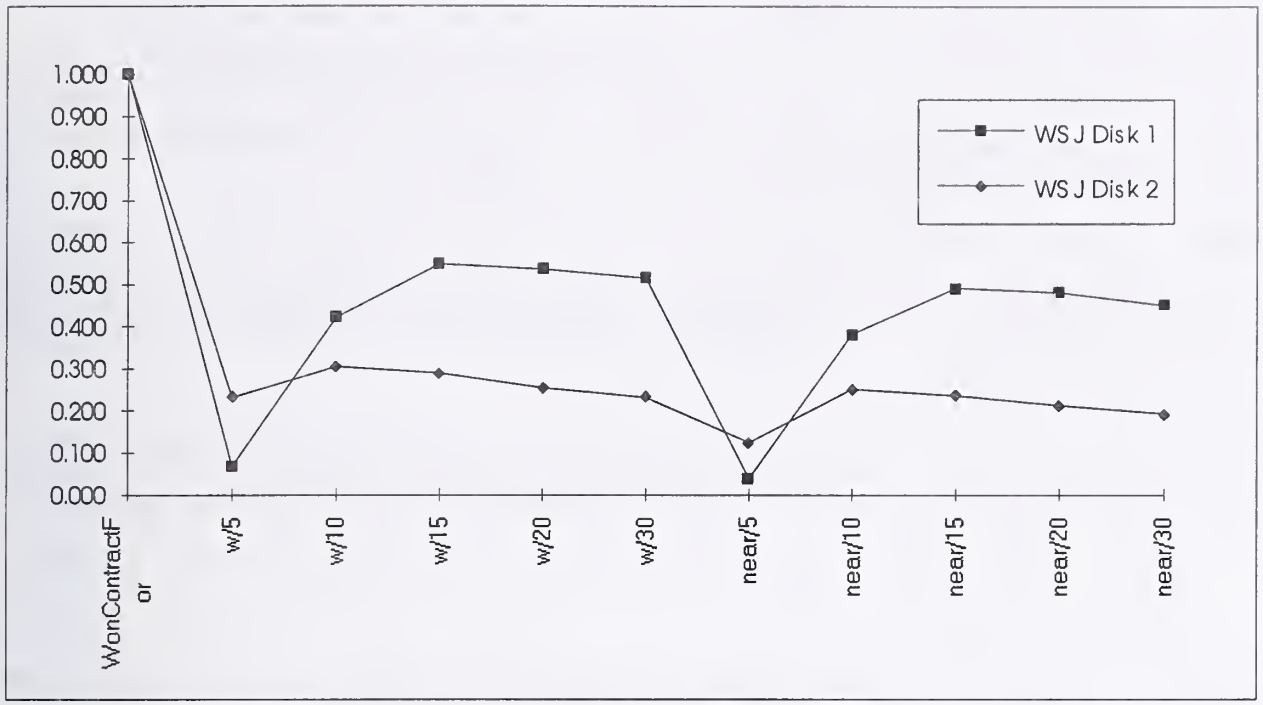


Figure 8: Topic 131 Precision Ratio

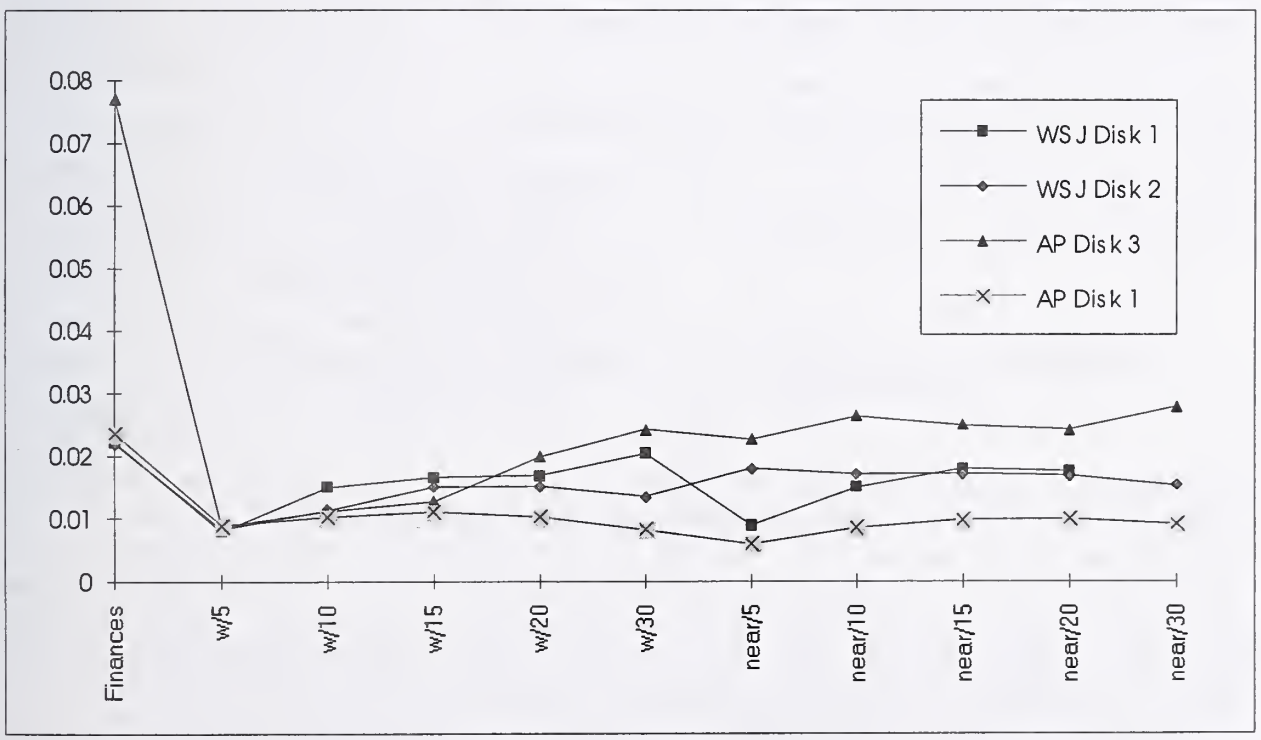


Figure 9. Topic 150 Precision Ratio

5. Analysis of Results

The methodology that we use in this experiment lends itself to more of a qualitative than quantitative analysis. Therefore, we discuss what we learned from each topic separately. The topic narratives, conceptual graphs, and conceptual graph narratives for the topics used in the experiment are shown in Figure 3. Prior to discussing the topics, we discuss our failure analysis process.

5.1. Failure Analysis

Two key issues that affect the validity of our results are the topic-graph mismatch, discussed in Section 2.1, and the limited number of relevance judgments, discussed in Section 2.5. We perform failure analysis on the WSJ Disk 2 results to determine how these issues affect the validity of our results. For each relation-based search we create three lists:

- **Hits.** A hit is a document returned by the relation-based search that is relevant in the TREC judgments.
- **False hits.** A false hit is a document returned by the relation-based search that is not relevant in the TREC relevance judgments.
- **Misses.** A miss is a document that is relevant in the TREC relevance judgments but was not returned by the relation-based search.

For each of these lists, we manually evaluate a random sample of the documents to determine if those documents are relevant to the conceptual graph representation of the topic. The list is sorted by date and every *n*th document is evaluated. Relevance of a document with respect to the conceptual graph is determined using the conceptual graph narrative as a guide. These results are organized in tables. For example, Table 2 shows the failure analysis results for Topic 111. The rightmost column, "% Deviation from TREC Judgments" shows the strength of the evidence that the TREC judgments are not valid for the conceptual graph. The equations for the deviations are as follows:

$$\text{Hits deviation} = 1 - \frac{\#\text{ Hits Relevant to Graph}}{\#\text{ Hits Analyzed}}$$

$$\text{False Hits deviation} = \frac{\#\text{ False Hits Relevant to Graph}}{\#\text{ False Hits Analyzed}}$$

$$\text{Misses deviation} = 1 - \frac{\#\text{ Misses Relevant to Graph}}{\#\text{ Misses Analyzed}}$$

Hits deviation is high when the documents returned by the relation-based search that are relevant to the TREC topic are not relevant to the conceptual graph. Hits deviation is often high when the conceptual graph is more specific than the topic. High Hits deviation indicates that the actual performance of the relation may be worse than computed using the TREC judgments. False Hits deviation is high when many of the false hits are relevant to the conceptual graph. There are two causes of False Hits deviation being high: the conceptual graph being more general than the topic and missing relevance judgments. Misses deviation is high when few of the misses are relevant to the conceptual graph. Like Hits deviation, this is primarily an issue when the conceptual graph is more specific than the topic. High False Hits deviation or Misses deviation indicates that the actual performance of the relation may be better than the value computed using the TREC judgments.

Failure analysis was performed on the WSJ Disk 2 collection. Results of failure analysis are presented in the individual topic discussions.

One limitation of our failure analysis process is that it indicates how the results of the relation-based searches might improve with better relevance judgments but gives no information on how the proximity-based alternatives would be affected.

5.2. Topic 111

The conceptual graph representing Topic 111 is more specific than the topic in that there are several concepts described in the TREC topic narrative that are not subsumed by the conceptual graph: "International Atomic Energy Agency", "Nuclear Non-proliferation Treaty", and "nuclear nations controlling nuclear technology transfer". Documents mentioning these concepts are relevant for the TREC topic but are not relevant to our conceptual graph.

The results for Topic 111 indicate that the relation performed marginally better than the proximity alternatives, although there were situations where proximity was better. For this topic, co-occurrence of concepts in proximity appears to be a good predictor of relevance in itself. Including specific relation words in the search does not appear to have significant benefit for this topic. Discussion of a nuclear wannabe near discussion of nuclear weapons generally indicates a connection between the two.

Failure analysis results for Topic 111 are shown in Table 2. This table shows that 10 items in the Hits list were analyzed, and all 10 of them are relevant to the graph. This means that our judgments and those of TREC agree completely for the Hits list, a deviation of 0%. Of the 10 False Hits analyzed, 7 are relevant to the graph, giving a 70% deviation. To have complete agreement with the TREC judgments, none of the False Hits would be relevant to the graph. Of the 12 Misses analyzed, 4 are relevant, giving a 67% deviation. For the Misses to have complete agreement with the TREC judgments, all of them would be relevant to the graph.

The deviations indicate that the real performance of the relation-based search should not be any worse than reported and may be significantly better. That is, the relation-based search results would likely improve significantly with relevance judgments that are more complete and based on the conceptual graph rather than the TREC topic. Due to time and resource constraints, we were unable to create custom relevance judgments for this topic as we did for Topic 122.

List	# Analyzed	# Relevant to Graph	% Deviation from TREC Judgments
Hits	10	10	0%
False Hits	10	7	70%
Misses	12	4	67%

Table 2. Topic 111 Failure Analysis Results

5.3. Topic 122

The conceptual graph for Topic 122 is more specific than the topic in some regards and more general in others. The conceptual graph is more specific in that it does not cover any phase of development other than testing, such as research, evaluation, or marketing. We chose to focus on "testing" for this topic because it is a clearly defined, specific relation. The conceptual graph is more general than the topic in that it does not require as complete information about the specific types of cancers the drug is designed to counter and the chemical/medical properties of the drug. Searching for these details would required more than a single concept-relation-concept graph.

Using the standard judgments, the performance of the relation-based search ranges from slightly worse to much worse than the best proximity alternative.

Failure analysis results for Topic 122 are shown in Table 3. The Hits deviation indicates that the real performance of the relation-based search might be significantly worse than reported. The False Hits and Misses deviations indicate that the performance might be significantly better. These failure analysis results strongly indicate that the conceptual graph is a poor match for the relevance judgments. To obtain more valid performance results, we created custom relevance judgments for this topic.

The graph in Figure 6 shows the dramatic difference in the performance reported using the standard judgments versus using the custom judgments. Using the custom judgment results, the relation-based search performance is significantly better than the proximity-based alternatives.

List	# Analyzed	# Relevant to Graph	% Deviation from TREC Judgments
Hits	7	3	57%
False Hits	10	7	70%
Misses	10	0	100%

Table 3. Topic 122 Failure Analysis Results

5.4. Topic 129

Topic 129 was easily representable as a simple concept-relation-concept graph. The failure analysis results shown in Table 4 show no deviation of any kind, suggesting that the results are valid.

The relation-based search performed much better than the proximity operator alternatives. The proximity operators, especially the broad ones, returned thousands of documents. Writing these results to a file took as much as an hour for each operator tested. Early results were so dramatic and collecting data for each operator took so long that results for some operators were not collected.

Topic 129 is an ideal topic for using relations. The two concepts USA and USSR co-occur frequently in the collections with a very large number of relations. Specifying the relation led to large performance improvements.

List	# Analyzed	# Relevant to Graph	% Deviation from TREC Judgments
Hits	4	4	0%
False Hits	10	0	0%
Misses	3	3	0%

Table 4. Topic 129 Failure Analysis Results

5.5. Topic 131

While the conceptual graph representing Topic 131 effectively captures the primary meaning of the topic, it does not require *"the number of aircraft to be delivered, the dollar size of the contract, and the aircraft type sought"*. Hence, the conceptual graph is more general than the topic: there are documents that are relevant to the conceptual graph but not relevant to the topic.

The performance results for this topic show that the relation-based search performed significantly better than the proximity-based alternatives. Topic 131 is a good topic for using relations. The two concepts co-

occur frequently with many different relationships. One of the most common relations is simply the implicit indication that McDonnell Douglas is the builder of the aircraft mentioned: *". . . a McDonnell Douglas F-15 . . ."*

The failure analysis résults shown in Table 5 suggest that the real performance of the relation-based search was no worse than reported and is likely to be considerably better. None of the Hits are not relevant to the graph but many of the False Hits are relevant to the graph and many of the Misses are not relevant to the graph.

List	# Analyzed	# Relevant to Graph	% Deviation from TREC Judgments
Hits	4	4	0%
False Hits	11	6	55%
Misses	8	3	63%

Table 5. Topic 131 Failure Analysis Results

5.6. Topic 150

Like Topic 129, Topic 150 was easily and effectively representable as a simple conceptual graph. This topic showed interesting behavior in that the performance of the relation-based search was only marginally better than the alternatives for the first two collections. This surprised us, as the concepts politician and campaign seem to be ones that would frequently co-occur with many different relations: finances, starts, makes vows in, discusses issues in, etc. After some consideration, we realized that the first two collections were both Wall Street Journal, which is fundamentally focused on financial issues. We hypothesized that performance of the relation-based search would be much better on a collection with a broader focus. The tests run on the Associated Press data support this hypothesis: the relation-based search performs much better than the alternatives.

The failure analysis results shown in Table 6 indicate that the performance results for this topic are valid. The one Hit document that was judged not relevant to the conceptual graph was marginally relevant to both the topic and the conceptual graph. That TREC judged it as relevant and we judged it as not relevant appears to be more a variation between judges than a graph-topic mismatch.

List	# Analyzed	# Relevant to Graph	% Deviation from TREC Judgments
Hits	10	9	10%
False Hits	11	0	0%
Misses	10	10	0%

Table 6. Topic 150 Failure Analysis Results

6. Conclusions

Can retrieval performance be improved by explicitly specifying and searching for relationships between concepts?

The data we collected provides evidence that, in some cases, retrieval performance can be improved by explicitly searching for relationships. In all but two of our tests, relations provided improved retrieval performance over all of the proximity-based alternatives. In the best case, performance was twenty times

better. In the worst case, performance was only slightly worse than the best alternative. In these worst cases, failure analysis suggests that the relation would have performed significantly better than the alternatives if we had used complete relevance judgments based on the conceptual graph rather than the topic. Hence, we believe that relations will often improve performance.

Under what conditions is the improvement likely to be justified?

Our experience suggests that using a relation for retrieval provides the greatest performance improvement when the two concepts it links frequently occur in proximity to each other with several different relationships. The effectiveness of relations is dependent on the concepts, the relation, and the data being searched. Some pairs of concepts, like *nuclear wannabe* and *nuclear weapon* (Topic 111), inherently have a small number of relationships between them. For these concepts, using relations will generally provide a small increase in performance. Other pairs of concepts inherently have a large number of relations, such as *U.S.A.* and *U.S.S.R.* (Topic 129). With these concepts, specifying a relation provides a large increase in performance and, for many purposes, may be essential to getting usable results. For some pairs of concepts, such as *politician* and *campaign* (Topic 150), the number of relations depends on the nature of the document collection. In this case, using a relation that is very common (such as *Finances* in the WSJ collection) provides relatively little performance gain but using a less common relation provides significant benefit.

The cost and value of using relations are both subjective measures that depend on the system and the user's needs. There are at least three costs associated with using relations: reduced recall, increased retrieval time, and the cost of building the supporting knowledge base.

Like any method of improving precision, relations do so at the expense of recall. The difference is that, for a given improvement in precision, relations may cause less degradation of recall than other methods, such as tightening proximity. Use of relations may provide significant benefit to a user whose focus is on precision or a balance of precision and recall. For a user who requires high recall, the degradation in recall caused by relations may not be acceptable.

Explicitly searching for relations increases the complexity of the search expression and therefore increases the processing time cost associated with the search. In Table 1 we show the percentage increase in performance provided by the relation-based search compared to the best proximity alternative. We have italicized the cases where the relative increase in performance exceeds the relative increase in time. This cutoff point is not necessarily the point where the time cost of using relations exceeds the value. A 300% increase in time is probably cheap if the current response time is 0.1 second but prohibitive if the current response time is 30 seconds. Similarly, a 30% improvement in performance is more significant when there are 1000 documents returned than when there are 10. In Table 1 note that the greatest performance improvements have the lowest time cost while the smallest performance improvements tend to have high time cost.

The approach to recognizing concepts and relations that we used in this experiment requires a knowledge base that takes substantial time and expertise to construct. The number of concepts and relations required in such a knowledge base depends on the domain to be searched but could easily be in the thousands or tens of thousands of concepts and hundreds of relations. If each concept and relation takes one hour to define, a knowledge base of 10,000 concepts and 200 relations would require 4.6 person years to construct.

7. Research Questions For Future Work

While the results of this experiment indicate that relations provide significant performance advantages, there are several related questions that could affect the ability to use this approach in a retrieval system.

Will our relation definitions work with other concepts? We defined five relations, each of which was used for a single pair of concepts. Are the searches for these relations general enough that they will work properly with a wide variety of concepts? If not, can they be redefined so that they do?

Can human users consistently identify the same relation for the same information need? One use of relations is to have a list of concepts and relations and allow the user to build a conceptual graph representing his or her information need. Given an information need, two specified concepts, and a list of relations, would users consistently identify the same relation between the concepts? What information or mechanisms could be provided to aid a user in selecting the correct relation?

How well can directionality be detected? Conceptual graphs are, by definition, directional. This is essential, as there is a major difference between the U.S.S.R. spying on the U.S. and the U.S. spying on the U.S.S.R. The relation-based searches in this experiment all used bidirectional search operators and therefore ignored directionality. What can be done to better represent directionality in relation-based searches?

Can relations be identified more effectively using a different (non-Boolean and proximity-based) search mechanism? In this experiment we used Boolean searching. How might other search mechanisms be used, and how do those results compare to ours?

Does explicitly specifying relations help users better frame their information need? That is, does the process of selecting a relation from a list or otherwise explicitly specifying the relation help the user to better understand what he or she is looking for?

8. Acknowledgments

The authors would like to thank Donna Harman for her patience and flexibility with our non-standard TREC experiment, Personal Library Systems for the use of Personal Librarian for this experiment, and Howard Turtle for his comments on our project plan and an earlier draft of this paper.

9. References

Brooks, B.C. (1986). Jason Farradane and relational indexing. *Journal of Information Science, 12,* 15-18.

Cimino, C. & Barnett, G.O. (1992). Analysis of physician questions in an ambulatory care setting. *Computers and Biomedical Research, 25*, 366-373.

Dick, Judith P. (1991). On the usefulness of conceptual graphs in representing knowledge for intelligent retrieval. *Proceedings of the Sixth Annual Workshop on Conceptual Graphs. Binghamton, NY, July 11-13, 1991.* 153-167.

Farradane, J. (1980). Relational indexing. *Journal of Information Science, 1,* 267-276, 313-324.

Farradane, J. & Thompson, D. (1980). Testing relation indexing by diagnostic computer program. *Journal of Information Science, 2*, 285-297.

Liddy, E.D. & Myaeng, S.H. (1994). DR-LINK: A system update for TREC-2. In Harmon, D.K. *Proceedings of the Second Text REtrieval Conference (TREC-2),* NIST Special Publication 500-215, 85-99.

McCune, Brian P., Tong, Richard M., Dean, Jeffrey S., & Shapiro, Daniel G. (1980). RUBRIC: A system for rule-based information retrieval. *IEEE Transactions on Software Engineering, SE-11(9),* 939-945.

Sowa, J.F. (1984). *Conceptual Structures: Information Processing In Mind and Machine.* Reading, MA:Addison-Wesley.

The FDF Query Generation Workbench

K-I. Yu **P. Scheibe** **F. Nordby**

Paracel, Inc.
Pasadena, CA

The TREC-2 experiments with the FDF-3 clearly showed the power of automated query generation from relevance information; the broader TREC-2 results also suggest that there may be no single "best" query generation method. Building on this experience, Paracel has begun the design and implementation of a Query Generation Workbench for the FDF-3, to provide a structure for implementing and selecting among a variety of relevance-based query generation methods. TREC-3 has provided an early large-scale test of this prototype, and we have used the opportunity to begin a study of the first generation of tools for the Workbench.

1. Introduction

In TREC-2, two methods were used to generate queries for the FDF-3 hardware text search system: for one a human expert manually generated queries for all 50 topics; the other used an automated term weighting scheme, augmented by operator-supplied suggestions. The results of this experiment were clear: the automated system produced superior results in less human time — including the time required to implement the system!

More broadly, the diversity of systems and results at TREC-2 suggests to us that there may not be any one "best" automated query generation system. Certainly there were systems that performed well overall; there were also many systems — including the human-generated FDF-3 queries — that showed superior performance for some specific topics.

Based on these observations, Paracel has begun the design and implementation of a Query Generation Workbench. This Workbench has two complementary aspects: in the broad sense, the Workbench is a structure for the implementation, evaluation and combination of different query generation methods; in the narrow sense, the Workbench comprises a collection of tools which can be combined within this structure to implement different query generation variants, and a query selector which can be used to automatically choose among a variety of query generation methods based on estimated performance for a particular query generation task.

TREC-3 has provided an early large-scale test of the Workbench. For this first test we populated the Workbench with a variety of conservatively-designed and -implemented tools; then we set the system on automatic and let it run its course. Thanks to the modular structure of the Workbench, and thanks to the operation of the query selection stage of the Workbench, we immediately derived an extra result: in addition to a set of queries, the selector returned a parameter-by-parameter, tool-by-tool, topic-by-topic comparison of the different query generation tools in the Workbench. We have begun an analysis of these data, and the preliminary results have shown some interesting patterns. We are now working to incorporate these results into the Workbench and its component tools.

```
{ 1 document -> return 1+ (
    'guilty' : 12 ;
    'insider''trading' : 21;
    'ivan''boesky' : 26 ;
    'trading' : 18 ) }
```

Figure 1. A simple FDF weighted set query. If the term "`'guilty'`" is matched once or more in a document, then 12 is added to the score for that document; similarly, if "`'ivan''boesky'`" is matched, 26 is added to the document score. With this query, the maximum score for any single document is 77 (i.e., 12+21+26+18).

2. The Query Generation Workbench

The goal of Paracel's FDF Query Generation Workbench project is to develop an "industrial strength" system for automating query construction from relevance data. While this system is oriented towards fully automatic query generation from relevance data, it also allows for user input at various stages of the query generation process. Finally, since the TREC results suggest that there may be no one "best" relevance-based query generation method, the Workbench makes provision for case-by-case selection among a collection of query generation methods and parameters.

The major early result of this project is a prototype Workbench architecture. This architecture is a uniform, modular framework for relevance-based automatic query generation. The Workbench structures query generation in stages, and provides plug-and-play use of different tools at each stage, allowing piece-by-piece customization of the query generation process. Also, since one of the stages in the Workbench is a query selection, or simple query-level data fusion, the Workbench allows for automatic parameter optimization and tool selection within the same framework.

The Workbench structures query generation as a four-stage process. This process starts with the enumeration of candidate terms: these terms may be anything for which the FDF search engine can match. From the candidate terms, a subset is selected for use in the query; then the selected terms are weighted and built into an a query for the FDF search engine. Finally, given a collection of queries — typically the collected results from various tools and parameter values in the first three stages — the performance of each query is estimated, and one query is selected for actual use as routing profile. In summary, the query formulation stages used in the Workbench are:

1. Enumerate a set of terms (words, word pairs, etc.).
2. Select a subset of the terms enumerated in step 1 for use in the final query.
3. Choose a weight for each term selected in step 2 and use these weights to formulate a query.
4. Given several queries using a variety of selection and weighting criteria in steps 1, 2 and 3, estimate the accuracy of each of these queries and choose the best.

In the first version of the Workbench, the stages have been populated with tools based primarily on a probabilistic model, and the tools currently all produce queries in a single form most suited to this model (figure 1 for a sample query). We are now working to incorporate methods from vector-space and other models into tools, so that we may use these methods within the framework of the Workbench.

2.1 Candidate Term Enumeration

The first stage of the Workbench is term enumeration: this stage simply lists a collection of relevance cues for possible use in query formulation. The terms produced by the present term enumeration tool are quite simple: words and word sequences[1] from the text of known relevant documents. In this process, case distinctions are ignored, but no stemming is performed. The enumeration tool is parameterized by the lengths of the allowed word sequences.

2.2 Term Selection

Given a list of candidate terms, the next stage of the Workbench is term selection. The first step in this process is stop word elimination: to this end, any term which either begins or ends with a "stop word" is discarded. For the TREC experiments, a "stop word" is either any word from a list of common English words, or any word found in more than half of the documents of any one of the TREC corpora.

Next, the term selectors gather data on the occurrences of the candidate terms, both in the relevant set and in the training database as a whole. These collected statistics are used to calculate a "selection score" for each of the candidate terms by any of a number of formulae, and the terms are sorted according to score. The prototype toolkit includes several term selection functions; for the TREC experiments we used two of these:

1. The odds of document relevance given term presence ("O"): $P(R \mid w_i = 1)$;
2. A heuristic ("VxLD") based on the normalized deviation of the term occurrence frequency and the in-document frequency.

Finally, given this ranking of the terms, some number are chosen from those with the highest scores: this selection process may impose other restrictions, such as a limit on the number and lengths of phrases used in a query.

2.3 Query Construction

The third stage of the Workbench is query construction. All of the query construction tools which currently populate the Workbench produce queries of the same general form (e.g. figure 1). To build queries, the tools use many of the same statistics gathered in the term selection process: from these data, the tools weight each selected term, then build a query is built to approximate of the desired scoring function. For the initial tool set we implemented a few simple term weighting functions; for TREC-3 we used two:

1. The odds of relevance given term presence ("O"), $P(R \mid w_i = 1)$; and
2. The log of the relative likelihood of term presence vs. document relevance ("LLR"), $\log\left(\frac{P(w_i = 1 \mid R)}{P(w_i = 1 \mid \bar{R})} \cdot \frac{P(w_i = 0 \mid \bar{R})}{P(w_i = 0 \mid R)}\right)$.

1. A.K.A. "phrases". This terminology should not be taken to suggest that these sequences necessarily have any linguistic significance; these "phrases" are just sequences of words which appear in the source text.

	Query #1	Query #2	Query #3	Query #4	Query #5
Subset #1	19 (1.0)	16 (3.0)	15 (5.0)	16 (3.0)	16 (3.0)
Subset #2	12 (5.0)	14 (2.0)	13 (4.0)	14 (2.0)	14 (2.0)
Subset #3	17 (5.0)	19 (3.0)	18 (4.0)	20 (1.5)	20 (1.5)
Subset #4	22 (1.0)	20 (3.5)	20 (3.5)	21 (2.0)	18 (5.0)
(Total)	**(12.0)**	**(11.5)**	**(16.5)**	**(8.5)**	**(11.5)**

Table 1. Query selection table for five methods on topic #125. Each column of the table shows the results for one query formulation; each row shows the results for one subset of the corpus. The entries (e.g., "14 (2.0)" for query #2, subset #2) show the recall at 50 documents retrieved for that query on that subset of the training set, and the ordinal rank of that result over all five query formulations. The final row, labeled "(Total)," shows the summed ranks for each query. The query selector picks the query with the lowest summed rank — in this case, query #4.

There are also a number of adjustments which may be made to the term list and the term scores in this stage. The first, and simpler, of these adjustments is a heuristic which we implemented to test the effect of adding a bonus for multiple term occurrences. To this end we doubled size of the query: under this adjustment if a term is found once in a document, the standard term weight is added to the document score; if the term is found twice or more in a document, a 50% bonus is added.

The other term weighting adjustment which is available is a compensation for the presence of sub-phrases in a query. For example, consider the query shown in figure 1: if a document includes a match for "`''insider''trading''`" then it certainly includes a match for "`''trading''`" — thus the presence of the sub-phrase adds no new information. If this query were subjected to phrase weight adjustment, the weight of the superphrase ("`''insider''trading''`") would be decreased by the combined weights of all its sub-phrases included in the query — the final weight would be 3 in this case.

2.4 Query Selection

The final stage in the Workbench is query selection: this stage might described as simple query-level data fusion. The inputs to this stage are any number of queries for the same topic, built with different term enumeration, term selection and query construction methods: from these queries, the Workbench selection tool chooses one query which shows the best estimated performance on the training collection.

To accomplish this, the selection tool divides the test collection into four subsets. Each of the candidate queries is run against the four training subsets, and the performance of each query on each subset is measured as the recall at 50 documents retrieved. The queries are assigned ordinal ranks for each subset, and these ordinal ranks for each query are summed to form an overall query score. The query with the lowest score is then selected. Table 1 shows an actual selection table for one of the TREC-3 topics.

2.5 Practical Matters

The process just described is, in terms of functional results, exactly what is done in the TREC query construction process; however, in practical terms, a number of optimizations

are applied. These optimizations impact the performance of the query generation process, but do not reflect on its overall behavior.

For example, a list of all the words in the training corpus is pre-computed and shared among the query construction processes for all topics. Also, word frequency data is collected in the compilation of this word list: this help in determining stop words and computing term selection scores and term weights. This index-style statistics collection works well for single words, but for it would be impractical to maintain statistics for every phrase found in the corpus. Instead, the term selectors compose queries and use the FDF to gather the desired statistics. (The statistics are passed on to the query constructors, so they need not re-collect the data.)

2.6 The TREC-3 Experiments

For this year's TREC experiments we wanted to begin a systematic evaluation of the results from last year's automated FDF query generation results. Accordingly, one of the inputs to the query selector is an emulation of last year's automated query generator: this method phrases up to three words, 50 terms, the "O" selector and the "O" constructor; also, the selector was restricted to only those phrases found in at least 30% of the relevant documents. The method did not use either phrase weight adjustment or a multiple-occurrence bonus.

All of the other inputs to the query selector used the "VxLD" term selector and the "LLR" query constructor with phrase weight adjustment: we had designed these tools to implement a simple log-likelihood model query generator, fixing the "mistakes" in TREC-2 query generator. Here again, the term selector was restricted to those phrases with a recall of at least 30%. In some cases, phrases up to three words were allowed; in others, only single-word terms were included. Term counts were varied from 32 to 64, and in some cases the number of phrases in the final query was restricted. Finally, multiple occur bonuses were used in some of the cases.

3. Results

We have derived two sets of results from the TREC-3 process: first, of course, the formal TREC results provide a gross comparison of the Workbench results to the other TREC systems; second, the data from the query selector show in detail the strengths and weaknesses of the various query generation tools. These latter results in particular are extremely valuable to the tool development process, as they allow point-by-point, parameter-by-parameter, stage-by-stage comparison of different query generation methods.

3.1 Query Selection Results

This year, to provide a reasonably close comparison against last year's results, we included a similar query formulation method as one input to the final query selection step. To this, we added a number of query formulations all based on the VxLD selection and the LLR weighting (with phrase weight correction) formulae; we used these because of limited time, and because our preliminary results suggested that these were over-all the best performers. For these alternative query formulations, we varied several of the secondary parameters: term count, phrase count, etc.

Overall, the VxLD/LLR methods far out-performed the O/O method: seldom was the O/O method chosen by the query selector, and often the O/O method showed the worst performance of all the methods. We have not yet performed an statistical analysis of this comparison, but we expect that the VxLD/LLR methods will be found to be generally superior to the O/O method with strong statistical significance.

That said, there were a few cases in which the O/O query far out-performed the VxLD/LLR queries. We are looking at the terms to try to determine what aspects of the O/O query formulation are responsible for these cases, to see if we can derive a hybrid query formulation method.

Beyond these broad results, there are many more details; to summarize just a few:

- Phrases can be very beneficial, as long as they don't replace good single-word terms in the query.
- Along the same lines, phrases of four words or longer tend to be superfluous: the shorter two- and three-word sub-phrases will already be in the query, and the four-word phrases will add no effective precision to the query.
- The VxLD heuristic for term selection seems to work quite well — the term lists seem, subjectively, appropriate to the topic — and they appear on the whole to provide good recall. More research is needed here, however.
- Similarly, the LLR query builder worked better than the O query builder, on the whole; however, as mentioned above, there were a few cases where the O queries outperformed the LLR queries. We don't yet know wether this is an artifact of term selection or weighting.
- Phrase weight adjustment was a powerful tool — sometimes increasing precision at fixed retrieval levels by a factor of two or more with all other parameters held constant.
- Multiple occurrence bonuses often but not always had a positive effect on query performance. More research and a more rigorous model are needed here. Also, new query construction tools are needed to build queries which reflect a broader range of query generation methods.
- The query selector is too simple: use of precision at 50 documents as ranking criterion should probably be replaced by another (perhaps selectable) metric — for example, one which estimates average precision. Also, some more care needs to be given to ensuring unbiased performance estimation in the query selector.

3.2 TREC Scores

Our analysis of the official TREC results is on-going. In gross terms, the overall performance of the Workbench was approximately median — measured as precision at 1000 documents, for example, the Workbench scored at or above median for 26 of the topics and below median for 24 of the topics. We are currently examining the results in greater detail, and we hope to learn more from that analysis.

4. Conclusions and Acknowledgments

Our efforts have been focussed on developing a framework for query generation, and populating that framework with an initial set of tool. The results have shown us a great deal about the component effects present in the query generation process, and we are learning more as we continue this work.

As we have shown, the avenues for further research and development in the FDF Query Generation Workbench are many and broad. In the main, we are working to incorporate the major lessons learned into the structure of the Workbench and its constituent tools; and we are working to extend the comparisons we began with our TREC-3 experiments. Also, since the FDF search engine is entirely language independent, we are starting to apply the Workbench to non-English query generation problems. Along these lines, we expect to participate in the non-English TREC exercises next year.

We would like to express our gratitude to the TREC sponsors and participants alike. The former have provided an excellent testbed for research and development in text retrieval systems; the latter have sown this fertile ground with ideas. We hope to continue to benefit from this abundance in the future.

Report on the TREC-3 Experiment: A Learning Scheme in a Vector Space Model

Jacques Savoy, Melchior Ndarugendamwo, Dana Vrajitoru

Faculté de droit et des sciences économiques
Université de Neuchâtel
Pierre-à-Mazel 7
CH - 2000 Neuchâtel (Switzerland)

Summary

This paper describes and evaluates a retrieval scheme, or more precisely an additional retrieval mechanism based on interdocument relationships, that can be integrated in almost all existing retrieval schemes (e.g., Boolean, hybrid Boolean, vector-processing or probabilistic models). The intent of our approach consists of inferring knowledge about document contents based on the relevance assessments of past queries. Through a learning process, our scheme establishes relevance links between documents found relevant for the same request. Based on this information and a list of retrieved records for the current request, the proposed mechanism tries to improve the ranking of the retrieved items in a sequence most likely to satisfy user intent. The underlying hypothesis of this mechanism states that future requests addressed to the system should have some degree of similarity with previous queries, or that the retrieval apparatus will process requests for which it has already found a partial, appropriate answer in the past.

Participation: Category: B Query: ad-hoc, fully automatic

Introduction

To find pertinent information from a large text collection, most retrieval models represent both documents and requests by a set of weighted keywords. To extract relevant records from this collection, the retrieval function computes a similarity value or estimates a probability of relevance based on both document and query surrogates.

When applying such a scheme, the system considers documents as separate entities. To relax this assumption, some studies have proposed various techniques and have reported evaluations describing the importance of interdocument relationships (e.g., [Kwok 88], [Turtle 91]). Our main research objective is also to analyze and assess interdocument relationships as a useful source of document contents evidence. In this vein, we have already investigated the relative importance of explicit (e.g., bibliographic reference), implicit (bibliographic coupling, co-citation) and computed links (nearest neighbor) between documents [Savoy 94a]. In this study, we are concerned with the means by which the system may have derived other relationships between documents based on past queries and their relevance assessments.

This paper is made up of two sections. The first describes our learning scheme and presents some related works. The second section shows and explains results obtained using the Wall Street Journal corpus, a subset of the TIPSTER-DARPA collection, and discusses some problems related to traditional evaluation methodology.

1. Learning Scheme

Evaluation of current retrieval models has shown that their retrieval effectiveness is far from perfect and one of the principal explanations of this lack of effectiveness is related to the ambiguity of natural language. This problem has two facets: on the one hand, the same idea or concept may be expressed by various forms [Furnas 87], and, on the other hand, the same word may have more than one meaning, even in a specialized corpus [Krovetz 92].

In order to resolve this difficulty, Blair [90] suggests that a retrieval model must have better document contents representation:

> "The central problem of Information Retrieval is how to represent documents for retrieval. The most intricate or carefully designed retrieval algorithm cannot compensate for inappropriately represented documents. ... The central task of Information Retrieval research is to understand how documents should be represented for effective retrieval.

This is primarily a problem of language and meaning." [Blair 90, vii]

This may lead to a perception of the retrieval system as an adaptive process, allowing better communication between the searcher and the indexer (or the author(s)) [Blair 90].

From a practical point of view, one feasible approach to the design of such a learning scheme consists of taking into account the knowledge obtained from past queries, or more precisely, from their relevance judgments, in order to enhance system's retrieval effectiveness over time. We also believe that documents found relevant for a given request do share similar concepts [Savoy 94b]. Thus, past queries and their relevance assessments may be a useful source of information about the meaning of documents and may be helpful in ranking the retrieved records in a sequence that more closely reflects the user's intent.

The first subsection describes the main principles underlying our learning scheme. The second presents the design of our adaptive model and the guidelines for its implementation. The third subsection shows statistics related to query similarities and the fourth one describes the main features of related researches.

1.1. Motivation

The aim of a learning scheme is to provide the system with the ability to record its successes and failures, and thus infer knowledge useful for increasing its performance over time. To define such a mechanism, we have to specify the underlying hypotheses, determine how the system learns and how it stores and uses the knowledge provided by previous experiments.

Our learning scheme is based on the following principles:

a) Documents known to be relevant to the same query tend to contain similar concepts and must deal with similar subjects;
b) No conclusions can be drawn about documents found nonrelevant for a given request.

On the one hand, our learning scheme is based exclusively on successes, i.e., on the presence of couples of retrieved and relevant documents. On the other hand, our procedure does not take into account the shared presence of retrieved and nonrelevant items. Nonrelevant records retrieved by the system are those documents that have at least one common keyword with the request. However, such keyword matching does not always imply word sense matching:

"Word sense mismatches are far more likely to appear in nonrelevant documents than in those that are relevant." [Krovetz 92, p. 139].

Our prior feeling is that negative relevance feedback information does not really represent useful information. By analogy, if you are lost in a desert, a negative relevance feedback only tells you that "you are on the wrong path" and does not provide "efficient" hints as to the path leading to the for i = 1, 2, ... nearest city. This fact is confirmed by relevance feedback studies which have demonstrated that positive relevance information depicts more valuable information than negative one [Salton 90]. However, this approach considers only relevance data given for the current request and a direct comparison with this scheme is therefore not suitable.

1.2. Implementation of our Learning Scheme

In order to represent the information given by the previous experiments or requests, we have designed a special interdocument relationship called a *relevance link*. This link type connects two documents found relevant for a given query. Associated with each link, a *relevance value* specifies how many times both the linked documents are found relevant.

To account for the information provided by the learning stage, our retrieval scheme works in two phases. In the first, the retrieval status value (RSV) of each document is computed according to a well-known retrieval scheme. To achieve this step, one can use the p-norm model, a vector-processing scheme or a probabilistic retrieval strategy. In the second stage, the ranking of retrieved documents is modified according to the presence of relevance links according to the following equation.

$$RSV(D_i) = RSV_{init}(D_i) + \sum_{k=1}^{s} \alpha_{ik} \cdot RSV_{init}(D_k) \quad \text{for } i = 1, 2, ..., m \qquad (1)$$

in which α_{ik} reflects the strength of the relationship between Documents i and k and s the number of neighbors of Document i. At the initial stage, the retrieval status value of a document depends only on the similarity between its surrogate and the query ($RSV_{init}(D_i)$, computed according to the Vector Space Model presented later in this paper). The value α_{ik} can be either a constant or a

function of the relevance value of the link connecting Documents i and k.

To illustrate the way our retrieval proposal works, Figure 1 depicts some relevance links with their respective relevance values. In the first step of our retrieval process, a vector-processing scheme attributes a retrieval status of 0.8 to Document 11. According to Formula 1, this weight is propagated through links to Documents 3, 7 and 10. If we define the strength of the link between Nodes 11 and 7 as 0.3, Document 7 will increase its retrieval status value by 0.24.

In order to improve the efficiency of our retrieval scheme and to guarantee a reasonable processing time, we modify the retrieval status value not for all retrieved records, but we select the first m best-ranked documents after the initial stage to activate the relevance links (the constant m in Equation 1).

We believe that relevance links indicate semantic relationships between documents and may be valuable in the searching process. Although Blair [90] considers such a scheme to be a useful pedagogical tool, he questions its retrieval effectiveness:

> "Bush [45] recognized early ... how inquirers could benefit from the "traces" left by searches conducted by informed inquirers. While this is an important notion, realistically each inquirer's searches are unique enough that a record of previous searches might only be marginally useful for finding specific information." [Blair 90, p. 181]

The main underlying hypothesis of our retrieval model is that coming requests have some relationships with previous ones. On the contrary, if future queries are totally dissimilar with past queries, our scheme will have little hope of improving and may possibly decrease the retrieval effectiveness of the response.

1.3. Similarity Between Queries

In order to provide an indication of the degree of similarity between requests in three test-collections, we have computed some statistics, depicted in Table 1. This table shows that the CISI collection included more pertinent records per query than the CACM corpus (perhaps "too many relevant documents per query" [Fox 83, p. 7]). The Wall Street Journal collection included in the TIPSTER-DARPA collection reveals a similar pattern.

The second part of Table 1 illustrates the computed similarity between requests according to their relevance assessments. For this computation, we used the Dice's simple coefficient [van Rijsbergen 79, p. 39]. Requests from the CISI test-collection reveal a higher degree of similarity between them than for those of the CACM or in the WSJ. However, the mean similarity between queries is rather low in two cases (CISI: 0.04; CACM: 0.0182) and very low for the subset of the TIPSTER collection (WSJ: 0.00228).

Moreover, the estimated standard error is relatively high indicating that the empirical distribution of the similarity values is mostly in the range 0.0 to 0.1. The second part of Table 1 confirms this fact. For the CISI corpus, 88.4% (526 over a total of 595) of the similarity values are less or equal to 0.1 while for the CACM and WSJ these numbers are 99.45% and 99.5% respectively.

In our evaluations, we have built two sets of relevance links, namely the RF set containing all relevance links and the RF1 set including all relevance links having a relevance value strictly greater than one. For example, in Figure 1, the RF1 set contains only one relevance link (between Document 3 and 7). Table 2 presents the statistics associated with both sets.

From this data, one can see that for the CISI collection, the set RF contains 66,067 links from which 63,889 (around 97%) have a relevance value of one (CACM: 96.7%). The WSJ corpus depicts a more extreme case (1,707,152 over 1,738,429 links in total or 98.2%).

Finally, it is interesting to note that when building a test-collection, we are trying (consciously or not) to write queries for which the relevance judgments are as dissimilar as possible, a phenomena reflected in the above statistics. Such a practice mirrors the designer's wishes that the underlying requests must cover different concepts contained in the corpus. When we designed our additional retrieval mechanism, we formulated a contradictory hypothesis which should hold in commercial retrieval services or, at least, we hope so.

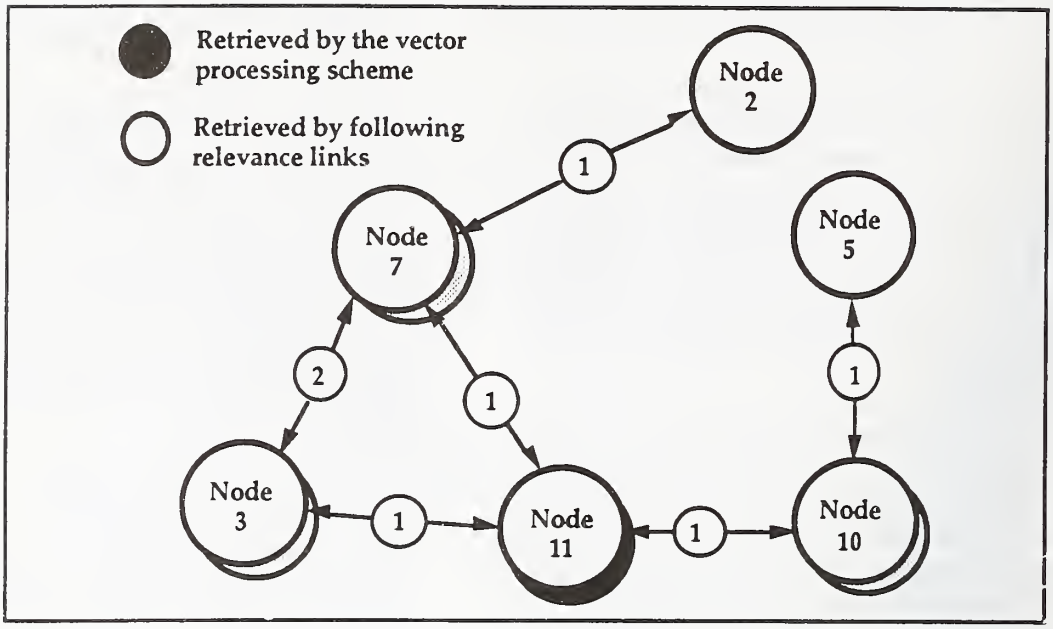


Figure 1: Retrieval of Information Using Relevance Links

Statistics \ Collection	CACM 50 queries	CISI 35 queries	WSJ 150 queries
# documents	3,204	1,460	173,252
# relevant documents	792	1,742	17,069
# distinct relevant documents	554	925	14,280
Mean rel. doc per request	15.84	49.77	113.79
Estimated standard error	12.59	39.96	104.56
Min. # relevant documents	1 (#q: 33)	1 (#q: 6)	2 (#q: 121)
Max. # relevant documents	51 (#q: 25)	144 (#q: 20)	591 (#q: 56)
Similarity between queries (Dice)			
Mean similarity measure	0.0182	0.04	0.00228
Estimated standard error	0.0756	0.0671	0.0151
# 0.00 < SIM ≤ 0.05	1,117	438	11,057
# 0.05 < SIM ≤ 0.1	40	88	66
# 0.10 < SIM ≤ 0.15	27	38	29
# 0.15 < SIM ≤ 0.2	6	17	10
# 0.20 < SIM ≤ 0.25	5	2	6
# 0.25 < SIM ≤ 0.3	3	4	2
# 0.30 < SIM ≤ 0.35	8	2	2
# 0.35 < SIM ≤ 0.4	5	1	0
# 0.40 < SIM ≤ 0.45	2	2	2
# 0.45 < SIM ≤ 0.50	1	0	0
# 0.50 < SIM ≤ 0.55	4	1	1
# 0.55 < SIM ≤ 0.60	4	2	0
# 0.60 < SIM ≤ 0.65	1	0	0
# 0.65 < SIM ≤ 0.70	1	0	0
# 0.70 < SIM ≤ 0.75	0	0	0
# 0.75 < SIM ≤ 0.80	0	0	0
# 0.80 < SIM ≤ 0.85	1	0	0
# 0.85 < SIM ≤ 1.00	0	0	0

Table 1: Relevance Information Characteristics

Statistics \ Collection	CACM 50 queries	CISI 35 queries	WSJ 150 queries
Relevance Link Value RF Set			
# links	8,876	66,067	1,738,429
$\alpha = 1$	8,265	63,889	1,707,152
$\alpha = 2$	495	2,044	30,337
$\alpha = 3$	90	121	907
$\alpha = 4$	23	13	31
$\alpha = 5$	3	0	2
$\alpha = 6$ and more	0	0	0
Mean α value	1.085	1.035	1.019
Estimated standard error	0.343	0.197	0.440
Relevance Link Value RF1 Set			
# links	611	2,178	31,277
$\alpha = 2$	495	2,044	30,377
$\alpha = 3$	90	121	907
$\alpha = 4$	23	13	31
$\alpha = 5$	3	0	2
$\alpha = 6$ and more	0	0	0
Mean α value	2.237	2.067	2.031
Estimated standard error	0.535	0.274	0.180

Table 2: Relevance Links Statistics

1.4. Related Research

In order to include permanently relevance feedback information, various researchers have proposed modification of the document surrogates in a vector-processing scheme. In this vein, Friedman et al. [71] present a framework within which the index term weight w_{ij} assigned to term t_j in document representative d_i can be modified to reflect user's judgments about document content. This proposition is grounded on three principles. The first one specifies that the modification of indexing term weights can occur only for the "good" keywords or for those appearing more frequently in the relevant documents. As a second principle, Friedman et al. [71] suggest deriving the new indexing term weight w'_{ij} in proportion to the existing weight w_{ij}. Thirdly, the authors suggest that the indexing weight modification must be based on the importance of term t_j in: (1) the current request, (2) the set of relevant documents, and (3) the sample of nonrelevant records.

In a related work, Brauen [71] also suggests transforming the document surrogates. In this approach, the system modifies only the document representatives of relevant records (relevance document modification). When implementing such a learning scheme, the system had to consider three cases:

1) a concept t_j is present in the request and absent from the relevant document d_i; thus, the system must add the "synonym" t_j to the corresponding document surrogate;
2) a concept t_j is present in both the request q_k and the relevant document d_i; the indexing term weight w_{ij} must be reinforced;
3) a concept t_j is present only in the relevant document d_i; the indexing term weight w_{ij} must be reduced.

From a different perspective, Gordon [88] suggests a learning scheme based on a genetic algorithm to enhance the retrieval effectiveness. In this approach, each document is described by various surrogates obtained using various binary indexing policies (e.g., based on document abstract, on titles, using full-text or derived from a manual indexing process). The retrieval system considers more than one description for each record and the competition between them will eliminate inappropriate surrogates while retaining more accurate ones. An iterative process affects document surrogates by including or removing index terms based on: (1) the reproduction of descriptions according to their average matching score in which a better representation has a higher chance to survive than others surrogates; and (2) cross-over between pairs of surrogates to generate new descriptors more appropriate to the retrieval of the corresponding document. The retrieval evaluations of the previously described learning schemes are based on

relatively small test-collections: ADI collection (82 documents, 11 queries) for Friedman's experiment, CRANFIELD corpus (424 documents, 155 queries) in Brauen's paper, and Gordon's scheme (18 documents).

Of course, other learning strategies have already been proposed and evaluated, and most of them are directly related to the probabilistic retrieval model [Kwok 90]. Current probabilistic retrieval models [Cooper 92], [Gey 94], [Fuhr 91], [Fuhr 94] consider statistical clues present in the texts of document and queries to infer a probability of relevance. Following [Gey 94], these are hints of the absolute and relative term frequency in the document and in the request, the inverse document frequency and the relative term frequency in the collection. Experimental results have shown attractive retrieval effectiveness. Moreover, Gey [94] has shown that one can compute the value of various parameters according to a given test-collection and report them for other test-collections.

When comparing these probabilistic models with our learning schem_, one can see that they do not operate at the same level of granularity. By analogy with physics, the probabilistic retrieval models lay stress on the components of a document; they operate on an atomic level, whereas our approach, considering words as ambiguous entities, works at a molecular level.

2. Evaluation

To evaluate our proposed strategy and in order to be able to manage a large collection, we have worked with the SMART system [Salton 71]. This vector-processing scheme retrieves, for each request, an ordered list of retrieved records forming the input of our retrieval scheme. To implement our learning model, we have written the needed programs in Smalltalk-80 (an interpreted object-oriented language) and communication between these two systems is achieved by a common file.

2.1. Evaluation of the Vector Space Model

To represent each document and each query by a set of weighted keywords, we have used the SMART indexing system. To select the more appropriate weighting scheme for this operation, we have conducted a set of experiments based on different weighting formulas.

Firstly, to assign an indexing weight w_{ij} reflecting the importance of each single-term t_j, $j = 1, 2, ..., t$, in a document d_i, we may use the following equation:

$$\text{NNN: } w_{ij} = tf_{ij} \tag{2}$$

where tf_{ij} depicts the frequency of the term t_j in the document d_i (or in the request).

To normalize each indexing weight between 0 and 1, we may consider the cosine normalization which is:

$$\text{LNC: } w_{ij} = \frac{\log(tf_{ij})+1}{\sqrt{\sum_{k=1}^{t} [\log(tf_{ik}) + 1]^2}} \tag{3}$$

Finally, we may also take account of the distribution of each indexing term in the collection by giving a higher weight to sparse words and a lower importance to more frequent terms (idf component).

$$\text{LTC: } w_{ij} = \frac{[\log(tf_{ij})+1] \cdot idf_j}{\sqrt{\sum_{k=1}^{t} ([\log(tf_{ik})+1] \cdot idf_k)^2}}$$

$$\text{with } idf_j = \log\left[\frac{n}{df_j}\right] \tag{4}$$

in which n represents the number of documents d_i in the collection, df_j the number of documents in which t_j occurs, and idf_j the inverse document frequency.

The retrieval effectiveness of various combinations of these weighting formulas are reported in Table 3. Since latter evaluation outcomes are computed according to the ten standard recall values, Table 3 depicts results obtained using 10 recall-precision points. Finally, to decide whether one search strategy is better than another, the following rule of thumb is used: a difference of at least 5% in average precision is generally considered significant and, a 10% difference is considered very significant [Sparck Jones 77, p. A25].

For an unknown reason, the best weighting scheme seems to include the idf component only to weight the keywords included in requests and not during the indexing of documents (doc = LNC, query = LTC). The presence of spelling errors can be a partial explanation of such an unexpected result. In the WSJ corpus, low-frequency words are often no longer English. Since the idf scheme assigns extremely high weights to those misspelled terms, the normalization procedure given by Equation 4 also attributes a high value to those terms. The documents containing such terms cannot be retrieved because they will have a relatively small retrieval status value.

In the following results, the weighting scheme "doc = LNC, query = LTC" has been used in the first stage of our retrieval system and forms the baseline of our comparisons. The evaluation under the label "UNINE1" reflects this weighting scheme for queries from #151 to #200.

2.2. Retrospective Evaluation

In order to evaluate a learning strategy, we may provide the learning system with all the available information (all the requests with their relevance assessments in our case). The retrieval effectiveness obtained under such circumstances is called a retrospective test or the apparent performance measure. The resulting average precision represents an upper bound of the performance of the underlying model. From a practical point of view, this measure is computed according to Equation 5,

$$P_{app} = \frac{1}{r} \cdot \sum_{k=1}^{r} AP_k(Q) \qquad (5)$$

in which AP_k denotes the average precision at ten standard recall value for the k^{th} query, considering that the learning scheme is fitted using the entire query sample Q (having a size denoted by r or 150 in this paper).

Such retrospective evaluation returns retrieval effectiveness values that are too optimistic (biased high), reflecting an unrealistic situation. For example, in Table 4a, the learning scheme using all the relevance links (RF set) returns performance results that are *too good to be true.*

	Precision (% change)
Model (# of queries) \ Collection	WSJ
Vector Space Model (150 queries) using 10 recall-precision points	
doc = NNN, query = NNN	8.24
doc = LNC, query = LNC	24.81 (+201.1%)
doc = LTC, query = LNC	26.16 (+217.5%)
doc = LTC, query = LTC	28.93 (+251.1%)
doc = LNC, query = LTC	31.94 (+287.6%)

Table 3: Evaluation of the Vector Processing Scheme Done by SMART

	Precision (% change)
Model \ Collection	WSJ
Vector-processing (r=150 queries) (doc = LNC, query = LTC, 1,000 doc.)	31.9
Full Relevance Feedback	
P_{app} (α = 0.1, m: 10)	73.1 (+129.6%)
P_{app} (α = 0.15, m: 10)	77.8 (+144.4%)
P_{app} (α = 0.2, m: 10)	79.9 (+151.0%)
P_{app} (α = 0.3, m: 10)	81.7 (+156.4%)
P_{app} (α = 0.5, m: 10)	82.9 (+160.1%)
P_{app} (α = 0.9, m: 10)	83.8 (+163.2%)
Full Relevance Feedback (α = 0.9)	
P_{app} (m: 5)	82.8
P_{app} (m: 10)	83.8 (+1.2%)
P_{app} (m: 20)	82.9 (+0.3%)
P_{app} (m: 30)	83.2 (+0.5%)
P_{app} (m: 50)	83.2 (+0.5%)
P_{app} (m: 100)	81.5 (-1.6%)

Table 4a: Evaluation of Vector-Space Model with Full Relevance Feedback (RF Set)

Model \ Collection	Precision (% change) WSJ
Vector-processing (r=150 queries) (doc = LNC, query = LTC, 1,000 doc.)	31.9
Full Relevance Feedback	
P_{app} (α = 0.1, m: 10)	35.8 (+12.4%)
P_{app} (α = 0.15, m: 10)	36.5 (+14.5%)
P_{app} (α = 0.2, m: 10)	36.9 (+15.9%)
P_{app} (α = 0.3, m: 10)	37.4 (+17.5%)
P_{app} (α = 0.5, m: 10)	37.7 (+18.4%)
P_{app} (α = 0.9, m: 10)	37.8 (+18.5%)
Full Relevance Feedback (α = 0.9)	
P_{app} (m: 5)	36.3
P_{app} (m: 10)	37.8 (+4.1%)
P_{app} (m: 20)	38.1 (+5.0%)
P_{app} (m: 30)	38.6 (+6.3%)
P_{app} (m: 50)	38.3 (+5.5%)
P_{app} (m: 100)	37.1 (+2.2%)

Table 4b: Evaluation of Vector-Space Model with Full Relevance Feedback (RF1 Set)

When the apparent performance measure is too optimistic, it is generally an indication that the underlying learning scheme is over-fitted, too narrow for the given data, and cannot forget the details. What we really expect from a learning model is its capability to generalize given information, to retain the main features of the given information and to find useful relationships between data. In our point of view, the learning knowledge derived from RF set is over-fitted and this fact will be confirmed when considering the following subsection. Thus, the performance obtained using the RF1 set seems to depict a more realistic situation (see Table 4b).

2.3. Predictive Evaluation

If the evaluation results under full relevance feedback are usually misleading, more accurate or more "honest" evaluation estimate must be discussed. The basic principle underlying such an evaluation methodology is the following: the performance of a retrieval system must be based on requests other than those given to the learning scheme. Since in each test-collection the number of available queries is relatively small, evaluation must use all the available requests to adjust its parameter settings, on the one hand, and, on the other, all the available queries must be used to measure the performance of the proposed retrieval scheme. This latter fact may contribute to an objective comparison with a system ignoring learning.

To take account of these criteria, the hold-out method suggests splitting the queries sample into two disjoint parts: one subsample will be applied in the learning stage and the other will be used during the evaluation process. This division must be carried out randomly, without looking at the requests themselves. However, not all queries can be exploited both in the learning scheme and during the evaluation.

To overcome this drawback, multiple train-and-test experiments or random subsampling approaches can be considered within which all queries are used for testing, and almost all requests for training [Stone 74]. More precisely, the leaving-one-out approach, a special case of the cross-validation method, represents a solution which works as follows. The query sample Q of size r is divided into r sets. In the k^{th} set, one can find all requests except the k^{th} one. The model is fitted according to r-1 requests and an evaluation measure is computed according to the k^{th} query (not included in the learning sample). The above procedure is repeated for k = 1, 2, ... r and we combine the r prediction values to obtain an average precision measure (see Equation 6),

$$P_{lv} = \frac{1}{r} \cdot \sum_{k=1}^{r} AP_k(Q\text{-}k) \qquad (6)$$

in which AP_k denotes the average precision at ten standard recall values for the k^{th} query under the

condition that the learning scheme is fitted using the query set Q minus this k^{th} request. Such an evaluation strategy results in a real predictive measure because the system does not have any information about the current request during both the learning and the retrieval stages.

Table 5a depicts the retrieval evaluation obtained using the RF set. From these data, one can conclude that taking account of all relevance links does not improve the retrieval effectiveness when the value of the parameter α is less than or equal to 0.1. Setting this parameter to a higher value significantly decreases the retrieval performance. When considering the impact of the parameter m, one can see that the best value seems to be five.

	Precision (% change)
Model \ Collection	WSJ
Vector-processing (r=150 queries) (doc = LNC, query = LTC, 1,000 doc.)	31.9
Leaving-one-out	
P_{lv} (α = 0.1, m: 10)	30.9 (-2.9%)
P_{lv} (α = 0.15, m: 10)	29.9 (-6.3%)
P_{lv} (α = 0.2, m: 10)	28.8 (-9.6%)
P_{lv} (α = 0.3, m: 10)	26.9 (-15.4%)
P_{lv} (α = 0.5, m: 10)	24.5 (-23.1%)
P_{lv} (α = 0.9, m: 10)	21.0 (-34.2%)
Leaving-one-out (α = 0.1)	
P_{lv} (m: 5)	31.6
P_{lv} (m: 10)	30.9 (-2.2%)
P_{lv} (m: 20)	29.2 (-7.6%)
P_{lv} (m: 30)	28.0 (-11.4%)
P_{lv} (m: 50)	25.8 (-18.4%)
P_{lv} (m: 100)	22.4 (-29.1%)

Table 5a: Evaluation of Vector-Space Model Using the Leaving-one-out Method (RF Set)

	Precision (% change)
Model \ Collection	WSJ
Vector-processing (r=150 queries) (doc = LNC, query = LTC, 1,000 doc.)	31.9
Leaving-one-out	
P_{lv} (α = 0.1, m: 10)	31.8 (+0.0%)
P_{lv} (α = 0.15, m: 10)	31.7 (-0.5%)
P_{lv} (α = 0.2, m: 10)	31.6 (-0.8%)
P_{lv} (α = 0.3, m: 10)	31.4 (-1.3%)
P_{lv} (α = 0.5, m: 10)	31.2 (-2.1%)
P_{lv} (α = 0.9, m: 10)	30.7 (-3.5%)
Leaving-one-out (α = 0.1)	
P_{lv} (m: 5)	31.9
P_{lv} (m: 10)	31.8 (-0.3%)
P_{lv} (m: 20)	31.7 (-0.6%)
P_{lv} (m: 30)	31.5 (-1.2%)
P_{lv} (m: 50)	31.3 (-1.9%)
P_{lv} (m: 100)	30.4 (-4.7%)

Table 5b: Evaluation of Vector-Space Model Using the Leaving-one-out Method (RF1 Set)

When only considering relevance links having a relevance value greater than 1 (RF1 set), the retrieval performance cannot increase significantly as shown in Table 5b. When testing the system with various values for the parameters α and m, we cannot find a significant change in the retrieval effectiveness over the baseline ignoring learning. We also have to try to take account for 2,000 retrieved records instead of 1,000, but this alternative does not present any significant change over the results depicted in Table 5b.

From these results, it seems clear that the queries included in the WSJ collection do not have any pertinent relationship between them, or at least, such relationships are not detected by our learning model. This fact confirms our prior feeling as stated in Section 1.3.

The retrieval results submitted to the conference board under the label "UNINE2" are obtained using $\alpha = 0.05$ and $m = 5$ representing a conservative setting. This parameter setting reflects our prior opinion that the relevance judgments of queries #151 to #200 will not have a high degree of similarity with older requests.

2.4. Analysis of Official Results

The official results are based on queries #151 through #200. The results obtained under the label "UNINE1" represents the baseline or the first stage or our retrieval strategy (vector-processing scheme with index term weight = LNC, search term weight = LTC). The performance under the column "UNINE2" is obtained using our additional retrieval scheme with $\alpha = 0.05$ and $m = 5$.

From Table 6, we cannot conclude that our additional retrieval strategy represents significant change over the vector-processing scheme. Our approach retrieves the same relevant records as the vector space model but ranks them in a more suitable sequence, especially for medium or high recall values.

To define the setting for the UNINE2 experiment, we are faced, by analogy, with the following dilemma:

Solution 1: you may win $500.

Solution 2: you obtain a lottery ticket for which the probability of winning $1,000 is 0.5 and the probability of winning $0 is 0.5.

In both approaches, the expected win is the same ($500), however, Solution 2 can be considered risky. Since we have a loathing for risk, we have chosen Solution 1 in our parameters setting.

Conclusion

This paper suggests a learning algorithm based on interdocument relationships established according to relevance assessments obtained from previous requests. The underlying hypothesis of this scheme states that the relevance judgments of future queries will have a high degree of similarity with the relevance assessments of previous requests. To take account of this information, we propose an additional additive scheme within which relevance links are considered to increase the similarity between documents and query, and thus modify the ranking of retrieved documents.

Based on the WSJ collection, a retrospective test shows very attractive retrieval performance but the leaving-one-out method, representing a more realistic predictive measure, does not confirm this previous evaluation. We can conclude that the results of a retrospective test must be interpreted with caution. Since the queries included in a test-collection are written such that they cover different topics contained in the corpus, they do not have a high degree of similarity between them. This fact contradicts the underlying hypothesis of our learning scheme and may be a plausible explanation for the absence of any significant retrieval enhancement. However, even in such circumstances, our retrieval scheme does not significantly decrease retrieval effectiveness over a baseline ignoring learning.

If, traditionally, learning schemes are used mainly with probabilistic retrieval models, our solution is advantageous by being integrated with various Boolean models (p-norm, fuzzy set extension, hybrid Boolean strategies) or with the vector-processing scheme.

In this study, we never take relevance judgments such as relevance feedback into account to reformulate the initial query [Salton 90]. Although we do not reject this attractive proposition, our objective is to evaluate the effectiveness of the initial search; therefore, relevance feedback can be used after this first search to enhance the retrieval effectiveness.

Acknowledgments

This research was supported by the SNFSR (Swiss National Foundation for Scientific Research)

under grant 21-37'345.93 and under grant SNFSR SPP 5NE3-33498. The authors would also like to thank Mr. J. Cavadini, former Minister of Education of the Canton of Neuchâtel, for his support in purchasing the needed hardware without which this experiment could not be possible. The authors also thank M. Choquette from University of Montreal for his help in using the SMART system.

Statistics \ Specification	UNINE1	UNINE2
Retrieved:	50000	50000
Relevant:	3913	3913
Relevant and retrieved:	3191	3191
Interpolated Recall - Precision		
at 0.00	80.25	81.07 (+1.0)
at 0.10	62.97	62.72 (-0.4)
at 0.20	54.23	53.89 (-0.6)
at 0.30	43.92	44.28 (+0.8)
at 0.40	36.81	37.99 (+3.2)
at 0.50	30.52	32.51 (+6.5)
at 0.60	25.13	26.88 (+6.9)
at 0.70	19.14	20.82 (+8.8)
at 0.80	13.40	15.20 (+13.4)
at 0.90	7.05	8.45 (+19.9)
at 1.00	1.23	1.29 (+4.9)
non-interpolated		
average precision	31.90	32.79 (+2.8)
Precision:		
at 5 docs:	52.0	52.0 (0.0)
at 10 docs:	47.4	49.0 (+3.4)
at 15 docs:	45.6	45.87 (+0.6)
at 20 docs:	43.2	43.6 (+0.9)
at 30 docs:	40.4	40.67 (+0.7)
at 100 docs:	28.06	28.36 (+1.1)
at 200 docs:	20.43	20.63 (+1.0)
at 500 docs:	11.1	11.16 (+0.5)
at 1000 docs:	6.38	6.38 (0.0)
R-Precision		
(precision after R docs ret), Exact:	34.42	35.02 (+1.7)

Table 6: Official Evaluation of Vector-Space Model (UNINE1) vs. Including Relevance Links (UNINE2)

References

[Blair 90] D. C. Blair: Language and Representation in Information Retrieval. Elsevier, Amsterdam (Holland), 1990.

[Brauen 71] T. Brauen: Document Vector Modifications. in The SMART Retrieval System - Experiments in Automatic Document Processing, G. Salton (Ed.), Prentice-Hall Inc., Englewood Cliffs, NJ, 1971, 456-484.

[Bush 45] V. Bush: As we may Think. Atlantic Monthly, 176(1), 1945, 101-108.

[Cooper 92] W. Cooper, F. C. Grey, D. P. Gabney: Probabilistic Retrieval Based on Staged Logistic Regression Proceedings ACM-SIGIR'92, Copenhague (DK), June 1992, 198-210.

[Fox 83] E. A. Fox: Characterization of Two Experimental Collections in Computer and Information Science Containing Textual and Bibliographic Concepts. Cornell University, Department of Computer Science, Technical Report TR 83-561, September 1983.

[Friedman 71] S. R. Friedman, J. A. Maceyak, S. F. Weiss: A Relevance Feedback System Based on Document Transformation. in The SMART Retrieval System - Experiments in Automatic Document Processing, G. Salton (Ed.), Prentice-Hall Inc., Englewood Cliffs, NJ, 1971, 447-455.

[Fuhr 91] N. Fuhr, C. Buckley: A Probabilistic Learning Approach for Document Indexing. ACM Transactions on Information Systems, 9(3), 1991, 223-248.

[Fuhr 94] N. Fuhr, U. Pfeifer: Probabilistic Information Retrieval as a Combination of Abstraction, Inductive Learning, and Probabilistic Assumptions. ACM Transactions on Information Systems, 12(1), 1994, 92-115.

[Furnas 87] G. Furnas, T. K. Landauer, L. M. Gomez, S. T. Dumais: The Vocabulary Problem in Human-System Communication. Communications of the ACM, 30(11), 1987, 964-971.

[Gey 94] F. C. Grey: Inferring Probability of Relevance Using the Method of Logistic Regression Proceedings ACM-SIGIR'94, Dublin (IR), July 1994, 222-231.

[Gordon 88] M. Gordon: Probabilistic and Genetic Algorithms for Document Retrieval. Communications of the ACM, 31(10), 1988, 1208-1218.

[Krovetz 92] R. Krovetz, W. B. Croft: Lexical Ambiguity and Information Retrieval. ACM-Transactions on Information Systems, 10(2), 1992, 115-141.

[Kwok 88] K. L. Kwok: On the Use of Bibliographically Related Titles for the Enhancement of Document Representations. Information Processing & Management, 24(2), 1988, 123-131.

[Kwok 90] K. L. Kwok: Experiments with a Component Theory of Probabilistic Information Retrieval Based on Single Terms as Document Components. ACM Transactions on Information Systems, 8(4), 1990, 363-386.

[van Rijsbergen 79] C. J. van Rijsbergen: Information Retrieval. Butterworths, 2nd edition, London (UK), 1979.

[Salton 71] G. Salton (Ed.): The SMART Retrieval System - Experiments in Automatic Document Processing. Prentice-Hall Inc., Englewood Cliffs (New Jersey), 1971.

[Salton 90] G. Salton, C. Buckley: Improving Retrieval Performance by Relevance Feedback. Journal of the American Society for Information Science, 41(4), 1990, 288-297.

[Savoy 94a] J. Savoy: Ranking Schemes in Hybrid Boolean Systems: A New Approach. ACM Transactions on Information Systems, 1994, accepted with revisions.

[Savoy 94b] J. Savoy: A Learning Scheme for Information Retrieval in Hypertext. Information Processing & Management, 30(4), 1994, 515-533.

[Sparck Jones 77] K. Sparck Jones, R. G. Bates: Research on Automatic Indexing 1974-1976. Technical Report, Computer Laboratory, University of Cambridge (England), 1977.

[Stone 74] M. Stone: Cross-Validatory Choice and Assessment of Statistical Predictions. Journal of the Royal Statistical Society, Series B, 36(2), 1974, 111-147.

[Turtle 91] H. Turtle, W. B. Croft: Evaluation of an Inference Network-Based Retrieval Model. ACM Transactions on Information Systems, 9(3), 1991, 187-222.

Using Database Schemas to Detect Relevant Information

James Driscoll, Gary Theis, Gene Billings

Department of Computer Science
University of Central Florida
Orlando, Florida 32816
phone: (407) 823-2341
fax: (407) 823-5419
e-mail: driscoll@cs.ucf.edu

Abstract

The Entity-Relationship (ER) Model is used as a basis for descriptions of information preferences (profiles) in the information filtering process. We view a profile as having both a static aspect and a dynamic aspect. It is shown that the static aspect of a profile can be represented as an ER schema; and the dynamic aspect of the profile can be represented by synonyms of schema components and domain values for schema attributes. The TREC-3 filtering task is accomplished using this technique to generate a "custom" filter for each routing topic. These are true filters in the sense that each one moves across the entire document collection to produce the TREC ranked list results. Our network environment to allow any number of filters to run simultaneously is also described.

Introduction - The Filtering Task and Profiles

The filtering process is based on descriptions of individual (or group) information preferences, often called profiles. Profiles typically represent long-term interests. Information filtering is concerned with repeated uses of the profiles, and the profile is assumed to be a correct specification of an information interest [1].

For our participation in TREC-3, we focus on the filtering task and view a profile as having both a static aspect and a dynamic aspect. We present a procedure for representing the user need statement of a TREC topic as a database Entity-Relationship (ER) schema. An ER schema becomes the static aspect of a profile. For a schema, a synonym list is created for each of the schema components, and a domain value list is created for each of the schema attributes. These lists become the major part of the dynamic aspect of a profile. Our filtering procedure uses the dynamic aspect of a profile to detect relevant documents.

For the TREC filtering task, there are fifty topics or descriptions of information to be considered, and these must be transformed into filter profiles. Each TREC topic is in the form of a highly-formatted, natural language, user need statement. Refer to Figure 1 for an example. This is TREC Topic 122 which concerns new cancer fighting drugs.

<top>

<head> Tipster Topic Description

<num> Number: 122

<dom> Domain: Medical & Biological

<title> Topic: RDT&E of New Cancer Fighting Drugs

<desc> Description:

Document will report on the research, development, testing, and evaluation (RDT&E) of a new anti-cancer drug developed anywhere in the world.

<narr> Narrative:

A relevant document will report on any phase in the worldwide process of bringing new cancer fighting drugs to market, from conceptualization to government marketing approval. The laboratory or company responsible for the drug project, the specific type of cancer(s) which the drug is designed to counter, and the chemical/medical properties of the drug must be identified.

<con> Concept(s):

1. cancer, leukemia

2. drug, chemotherapy

<fac> Factor(s):

<def> Definition(s):

</top>

Figure 1. TREC-3 Routing Topic 122.

Semantic Modeling

From a database point of view, it is interesting to note that each TREC topic represents the data requirements analysis of a small enterprise (real-world situation). Semantic modeling can be used to capture such an analysis. The Entity-Relationship model is the best-known semantic model [2,3]. Briefly, the ER model includes the following semantic concepts:

Entity Set:

> This is a collection of objects which have common attributes. Each attribute is associated with a domain of possible values. Objects can have a physical existence (such as a person) or a conceptual existence (such as a company, or laboratory). Some attributes can be used to identify an object in an entity set (such as Social Security Number for a person). Some entity sets may be weak because objects in the entity set are identified by being related to specific objects from another entity set.

Relationship:

> This is a set of associations among objects in one entity set and objects in other entity sets. For example, between the entity set of drugs and the entity set of cancers, there can be a relationship representing which drug counters which cancer. Each entity set that participates in a relationship plays a particular role in the relationship. Relationships can also have attributes.

Specialization, Generalization, Categorization:

> These concepts describe the superclass/subclass relationships that can exist among entity sets. Subclasses can inherit attributes, predicates can be used to define subclasses, multiple subclasses can be disjoint or overlapping, and the union of classes can be formed.

An ER diagram or schema is a technique for representing the logical structure of a database in a pictorial manner. As such, it provides a simple and readily understood means of communicating the salient features of the design of any given database [2]. The popularity of the ER model as an approach to database design can be attributed to the ER diagramming technique. The major diagramming rules follow [3]:

> Each entity set is shown as a rectangle.
>
> Each attribute is shown with an ellipse.
>
> Each relationship is shown as a diamond with lines to the participating entity sets, and roles may be identified by labeling the lines.
>
> A weak entity set and its identifying relationship are distinguished by using double lines for the rectangle and the relationship.
>
> A subclass relationship is indicated with a line and a subset symbol and possibly a predicate.
>
> Overlapping subclasses are indicated with a circled "o".
>
> Disjoint subclasses are indicated with a circled "d".
>
> The union of classes is indicated by a circled "u".

There are other semantic concepts that can be expressed using an ER schema, but the above concepts are enough to demonstrate our idea for creating and using profiles for information filtering.

Example

Consider, again, TREC Topic 122 shown in Figure 1. An ER Model schema for this topic is shown in Figure 2. Like the user need statement in Figure 1, the schema specifies the information that must be detected within a section of text to decide whether or not the text is relevant to Topic 122.

ER schemas can be created following some simple rules if one has a narrative description of the database requirements. The nouns appearing in the narrative will tend to give rise to entity sets, verbs will tend to indicate relationships, adjectives will generally indicate predicates, additional nouns that modify other nouns tend to be entity attributes, and so on.

By comparing the text of Topic 122 in Figure 1 to the ER schema in Figure 2, one can see that the schema reflects the sentences read in the topic. The ER diagram is also "atomic" in the sense that every component is labeled with a single word found in the topic. For example, the phrase "drug project" in the topic became the entity set "project" with a specialized entity for "drug project", where the adjective "drug" became a predicate for the superclass/subclass relationship.

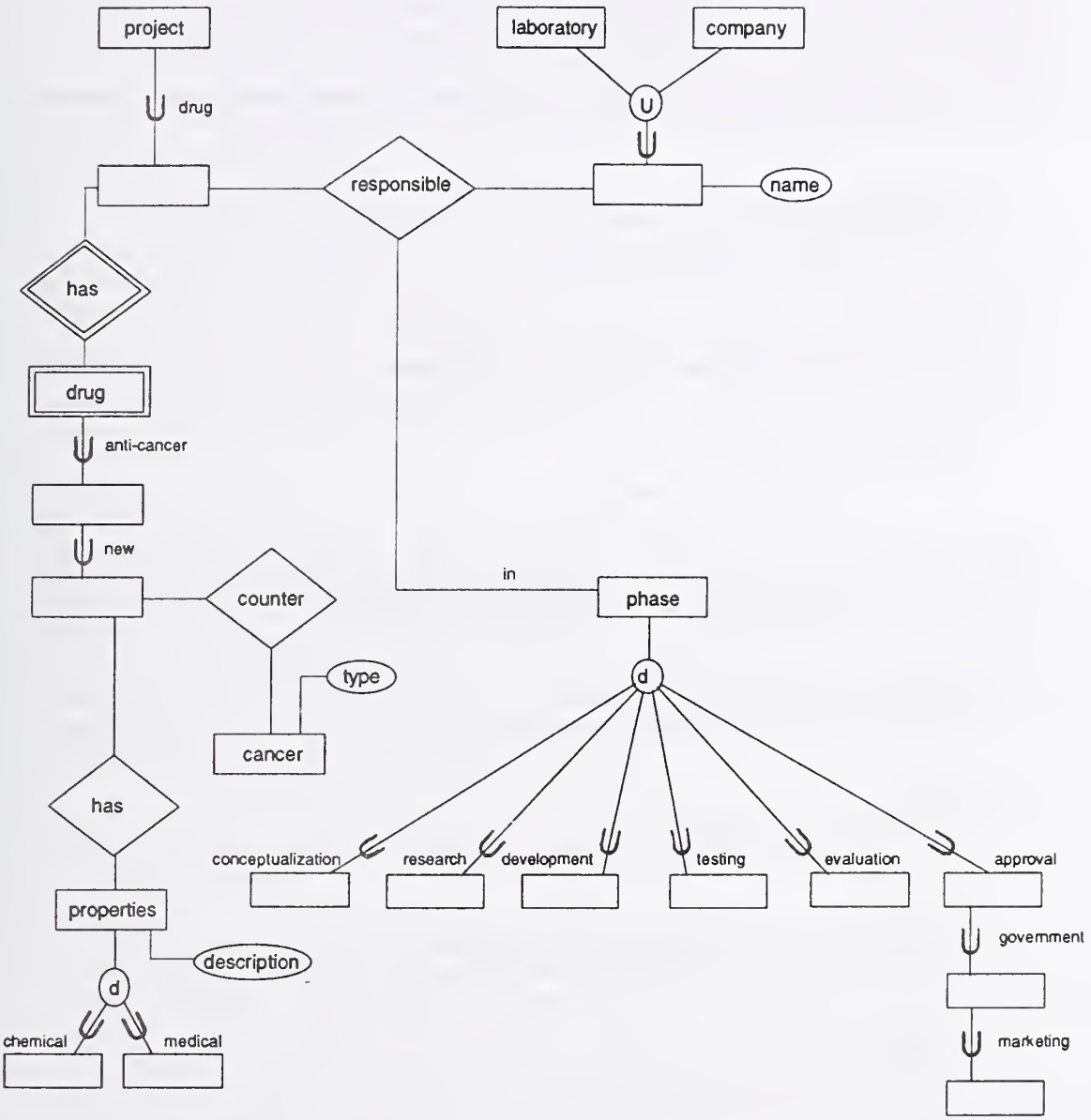


Figure 2. ER Schema for TREC-3 Routing Topic 122.

Procedure for Making a Profile

We have established a filtering system based on profiles represented by ER schemas. In our system, a profile has a static and dynamic aspect. The static aspect of a profile is an atomic ER schema. The dynamic aspect of a profile consists of a series of lists (files) based on the ER schema. A list is either a synonym list or a domain list. A synonym list is created for each labeled component of the ER diagram that is significant. Sometimes, several synonym lists can be merged into one list. A domain list is created for each attribute in the ER diagram. Sometimes, the values that should be in a domain list are not known, or only partially known.

Considering the ER schema in Figure 2, the following lists could be created as significant lists:

List 1
A synonym list for the word "counter":
counter, cure, block, control, . . .

List 2
A synonym list for the words "company" or "laboratory":
company, laboratory, Inc., Co., Incorporated, . . .

List 3
A synonym list for the word "drug":
drug, medicine, medication, . . .

List 4
A synonym list for the words "conceptualization", . . . , "approval":
conceptualization, . . . , approval, study, experiment, . . .

List 5
A synonym list for the word "properties":
properties, attributes, characteristics, . . .

List 6
A synonym list for the word "cancer":
cancer, cancerous, carcinogen, carcinoma, . . .

List 7
A domain list for names of companies or laboratories:
SQUIBB, ROCHE, <others are not known, for now>.

List 8
A domain list for types of cancer:
kidney, lung, skin, ovarian, . . .

List 9
A domain list for descriptions of properties:
<unknown, for now>.

The next step in completing the profile is to indicate the possibilities for finding information which satisfy the schema. This is part of the dynamic aspect of a profile. As an example, a simple technique (which we used in the TREC-3 routing experiments) is to weight each of the lists, use a window of text, and then count "hits" in the window for combinations of the lists. We consider the window size, the file combinations, and list weights as a part of the dynamic aspect of a profile called the insertion criteria for an ER schema.

For example, one variation of the lists could be to consider hits in List 1, List 2, List 3, List 4, List 5, List 6, and List 8 while ignoring List 7 (since only two values are known) and ignoring List 9 (since no values are known). There are several variations which could be used to indicate the presence of relevant information in this example. For TREC experiments, we ranked relevant documents according to a value determined by summing the "hits" in weighted variations of the above lists.

To summarize, we consider the static aspect of a profile to be the ER schema, and we consider the dynamic aspect of a profile to be the values placed in each of the domain and synonym files, the window size, and the various file combinations and file weights. At the present time we manually create filter profiles. We create an ER schema after reading a TREC topic. Next, we determine the significant domain and synonym lists from the ER schema. Then we initially populate the lists using thesauri, dictionaries, and whatever other reference sources we can find; sometimes a list remains empty. Finally, we create an information (INF) file to specify the window size, and the various file combinations and file weights.

Details

To establish a filter for a TREC topic, and do TREC filtering experiments, we have a standard text scanning program which inputs the window size, the domain and synonym files, and one or more variations (which also indicate weights) of the lists. The scanning program then moves across TREC document collections, producing a ranked list of relevant documents. We have used the TREC training data to modify the dynamic aspect of a profile. This is accomplished by using viewed relevant and non-relevant documents to adjust the window size and make additions and deletions to the domain and synonym lists. We have developed a few utility programs to help us do this quickly.

In its current implementation, the scanner requires a stream of text delimited by standard SGML. The only specific markers actually used are the <DOC>, </DOC>, <DOCNO>, and </DOCNO> markers.

Figure 3 provides an example of an INF file for the Topic 122 ER schema displayed in Figure 2. The INF file indicates the topic number, the size of the text window to be used, the number of synonym and domain files, an output filename, the actual file names of the synonym and domain files, a minimum document relevancy value to consider valid, and the number of combinations followed by the weighted combinations that result in an acceptable insertion in the ER schema. In Figure 3, there are two combinations. The first combination is the one mentioned in the previous section; it is appropriate in the case that few or no domain values are known for company/lab names and descriptions of properties. The second combination is one that could result after filter training, when company/lab names have been identified and descriptions of properties can be specified. Since the topic statement in Figure 1 specifies three items of information that must be present in a relevant document, there are three attributes within

the ER schema for the topic and three domain files specified in the INF file. These three items are each given a high weight in the second combination.

Examples of the synonym and domain files for the INF file in Figure 3 appear as List 1 through List 9 in the previous section.

A central part of the scanner is the "text window". This structure is essentially an array which contains the current group of words being evaluated for local purposes. At any given point in processing, the text window contains the last X words read from the text, where X is specified as the window size in the INF file. This provides for a variety of local evaluation sizes, with an appropriate size selected based on the schema being checked (e.g., searching on a paragraph-by-paragraph size of roughly 100 words, as opposed to a sentence-by-sentence search of 20 words at a time). The text window's usage gives rise to the terms local and global. Local refers to an evaluation done exclusively on the text within the window, and global refers to an evaluation on the entire text of a document.

Documents are evaluated in a single-pass through the text. Document text begins immediately after the document ID, and ends at the document end marker. As each new word is scanned into the text window, it is compared to the entries in the files. This is accomplished by having read all the file entries into a memory resident hash table prior to the scanning process. If a match or matches are found, they are tallied in an array which contains the number of matches currently within the window, by file. At the same time, match counts that are passing out the end of the text window are subtracted from the array.

When the current word registers a match, there is an immediate evaluation of the current window's contents. For each valid combination specified in the INF file, there is determined a combination value. The combination value is the sum of the quotients of the number of non-zero, required matches (zero matches or non-required files add zero to the sum) for each file multiplied by 1 minus the result of 1 divided by the sum of the "parts" specified in the INF file for each particular file in the combination being evaluated plus the total number of files.

122	Refers to Topic 122.
155	Indicates a 155 word window.
9	There are nine synonym and domain files.
t122.out	Output file for ranked documents.
counter.syn	Synonym file.
COorLAB.syn	Synonym file.
drug.syn	Synonym file.
con-app.syn	Synonym file.
properties.syn	Synonym file.
cancer.syn	Synonym file.
COorLAB_name.dom	Domain file.
cancer_type.dom	Domain file.
properties_desc.dom	Domain file.
0.00	Minimum allowable document weight to output.
2	There are two combinations.
1 1 1 1 1 1 0 1 0	One part for every "established" file.
1 1 1 1 1 1 6 6 6	Emphasis on relevancy requirement.

Figure 3. An INF file for the ER Schema in Figure 2.

Only the highest combination value encountered is retained, such that at the end of the document, there will be a set of combination values which are maximums for the entire document. Concurrent with these local evaluations, a global document match array is maintained for combined local and global weighting at the end of the document.

Once the entire text of a document has been scanned, a local weight is determined by summing the squares of the best combination weights achieved within the document. Following this, a series of combination values are calculated, then summed, and squared to arrive at a global weight. This operation is identical to the combination calculations for local weight except that the global match count array is used instead of the array for the window. The final weight of the document is then reported as 75% of the local weight added to 25% of the global weight.

Network Environment

Figure 4 diagrams the network that enabled twenty undergraduate students in a Computer Science database class to develop ER schema synonym and domain files, and establish filters for each of the TREC-3 routing topics.

For training filters:

1. The Vol. 1 CD was copied to the hard drive of a PC running Linux (a public domain version of Unix) and functioning as an NFS node on the network.

2. The Vol. 2 CD was copied to the hard drive of a SPARCserver 690MP (four processors) on the network.

3. Students ran filters and viewed training text from 32 RISC 6000 machines across the network.

For the UCF101 and UCFSJM runs:

1. The Vol. 3 CD was copied to the hard drive of the SPARCserver 690MP (four processors) on the network.

2. Most filters were run on the SPARCserver 690MP (a few were run on the RISC 6000 machines).

Each of the 32 RISC 6000 machines had 8 MB of RAM. The NFS node had 16 MB of RAM, and the SPARCserver 690MP had 128 MB RAM. All these machines (except for the NFS node) were shared with normal University and Department computing.

During training, students began to partition the Vol. 1 and Vol. 2 document collections and submit each of their filters in parallel by remote login to the 32 RISC 6000 machines. This caused network problems between the RISC machines and the SPARCserver. The NFS node connection never failed! For the UCF101 and UCFSJM runs, we never exceeded eight filters running simultaneously.

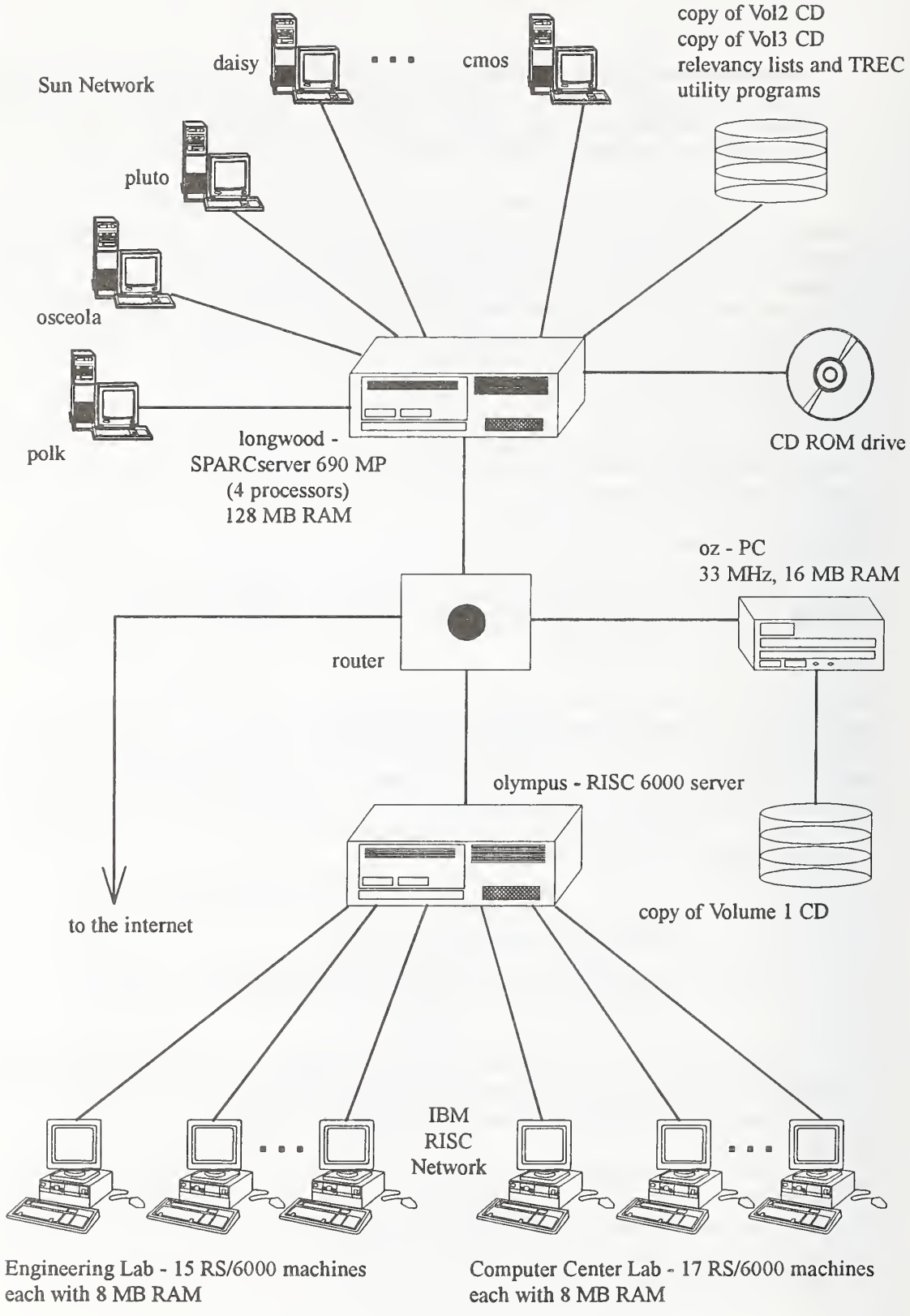


Figure 4. The network environment.

Performance Results and Analysis

One of our experiments was a Category B filtering task (UCFSJM) using a document collection representing a single domain (newspaper documents). There were three participants in this task. Four of the topics had no relevant documents. Our filtering approach had the best average precision for 28 topics, the median average precision for 16 topics, and the worst average precision for two topics. Our overall average precision was .3326 for this experiment.

Another experiment was the regular filtering task (UCF101) using approximately one gigabyte of text representing several domains. Our filtering approach had the best average precision for 6 topics. For 26 of the topics, our average precision was above the median. For 24 of the topics, our average precision was below the median. Our overall average precision was .3278 for this experiment.

Acknowledgments

All the students taking our undergraduate database course (COP 4710, Databases) during the Spring Semester 1994 and the Summer Semester 1994 contributed to the success of UCF's participation in TREC-3. Several undergraduate students did more than expected. They are Sara Abbott, Jackie Truong, and Cynthia Eckman. Each one of these students submitted a filter result for the UCFSJM run. Another undergraduate student, Kai-Lin Hu, deserves recognition for her help creating Entity-Relationship diagrams for all fifty TREC-3 routing topics and helping with documentation for the project. Cathy McGrew (graduate student) coordinated and did most of the preparation for the undergraduate students' experiments. The network environment for our TREC-3 participation was possible because Don Harper (Computer Science) and Adel Chehab (Computer Services) were so willing to establish the storage areas and coordinate the data access among the Computer Science Sun network, Computer Services' IBM RISC network, and the NFS node.

References

[1] N. J. Belkin and W. Bruce Croft, "Information Filtering and Information Retrieval: Two Sides to the Same Coin?", *Communications of the ACM*, Vol. 35, No. 12 pp. 29-38, December 1992.

[2] C. J. Date, *An Introduction to Database Systems (Sixth Edition)*, Addison-Wesley Publishing Company, Inc., 1995.

[3] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems (Second Edition)*, The Benjamin/Cummings Publishing Company, Inc., 1994.

[4] M. T. Ozsu and P. Valduriez, *Principles of Distributed Database Systems*, Prentice Hall, 1991.

A Statistical Analysis of the TREC-3 Data

By Jean Tague-Sutcliffe and James Blustein
University of Western Ontario
London, Ontario N6G 1H1, Canada
tague@julian.uwo.ca

Abstract

A statistical analysis of the TREC-3 data shows that performance differences across queries is greater than performance differences across participant runs. Generally, groups of runs which do not differ significantly at large, sometimes accounting for over half the runs. Correlation among the various performance measures is high.

1. Introduction

Although the purpose of the TREC trials is primarily to learn from one another what works and what does not work in information retrieval, rather than picking winners and losers, there is a need to determine which runs produce results which are significantly different from the results of other runs. By significantly different we mean that, by standard statistical tests, the differences among the performance scores for the various runs, averaged over queries, appear to be greater than what might be expected by chance. Only by looking at statistically significant differences can we generalize the TREC results to other queries and databases.

The question of chance arises because the set of fifty queries actually processed in the TREC-3 trials is really a random sample from the population of all possible queries which could be asked of the database. We assume our results hold not just for the particular set of queries we used in TREC-3, but for any similar set of queries. The function of statistical testing is to determine which differences among run means appear to be real and which differences appear to be the result of sampling variation. These conclusions can be drawn only with a predetermined error probability of saying there is a difference in runs when there is not, the alpha error probability, usually set at .05. At the same time, there is also an undetermined beta error probability of saying there is no difference when there actually is. In choosing a statistical test, one attempts to minimize beta for the preset alpha value.

In this paper, we will look at the variables which have been used to summarize the output from each TREC-3 run and at the results of statistical tests primarily using the analysis of variance (ANOVA) followed by a posteriori tests of individual differences between the means of pairs of runs. The ANOVA technique makes a number of assumptions about the data, but when it may be used it is to be preferred to the nonparametric approach, called the

Friedman test, which makes no assumptions beyond a level of measurement at least ordinal. The reason for this preference is that nonparametric tests, in general, have a higher beta error probability than the corresponding parametric tests. However, we will also look at two other approaches, for comparison with the primary one: ANOVA applied to an arcsine transformation of the original data and the nonparametric Friedman test. However, the nonparametric test is based on a rank-transformation of the data, so that a certain amount of information about differences in performance is being ignored. The comparative ordering of the runs will be by average rank, rather than by the original scores (such as average precision) and so the ordering may change.

All of these approaches control the alpha error probability at .05 both for the initial test, whether or not there is overall a significant difference among a set of treatments (in our case runs), and for the set of a posteriori tests which determine which pairs of means are significant different. As Berenson, Levine, and Goldstein (1983) say relative to an experiment in which c treatments (e.g., runs) are being compared and where H_0, the null hypothesis, is that there is no difference between means:

> In an effort to determine which of the c means are significantly different from the others, it is improper for the researcher to use all possible two-sample t tests to examine all pairwise comparisons between the means; all such comparisons would not be independent and, if c was large enough, it is likely that the difference between the largest and smallest of the <means> would be declared significant even if the null hypothesis were true. That is, the greater the number of groups (i.e., levels of a factor) c, the greater the number of pairwise comparisons [i.e., c(c-1)/2] between means, and the more likely it would become to erroneously reject one or more of them--even if H_0 were true. Thus, if several pairwise comparisons were made, each at the α level, the probability of incorrectly rejecting H_0 at least once would increase with c and would exceed α.
> (page 86-87)

In fact, in the case of the TREC-3 Ad Hoc data, where there are 42 runs, there are 42(41)/2=861 possible pairwise comparisons and so, if each of these were tested at the α=.05 level, the probability of incorrectly rejecting H_0 at least once would be $1 - (.95)^{861}$, i.e., almost a certainty.

As Berenson et al note, several a posteriori multiple comparison procedures have been devised for investigating significant differences following a significant ANOVA. The one which we use is the Scheffé test, which determines a minimum significant difference, based on the number of means being compared and alpha, such that any pair of means differ significant if their difference exceeds this value. Generally speaking, this minimum significant difference will increase with the number of means being compared, since it is, for example, much more likely we will get a large difference by chance when we are looking at 861 differences, rather than a single difference.

2. Performance Measures

There are a number of ways of describing the effectiveness of each TREC participant strategy or run for each query. The query run performance measures used in the TREC-3 analysis carried out at NIST are the following:

- Average Precision, defined as the average of the precision values at the points relevant documents were retrieved in the run;
- R Precision, defined as the precision after R documents are retrieved in the run, where R is the number of relevant documents for the query;
- Precision at Standard Recall Levels, where the levels are
  0, 0.1, 0.2. 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 1.
- Precision at Standard Numbers of Documents Retrieved, where the numbers of documents are 5, 10, 15, 20, 30, 100, 200, 500, and 1000.

In addition, we examined the following:

- Precision averaged over the 11 Standard Recall Levels
- Precision averaged over the 9 Number of Document Levels.

For each of these variables, the following procedures were carried out:

- Determination of the means and variances over queries for all runs,
- Hartley test to determine if the ANOVA assumptions are satisfied,
- Arcsine transformation of variable if the ANOVA assumptions are not satisfied,
- Rank transformation of variable if the ANOVA assumptions are not satisfied,
- Analysis of Variance (ANOVA) on scores and transformed scores if necessary to determine if there is an over-all difference in the means for the runs,
- Scheffé tests to determine which pairs of run means differ significantly and to group runs for which there is no significant difference in means.
- Friedman nonparametric test on ranks, as described in Conover (1980), to assess which pairs of run means differ significantly if ANOVA assumptions not satisfied.

3. The Analysis of Variance

The assumptions of ANOVAapplied to the TREC-3 data, are as follows:

- the effectiveness scores represent a random sample, i.e., are independent of one another;
- the effectiveness scores are approximately normally distributed
- the variance of the effectiveness scores is approximately the same for all runs

ANOVA is robust (i.e., still valid) under moderate departures from the last two assumptions. If the last two assumptions are not satisfied for data which, essentially, is a proportion or percentage, the usual procedure is to apply transformation consisting of taking the arcsine of the square root of the original scores (the arcsine transformation). ANOVA is then applied to the transformed scores. Alternatively, one can carry out a nonparametric Friedman test, which makes no assumptions about the variables, but which replaces the original scores by their ranks.

The ANOVA model is a repeated measures design, where the runs were performed on the same set of queries; its mathematical form is:

$$Y_{ij}=\mu+\alpha_i+\beta_j+\epsilon_{ij}$$

where Y_{ij} is the score for the ith participant on the jth query

μ is the overall mean score

α_i is the effect of the ith run

β_j is the effect of the jth query

ϵ_{ij} is the random variation about the mean

The function of the analysis of variance is to determine if the run effects (the α_i) are different from zero. The logic of the procedure is that, if the means show no more variability than what would be expected if they were the means of random samples from the same population of scores, then the run effects are zero (the null hypothesis H_0 is true). One can also test whether or not the variability of the query means is greater than would be expected by chance (i.e., whether or not the $\beta j = 0$).

In the Scheffé test a minimum significant difference is determined based on the underlying random variation and the number of runs. If two participant means do not differ beyond this minimum significant difference, they are assigned to the same group, indicated in the tables by the same alphabetic symbol.

4. Results

The Hartley test showed some evidence that the original scores in the Ad Hoc data set did not satisfy the equality of variance assumption of ANOVA. For this reason, an arcsine transformation was applied to stabilize the variances and the rank-based Friedman test was carried out in addition to ANOVA for this data. However, the resulting groupings showed very few differences from the nontransformed data. Analysis is given for nontransformed scores, and where there is a difference with the arcsine-transformed data in the top and the bottom group in the Scheffé test it is noted. The rank-based analysis is presented in a separate table, when carried out, since it produces a different ranking of the runs.

The analysis of variance table and the Scheffé groups for the variable Average Precision are shown in Tables 1 and 3. The variability attributable to the various effects (runs, queries, error) is shown in the fourth column of Table 1, labelled 'Mean Square'. The F values indicate that the runs and queries effects are significantly different from zero at both the α=.05 and the α=.01 significance levels. It can be seen, from Table 1, that the variance due to queries is much greater than that due than that due to runs. Thus, it appears that runs are preforming differentially over the queries, so that for some queries some approaches are best and for other queries other approaches are best.

Source of Variation	DF	Sum of Squares	Mean Square	F Value
Runs	41	15.42	0.94	34.44**
Query	49	46.25	0.94	86.46**
Error	2009	21.93	0.01	
Total	2099	83.60		

**Probability of F < .0001.

Table 1 -- Analysis of Variance of Average Precision, Ad Hoc Data

In Table 3, we see that the top group, represented by the letter A in the Scheffé groupings, consists of the top-ranking 20 runs of the 42 runs and that the corresponding range of mean average precision values which do not differ significantly from one another varies from 0.269 to 0.423. The B group includes the 21st run and all those runs which do not vary significantly from it, namely the runs from rank 2 to rank 24. The C group includes the 25th ranking run and all those runs which do not vary significantly from it. . Other groups are formed in a similar fashion. There is a great deal of overlap among the groups, but one can see that several sets of three groups, for example, groups A, F, and M, will 'cover' the set of runs.

Using the arcsine transformation produces a marginal change in the groupings: two runs added to the top group and three runs removed from the bottom group. More changes can be observed from the rank-transformation based results in Table 4. The A group now contains 18, rather than 20, runs, a not surprising result since ranks have now been substituted for precision scores. Note, also, some variation in the ordering, as a result of the fact that we are averaging ranks, rather than original scores.

The wide range of mean average precision values which do not differ significantly and the small number of differing groups is surprising. It can be attributed to the effect we noted earlier, namely that there is a great deal more variability resulting from the queries than from the runs, so that runs perform very differently with different queries. Rankings of runs are not very stable from one query to another.

Using a different performance measure does not seem to change this pattern very much. Similar findings resulted from the ANOVA and the Scheffé test for the other variables. Tables 5 and 6 show, for example, the Scheffé groupings obtained with the variables R-precision and Precision at 100 documents retrieved, respectively, for the Ad Hoc results. These variables are even less discriminating, with the top-ranked 28 runs and 31 runs, respectively, in the A group.

The ANOVA and Scheffé tests for the Routing data show a similar pattern (Tables 2 and 7). The variance resulting from queries is over five times that resulting from the runs. Of the 34 runs, 23 lie in the top A group of runs which do not differ significantly. Also, as with the Ad Hoc data, very little difference in ranking resulted from using the other variables.

Is there any value in using all the variables described in the 'Variables' section above? The results from this analysis indicate the answer to this question is 'no'. The rankings of runs obtained using different variables are very similar. The correlations between seven of the variables is shown in Table 8: average precision, R-precision, precision at 30, 10, and 200 documents retrieved and interpolated precision at .5 and .9 recall All correlations are above the .9 value except for those with precision at .9 recall. The reason for this anomoly is that

interpolated values for high recall levels are not very reliable, as, for those runs which did not achieve a total recall, the precision for a recall of one was set to zero.

Source of Variation	DF	Sum of Squares	Mean Square	F Value
Runs	33	6.18	0.19	21.65**
Query	49	49.54	1.01	118.10*
Error	1617	13.84	0.01	
Total	1699	69.56		

**Probability of F < .0001.

Table 2 -- Analysis of Variance of Average Precision, Routing Data.

Another question one might ask of the multiple effectiveness measures is: which one appears to be the most discriminating in terms of showing significant differences among the runs. Table 9 was compiled to answer this question. It shows, for each of the measures, the number and percentage of the runs in the top group (A group) for both the ad hoc and the routing data. Two additional variables were were added to those which have been heretofore calculated from TREC tests: precision averaged over all nine levels of numbers of retrieved documents and precision averaged over all eleven levels of recall.

These results indicate that precision at very high low and very high values of the number of documents retrieved (n) and the recall level (r) are not very discriminating, tending to lump most participants into a singlegroup. Of the original effectiveness measures, the best discriminator is average precision, followed by R-precision. The two added performance measures do better at discriminating than the original measures. However, there is some concern that the scores do not meet the first assumption of the the analysis of variance, independence of the scores, since the precision score at each number of retrieved documents or recall level for a query will be related to the score at the previous level. The numerator in the precision score is a cumulation which includes the numerator in the previous score.

4. Conclusions

The lack of significant differences in the results of TREC-3 should not be interpreted as

indicating that it does not really matter how we do retrieval. The interesting fact to emerge from the analysis of variance is the high variability over queries. What this means is that some approaches are working well with some queries and other approaches well with other queries. The challenge will be to find out what characterizes the queries and the retrieval approaches which work well together. A multi-approach system can then determine, based on the characteristics of the query, what the optimal approach will be.

References

W.J. Conover (1980), Practical Nonparametric Statistics, 2d edution, New York, Wiley.
Scheffe Grouping
M. Berenson et al. (1983), Intermediate Statistical Methods and Applications: a Computer Package Approach. Englewood Cliffs, N.J.:Prentice-Hall.

Scheffé Grouping									Mean	Run
						A			0.42262	INQ102
		B				A			0.40118	city1
		B				A		C	0.37145	Brkly7
		B		D		A		C	0.36586	INQ101
E		B		D		A		C	0.35393	ASSCTV2
E		B		D		A		C	0.35037	ASSCTV1
E		B		D		A		C	F 0.34186	CrnlEA
E		B		D		A		C	F 0.33733	city2
E		B		D		A	G	C	F 0.33016	CrnlLA
E		B		D	H	A	G	C	F 0.31574	westp1
E		B	I	D	H	A	G	C	F 0.30207	VTc2s2
E		B	I	D	H	A	G	C	F 0.30012	pircs1
E	J	B	I	D	H	A	G	C	F 0.29162	ETH002
E	J	B	I	D	H	A	G	C	F 0.29141	VTc5s2
E	J	B	I	D	H	A	G	C	F 0.29129	pircs2
E	J	B	I	D	H	A	G	C	F 0.27749	Brkly6
E	J	B	I	D	H	A	G	C	F 0.27367	ETH001
E	J	B	I	D	H	A	G	C	F 0.27349	nyuir2
E	J	B	I	D	H	A	G	C	F 0.27222	nyuir1
E	J	B	I	D	H	A	G	C	F 0.2689	TOPIC4
E	J	B	I	D	H	*	G	C	F 0.25806	CLARTA
E	J	B	I	D	H	*	G	C	F 0.25773	dortD2
E	J	B	I	D	H		G	C	F 0.25311	citri1
E	J	B	I	D	H		G	C	F 0.24433	dortD1
E	J		I	D	H	K	G	C	F 0.23931	rutfua1
E	J		I	D	H	K	G	C	F 0.23926	lsia0mw20f
E	J		I	D	H	K	G	C	F 0.23255	lsia0mf
E	J		I	D	H	K	G	C	F 0.22541	rutfua2
E	J		I	D	H	K	G	C	F 0.22487	CLARTM
E	J	L	I	D	H	K	G		F 0.2092	xerox3
E	J	L	I	D	H	K	G		F 0.20884	siems1
E	J	L	I		H	K	G		F 0.20683	citri2
E	J	L	I		H	K	G		F 0.20613	erima1
	J	L	I		H	K	G		F 0.18726	siems2
	J	L	I		H	K	G	M	** 0.17518	padre2
	J	L	I		H	K		M	** 0.16929	xerox4
	J	L	I			K		M	** 0.14481	padre1
		L				K		M	0.0823	ACQNT1
		L						M	0.06245	virtu1
								M	0.02865	TOPIC3

*Included in A group when arcsine transformation is applied.
**Not included in M group when arcsine transformation is applied.

Table 3--Scheffé Test for Average Precision, Ad Hoc Data.
Minimum Significant Difference= 0.158, Alpha=.05.
Means with the same letter are not significantly different.

Scheffé Grouping												Mean Rank	Run
							A					36.9	INQ102
				B			A					34.39	city a1
				B			A					32.78	Brkly7
				B			A			C		32.3	INQ101
		E		B	D		A			C		32.13	ASSCTV2
		E		B	D		A			C	F	32.04	ASSCTV1
		E		B	D		A			C	F	32.01	city a2
		E		B	D		A			C	F	31.48	CrnlLA
		E		B	D		A		G	C	F	30.06	CrnlEA
		E		B	D		A		G	C	F	28.75	westp1
		E		B	D	H	A		G	C	F	27.14	ETH002
		E		B	D	H	A		G	C	F	26.32	pircs1
I		E		B	D	H	A		G	C	F	25.96	VTc2s2
I		E		B	D	H	A		G	C	F	25.89	Brkly6
I		E	J	B	D	H	A		G	C	F	25.59	pircs2
I	K	E	J	B	D	H	A		G	C	F	24.48	ETH001
I	K	E	J	B	D	H	A		G	C	F	24.35	VTc5s2
I	K	E	J	B	D	H	A		G	C	F	23.37	nyuir2
I	K	E	J	B	D	H		L	G	C	F	23.23	nyuir1
I	K	E	J	B	D	H		L	G	C	F	22.77	TOPIC4
I	K	E	J	B	D	H		L	G	C	F	21.02	dortD2
I	K	E	J		D	H		L	G	C	F	20.81	CLARTA
I	K	E	J		D	H		L	G	C	F	20.62	citri1
I	K	E	J		D	H		L	G	C	F	20.03	lsia0mw20f
I	K	E	J		D	H		L	G	C	F	20.01	rutfua1
I	K	E	J	M	D	H		L	G	C	F	19.26	dortD1
I	K	E	J	M	D	H		L	G		F	18.62	lsia0mf
I	K		J	M		H		L	G		F	18.58	rutfua2
I	K		J	M		H		L	G			18.33	CLARTM
I	K		J	M		H		L	G			17.98	siems1
I	K		J	M		H		L	G			16.13	xerox3
I	K		J	M		H		L	N			16.02	siems2
I	K		J	M		H		L	N			15.85	erima1
I	K		J	M	O	H		L	N			15.63	citri2
I	K		J	M	O	H		L	N			12.36	padre1
	K		J	M	O			L	N			12.18	padre2
	K			M	O			L	N			11.38	xerox4
				M	O			L	N			6.39	ACQNT1
					O				N			4.43	virtu1

Table 4--Friedman Test for Average Precision Ranks, Ad Hoc Data, Alpha = .05. Means with the same letter are not significantly different.

Scheffé Grouping								Mean	Run
				A				0.45238	INQ102
	B			A				0.42169	city a1
	B			A		C		0.41522	Brkly7
	B	D		A		C		0.4088	INQ101
	B	D		A		C		0.39989	ASSCTV2
	B	D		A		C		0.39482	ASSCTV1
E	B	D		A		C		0.38899	CmlEA
E	B	D		A		C	F	0.38155	city a2
E	B	D		A	G	C	F	0.37798	westp1
E	B	D		A	G	C	F	0.37679	CmlLA
E	B	D	H	A	G	C	F	0.35382	VTc2s2
E	B	D	H	A	G	C	F	0.35104	TOPIC4
E	B	D	H	A	G	C	F	0.34982	Brkly6
E	B	D	H	A	G	C	F	0.34844	pircs1
E	B	D	H	A	G	C	F	0.34749	ETH002
E	B	D	H	A	G	C	F	0.34042	VTc5s2
E	B	D	H	A	G	C	F	0.34015	pircs2
E	B	D	H	A	G	C	F	0.33741	ETH001
E	B	D	H	A	G	C	F	0.32318	nyuir1
E	B	D	H	A	G	C	F	0.32313	nyuir2
E	B	D	H	A	G	C	F	0.32276	citri1
E	B	D	H	A	G	C	F	0.32214	dortD2
E	B	D	H	A	G	C	F	0.31842	CLARTA
E	B	D	H	A	G	C	F	0.31635	rutfua1
E	B	D	H	A	G	C	F	0.31279	dortD1
E	B	D	H	A	G	C	F	0.30912	rutfua2
E	B	D	H	A	G	C	F	0.30711	lsia0mw20f
E	B	D	H	A	G	C	F	0.30303	lsia0mf
E	B	D	H	*	G	C	F	0.29106	citri2
E	B	D	H	I	G	C	F	0.28411	CLARTM
E	B	D	H	I	G	C	F	0.2822	siems1
E	B	D	H	I	G	C	F	0.27648	xerox3
E	J	D	H	I	G	C	F	0.26676	erima1
E	J	D	H	I	G		F	0.26349	siems2
E	J		H	I	G		F	0.23955	xerox4
	J		H	I	G			0.22789	padre2
	J		H	I				0.21768	padre1
	J			I		K		0.14588	ACQNT1
	J					K		0.11704	virtu1
						K		0.06099	TOPIC3

*Included in A group when arcsine transformation is applied.

Table 5 -- Scheffé Groups for R-Precision, Ad Hoc Data
Minimum Significant Difference= 0.1507, alpha=.05.
Means with the same letter are not significantly different.

Scheffé Grouping							Mean	Run
				A			0.49082	INQ102
	B			A			0.47592	citya1
	B			A		C	0.46633	Brkly7
	B	D		A		C	0.44204	ASSCTV2
E	B	D		A		C	0.44041	INQ101
E	B	D		A		C	0.43612	ASSCTV1
E	B	D		A		C	0.43429	citya2
E	B	D		A		C	F 0.42245	CrnlLA
E	B	D		A		C	F 0.41898	CrnlEA
E	B	D		A		C	F 0.40735	westp1
E	B	D		A		C	F 0.40612	VTc2s2
E	B	D		A		C	F 0.40571	TOPIC4
E	B	D		A		C	F 0.40041	ETH002
E	B	D		A		C	F 0.39898	VTc5s2
E	B	D		A		C	F 0.39327	Brkly6
E	B	D		A	G	C	F 0.38122	pircs1
E	B	D		A	G	C	F 0.38122	pircs2
E	B	D		A	G	C	F 0.37327	CLARTA
E	B	D		A	G	C	F 0.37204	ETH001
E	B	D		A	G	C	F 0.37122	rutfua1
E	B	D		A	G	C	F 0.36776	citri1
E	B	D		A	G	C	F 0.36224	nyuir1
E	B	D		A	G	C	F 0.36163	nyuir2
E	B	D		A	G	C	F 0.35878	rutfua2
E	B	D		A	G	C	F 0.35673	dortD2
E	B	D		A	G	C	F 0.35102	citri2
E	B	D		A	G	C	F 0.34939	CLARTM
E	B	D		A	G	C	F 0.33755	dortD1
E	B	D		A	G	C	F 0.32755	lsia0mw20f
E	B	D		A*	G	C	F 0.32673	siems1
E	B	D	H	A*	G	C	F 0.31571	lsia0mf
E	B	D	H		G	C	F 0.29816	xerox3
E	B	D	H		G	C	F 0.29204	erima1
E		D	H		G	C	F 0.28286	siems2
E		D	H		G		F 0.27184	padre2
E			H		G		F 0.25592	padre1
E			H		G		F 0.25571	xerox4
			H	I			0.19592	ACQNT1
			H	I			0.13735	virtu1
				I			0.05633	TOPIC3

*Not in A group in arcsine transformed data.

Table 6 -- Scheffé's Test for Precision at 100 Documents Retrieved, Ad Hoc Data.
Minimum Significant Difference= 0.1856, alpha=.05.
Means with the same letter are not significantly different.

Scheffé Grouping					Mean	Run
		A			0.4068	cityr1
	B	A			0.3887	pircs3
	B	A			0.3879	INQ104
	B	A			0.3838	INQ103
	B	A			0.3824	dortR1
	B	A		C	0.3748	pircs4
	B	A		C	0.3737	lsir2
	B	A		C	0.3724	crnlQR
	B	A		C	0.3699	crnlRR
	B	A		C	0.3642	Brkly8
	B	A		C	0.3621	cityr2
	B	A		C	0.3535	westp2
	B	A		C	0.3373	losPA1
E	B	A		C	0.3277	UCF101
E	B	A		C	0.3244	nyuir
E	B	A		C	0.3188	FDF2
E	B	A		C	0.3155	FDF1
E	B	A		C	0.3154	ETH004
E	B	A		C	0.3139	CLARTA
E	B	A		C	0.3111	xerox2
E	B	A	G	C	0.3092	ETH003
E	B	A	G	C	0.2879	lsir1
E	B	A	G	C	0.2867	xerox1
E	B	G	C	F	0.2774	TOPIC2
E	B	G	C	F	0.2754	rutir2
E	B	G	C	F	0.2742	nyuir1
E	B	G	C	F	0.2717	virtu2
E	B	G	C	F	0.2641	ACQNT2
E		G	C	F	0.2528	erimr1
E		G	C	F	0.2498	cityi1
E		G		F	0.2243	TOPIC1
E		G		F	0.2045	rutir1
		G		F	0.1854	rutfur1
		G			0.1817	rutfur2

Table 7 -- Scheffé's Test for Average Precision, Routing Data.
Minimum Significant Difference= 0.1277, alpha=.05.
Means with the same letter are not significantly different.

Ad Hoc Data

	Aver. Prec.	R Prec.	Precision at N=30	N=100	N=200	R=.5	R=.9
Ave.Prec.	1.000	0.987	0.956	0.972	0.983	0.987	0.766
R Prec.		1.000	0.977	0.989	0.993	0.968	0.704
N=30			1.000	0.986	0.974	0.916	0.636
N=100				1.000	0.993	0.940	0.674
N=200					1.000	0.965	0.694
R=.5						1.000	0.750
R=.9							1.000

Routing Data

	Aver. Prec.	R Prec.	Precision at N=30	N=100	N=200	R=.5	R=.9
Ave.Prec.	1.000	0.988	0.928	0.974	0.970	0.984	0.844
R Prec.		1.000	0.921	0.968	0.971	0.979	0.782
N=30			1.000	0.968	0.922	0.876	0.707
N=100				1.000	0.985	0.948	0.794
N=200					1.000	0.963	0.805
R=.5						1.000	0.838
R=.9							1.000

Table 8 -- Correlation of Selected Performance Measures.

Variable	Ad-Hoc Num.	%	Routing Num.	%
Ave. Precision	20	47.62	22	64.71
R-Precision	28	66.67	28	82.35
Precision at n=5	42	100.00	33	97.06
n=10	40	95.24	32	94.12
n=15	40	95.24	31	91.1
n=20	39	92.86	27	79.4
n=30	36	85.71	27	79.41
n=100	31	73.81	28	82.35
n=200	34	80.95	30	88.24
n=50	36	85.71	31	91.18
n=1000	38	90.48	31	91.18
Precision at r=0	39	92.86	34	100.00
r=.1	29	69.05	31	91.18
r=.2	27	64.29	30	88.24
r=.	30	71.43	26	76.47
r=.4	29	69.05	28	82.35
r=.5	28	66.67	30	88.24
r=.6	30	71.43	27	79.41
r=.7	27	64.29	27	79.41
r=.8	31	73.81	28	82.35
r=.9	18	42.86	30	88.24
r=1	42	100.00	34	100.00
Precision avererage over				
9 levels of n	14	33.33	17	50.00
11 levels of r	7	16.67	13	38.24

Table 9 -- Size of Top Group of Runs (A Group)

APPENDIX A

This appendix contains results for all TREC-3 participants. The first three pages list the system tags, organizations, and query construction methods for all runs. The next 9 pages describe the evaluation techniques and measures used. The rest of the appendix contains results.

ADHOC RUNS

CATEGORY A DATA

Tag	Organization	Query Construction Method
ACQNT1	National Security Agency	automatic
ASSCTV1	Mead Data Central, Inc.	manual
ASSCTV2	Mead Data Central, Inc.	manual
Brkly6	University of California, Berkeley	automatic
Brkly7	University of California, Berkeley	manual
citri1	Royal Melbourne Institute of Technology	automatic
citri2	Royal Melbourne Institute of Technology	automatic
citya1	City University, London	automatic
citya2	City University, London	automatic
CLARTA	Carnegie Mellon University / CLARITECH	automatic
CLARTM	Carnegie Mellon University / CLARITECH	manual
CrnlEA	Cornell University	automatic
CrnlLA	Cornell University	automatic
dortD1	Universitaet Dortmund	automatic
dortD2	Universitaet Dortmund	automatic
erima1	The Environmental Research Institute of Michigan	automatic
ETH001	Swiss Federal Institute of Technology	automatic
ETH002	Swiss Federal Institute of Technology	automatic
INQ101	University of Massachusetts, Amherst	automatic
INQ102	University of Massachusetts, Amherst	manual
lsia0mf	Bellcore	automatic
lsia0mw20f	Bellcore	automatic
nyuir1	New York University	automatic
nyuir2	New York University	automatic
padre1	Australian National University	manual
padre2	Australian National University	automatic
pircs1	Queens College, CUNY	automatic
pircs2	Queens College, CUNY	automatic
rutfua1	Rutgers University	manual
rutfua2	Rutgers University	manual
siems1	Siemens Corporate Research Inc.	automatic
siems2	Siemens Corporate Research Inc.	automatic
TOPIC3	Verity, Inc.	manual
TOPIC4	Verity, Inc.	interactive
virtu1	NEC Corporation	automatic
VTc2s2	Virginia Polytechnic Institute	manual
VTc5s2	Virginia Polytechnic Institute	manual
westp1	Westlaw Publishing Company	automatic
xerox3	Xerox Palo Alto Research Center	automatic
xerox4	Xerox Palo Alto Research Center	automatic

CATEGORY B DATA

Tag	Organization	Query Construction Method
DCUNL1	Dublin City University	manual
DCUNL2	Dublin City University	manual
dgros1	George Mason University	automatic

CATEGORY B DATA (Continued)

Tag	Organization	Query Construction Method
expst2	Mayo Clinic/Foundation	automatic
UniNE1	Universite de Neuchatel, Switzerland	automatic
UniNE2	Universite de Neuchatel, Switzerland	automatic

ROUTING RUNS

CATEGORY A DATA

Tag	Organization	Query Construction Method
ACQNT2	National Security Agency	automatic
Brkly8	University of California, Berkeley	automatic
cityi1	City University, London	interactive
cityr1	City University, London	automatic
cityr2	City University, London	automatic
CLARTA	Carnegie Mellon University / CLARITECH	automatic
crnlQR	Cornell University	automatic
crnlRR	Cornell University	automatic
dortR1	Universitaet Dortmund	automatic
erimr1	The Environmental Research Institute of Michigan	automatic
ETH003	Swiss Federal Institute of Technology	automatic
ETH004	Swiss Federal Institute of Technology	automatic
FDF1	TRW, Paracel	automatic
FDF2	TRW, Paracel	automatic
INQ103	University of Massachusetts, Amherst	automatic
INQ104	University of Massachusetts, Amherst	manual
losPA1	Logicon Operating Systems	automatic
lsir1	Bellcore	automatic
lsir2	Bellcore	automatic
nyuir1	New York University	automatic
nyuir2	New York University	automatic
pircs3	Queens College, CUNY	automatic
pircs4	Queens College, CUNY	automatic
pixtex	DeMontfort University	manual
rutfur1	Rutgers University	manual
rutfur2	Rutgers University	manual
rutir1	Rutgers University	interactive
rutir2	Rutgers University	interactive
TOPIC1	Verity, Inc.	manual
TOPIC2	Verity, Inc.	interactive
UCF101	University of Central Florida	manual
virtu2	NEC Corporation	manual
westp2	Westlaw Publishing Company	automatic
xerox1	Xerox Palo Alto Research Center	automatic
xerox2	Xerox Palo Alto Research Center	automatic

CATEGORY B DATA

Tag	Organization	Query Construction Method
expst1	Mayo Clinic/Foundation	automatic
stpat1	University of Toronto	interactive
UCFSJM	University of Central Florida	manual

OTHER DATA

Tag	Organization	Query Construction Method
TeknosN05	University of Minnesota	manual
TeknosN10	University of Minnesota	manual
TeknosN15	University of Minnesota	manual
TeknosN20	University of Minnesota	manual
TeknosN30	University of Minnesota	manual
TeknosRel	University of Minnesota	manual
TeknosW05	University of Minnesota	manual
TeknosW10	University of Minnesota	manual
TeknosW15	University of Minnesota	manual
TeknosW20	University of Minnesota	manual
TeknosW30	University of Minnesota	manual

Evaluation Techniques and Measures

Categories

The results are separated into categories according to two tasks, adhoc or routing. Table 1 contains a further breakdown of runs depending on how the query was built (automatic, manual, or interactive) and the magnitude of the task (Category A or Category B).

TABLE 1

Runs broken down by category and query construction method.

48 Adhoc Runs				
Category	Automatic	Manual	Interactive	Total
A	28	12	2	42
B	4	2	0	6
49 Routing Runs				
Category	Automatic	Manual	Interactive	Total
A	24	18	4	46
B	1	1	1	3

I. Adhoc.
Retrieval using an "adhoc" topic such as a researcher might use in a library environment. In TREC this implies that the input topic has no training material such as relevance judgments to aid in the construction of the input query.

A. Category A.
Systems running TREC topics against all documents from Disks 1 and 2 of the Tipster Collection.

B. Category B.
Systems running TREC topics against the Wall Street Journal (WSJ) on Disks 1 and 2 of the Tipster Collection. (Intended for new groups, allowing them to scale their systems to handle large collections.)

II. Routing.
Retrieval using a "routing" query such as a profile to filter some incoming document stream. In TREC this implies that the input topic has training material, including relevance judgments against the training documents, to use in constructing the input query or profile. This query is then used against new documents (the test documents).

A. Category A.
Systems running TREC topics against all documents from Disk 3 of the Tipster Collection.

B. Category B.
Systems running TREC topics against only documents from the San Jose Mercury News (SJMN) on Disk 3 of the Tipster Collection. (Intended for new

groups, allowing them to scale their systems to handle large collections.)

Evaluation Measures

I. Recall.

A measure of the ability of a system to present all relevant items,

$$\text{recall} = \frac{\text{number of relevant items retrieved}}{\text{number of relevant items in collection}}.$$

II. Precision.

A measure of the ability of a system to present only relevant items,

$$\text{precision} = \frac{\text{number of relevant items retrieved}}{\text{total number of items retrieved}}.$$

III. Fallout.

A measure of the ability of a system to filter out non-relevant items,

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

System Results Description

Each page contains all the results for one system comprised of a header, 4 tables, and 3 graphs described as follows:

Header

- The header contains the task and organization name, where task is either adhoc or routing and the organization is the organization name producing the run described on the page.

Tables

Tables are generated by *trec_eval* courtesy of Chris Buckley using the SMART methodology as defined in Salton and McGill [1].

I. "Summary Statistics" Table.
Table 2 is a sample "Summary Statistics" Table.

TABLE 2
Sample "Summary Statistics" Table.

Summary Statistics	
Run Number	CrnlEA–category A, automatic
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	7267

A. Run Number.
An identifier for a system of the form: *Tag - Category, Query Construction Method*, where *Tag* is the id of the run provided by the participant, *Category* is either Category A or Category B (full documents or a subset of full documents), and *Query Construction Method* is either automatic, manual, or interactive.

B. Number of Topics.
Number of topics searched in this run (normally 50 topics are run for each task).

C. Total number of documents over all topics (the number of topics shown in B).

i. Retrieved.
Number of documents retrieved and sent in. This is normally 50,000 (50 topics × 1000 documents), but could be less.

ii. Relevant.
Total possible relevant documents within a given task and category. Table 3 contains the possible relevant documents for TREC-3.

TABLE 3
Possible relevant documents within a given task and category for TREC-3.

Category	Adhoc	Routing
A	9805	9353
B	3913	2559

iii. Rel_ret.
Total number of relevant documents returned by a run for all the topics, i.e. the Number of Topics shown on the second line of the Summary Table.

II. "Recall Level Precision Averages" Table.
Table 4 is a sample "Recall Level Precision Averages" Table.

TABLE 4
Sample "Recall Level Precision Averages" Table.

Recall Level Precision Averages	
Recall	Precision
0.00	0.7759
0.10	0.5910
0.20	0.5244
0.30	0.4735
0.40	0.4213
0.50	0.3595
0.60	0.3031
0.70	0.2365
0.80	0.1605
0.90	0.0664
1.00	0.0031
Average precision over all relevant docs	
non-interpolated	0.3419

A. Precision at 11 recall cutoff values.
Each recall-precision average is computed by summing the precisions at the specified recall cutoff value (denoted by $\sum P_R$ where P_R is the precision at recall cutoff value R) and then dividing by the number of topics, for TREC normally $NUM = 50$.

$$\frac{\sum_{i=1}^{NUM} P_R}{NUM} \qquad R = \{0.0, 0.1, 0.2, 0.3, \ldots, 1.0\}$$

- Interpolating recall-precision.
  In order to graphically show precision at various recall averages, interpolation must be used. The interpolated precision at a recall cutoff R is defined to be the maximum precision at all points $\leq$ R.

  For example, if there are only 3 relevant documents retrieved, and these are retrieved at ranks 4, 9, and 20, then the exact recall points are 0.33, 0.67, and 1.0. Interpolated precisions are computed using the "true" recall values (precision 0.25 at recall 0.33, precision 0.22 at recall 0.67, and precision 0.15 at recall 1.0, respectively) and mapping them to the 11 recall cutoff values using the above rule. Therefore, the precisions at recall points 0.0, 0.1, 0.2, 0.3 are 0.25, the precision at recall points 0.4, 0.5 0.6, are 0.22 and precision at recall points 0.7, 0.8, 0.9, 1.0 are 0.15. Note that theoretically precision is not defined at a recall of 0.0, however this interpolation rule allows values to be determined.

B. Average precision over all relevant documents, non-interpolated.
   This measure is not an average of the above cutoff values, but an average calculated after each relevant returned document.

   Consider a system returning 10 documents and four of the 10 documents are relevant. The rankings of the four documents are 1, 2, 4, 7 giving precisions of 1, 1, 0.75, and 0.57, respectively. By averaging the 4 precisions the single value measure of average precision over all relevant documents is 0.83.

III. "Document Level Averages" Table.
Table 5 is a sample "Document Level Averages" Table.

TABLE 5
Sample "Document Level Averages" Table.

Document Level Averages	
	Precision
At 5 docs	0.5760
At 10 docs	0.5540
At 15 docs	0.5480
At 20 docs	0.5370
At 30 docs	0.5187
At 100 docs	0.4168
At 200 docs	0.3413
At 500 docs	0.2254
At 1000 docs	0.1453
R−Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3890

A. Precision at 9 document cutoff values.
Each document precision average is computed by summing the (50) precisions at the specified document cutoff value divided by the number of topics (50).

B. R-Precision is the precision after R documents (whether relevant or non-relevant) have been retrieved, where R is the number of relevant documents for a topic. R-Precisions are computed, one for each query (50), and then they are averaged.

Suppose a topic with 50 relevant documents was run returning 200 documents. In the top 50 documents returned, 17 of them are relevant. Then the R-Precision is $\frac{17}{50}$ or 0.34.

IV. "Recall Fallout Averages" Table.
Table 6 is a sample "Recall Fallout Averages" Table.

TABLE 6
Sample "Recall Fallout Averages" Table.

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00008
0.30	0.00014
0.40	0.00023
0.50	0.00037
0.60	0.00058
0.70	0.00094
0.80	0.00175
0.90	0.00528
1.00	0.13424

A. Fallout at 11 recall cutoff values.
Each recall-fallout average is computed by summing the fallouts at the specified recall cutoff value (denoted by $\sum F_R$ where F_R is the fallout at recall cutoff value R) and then divided by the number of topics ($NUM = 50$).

$$\frac{\sum_{i=1}^{NUM} F_R}{NUM} \qquad R = \{0.0, 0.1, 0.2, 0.3, \ldots, 1.0\}$$

- Interpolating recall-fallout.
The recall at X non relevant documents is used for the recall at $X - 1$ non relevant documents. When fallout is not exactly defined at R, the fallout of $X - 1$ is used.

Tabular Interpretation

I. Recall Level Precision Averages.

A. Precision at 11 cutoff values.
This table allows comparisons of systems.

B. Average precision over all relevant documents.
This is a single valued number which reflects the performance over all relevant documents. It is intended to reward those systems retrieving relevant documents quickly (highly ranked).

II. Document Level Averages.

A. Precision at 9 document cutoff values.
Document level reflects actual measured system performance as a user might see it. However, the averages computed using document-level measures are difficult to compare.

B. R-Precision.
It is a measure that is intended to de-emphasize the exact ranking of the documents. As such, it is especially valuable for examining routing systems, where the real-life criteria is whether a document is given to a user, rather than at what rank the document would be. R-Precision is particularly useful in TREC where there are large numbers of relevant documents.

III. Recall Fallout Averages.
Fallout is a parallel measure to recall for measuring the nonrelevant documents. An effective retrieval system will exhibit maximum recall and minimal fallout.

Graphs

I. Recall-Precision Graph.
Figure 1 is a sample Recall-Precision Graph.

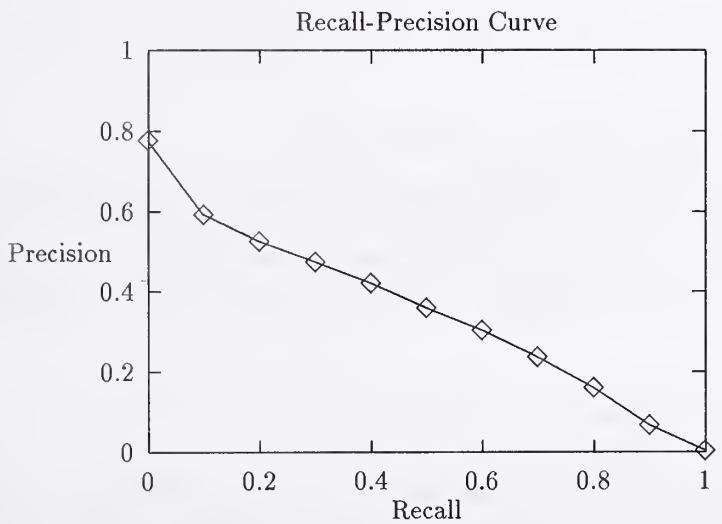


FIG. 1. *Sample Recall-Precision Graph.*

The Recall-Precision Graph is created using the 11 cutoff values from the Recall Level Precision Averages. This graph is useful for comparing systems. The graphs of

different systems can be superimposed on the same graph to determine which system is superior. Curves closest to the upper right-hand corner of the graph (where recall and precision are maximized) indicates the best performance. Comparisons are best made in three different recall ranges: 0 to 0.2, 0.2 - 0.8, and 0.8 to 1. These ranges characterize systems as high precision, middle recall, and high recall, respectively.

II. Average Precision Histogram.
Figure 2 is a sample Average Precision Histogram.

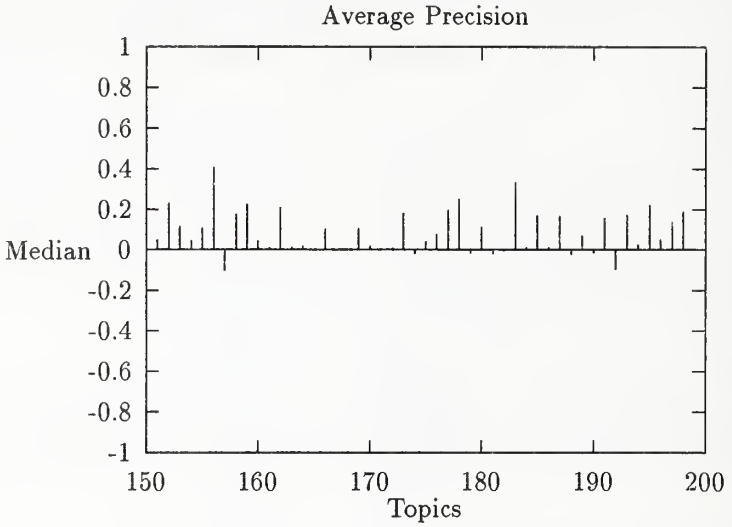


FIG. 2. *Sample Average Precision Histogram.*

The Average Precision Histogram measures the precision of a system on each topic against the median precision of all systems on that topic.

III. Recall-Log Fallout Curve.
Figure 3 is a sample Recall-Log Fallout Curve.

The Recall-Log Fallout Graph is generated using the 11 cutoff values from the Recall Fallout Averages (exactly like the Recall-Precision Graph). Curves closest to the lower left-hand corner of the graph (where recall is maximized and fallout is minimized) indicates the best performance. Fallout is graphed using a logarithmic scale for viewing purposes.

Graphical Interpretation

I. Recall-Precision Graph.
This graph is the most common used to compare information retrieval systems. Typically these graphs slope downward from left to right, enforcing the notion that as more relevant documents are retrieved (recall increases) then the more non relevant documents are retrieved (precision decreases).

II. Average Precision Histogram.
This graph is intended to give insight into the performance of individual systems and the types of topics they handle well.

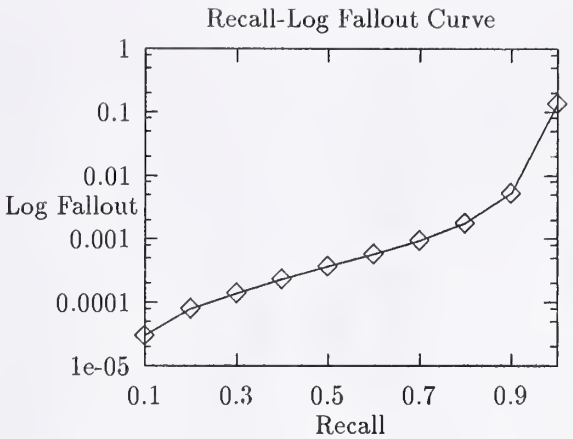


FIG. 3. *Sample Recall-Log Fallout Curve.*

III. Recall-Log Fallout Curve.
This graph illustrates the same concept as the recall-precision graph, except the fallout measure is used. As systems retrieve all relevant documents (recall increases) the number of non-relevant documents returned increases (fallout increases).

References

[1] G. SALTON AND M. MCGILL, *Introduction to Modern Information Retrieval*, McGraw-Hill, New York, first ed., 1983.

Summary Statistics	
Run Number	ACQNT1–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	2753

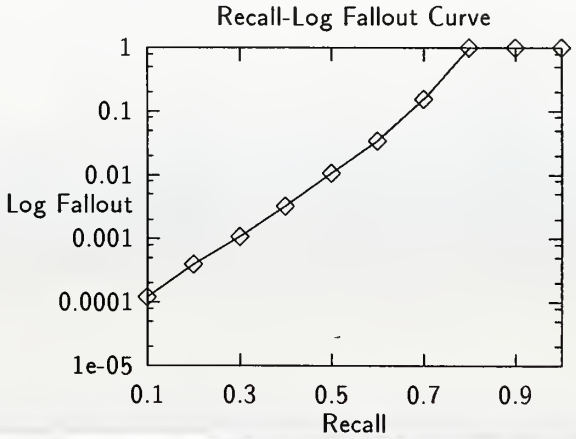
Recall Level Precision Averages	
Recall	Precision
0.00	0.5568
0.10	0.2622
0.20	0.1718
0.30	0.1042
0.40	0.0489
0.50	0.0191
0.60	0.0073
0.70	0.0019
0.80	0.0000
0.90	0.0000
1.00	0.0000

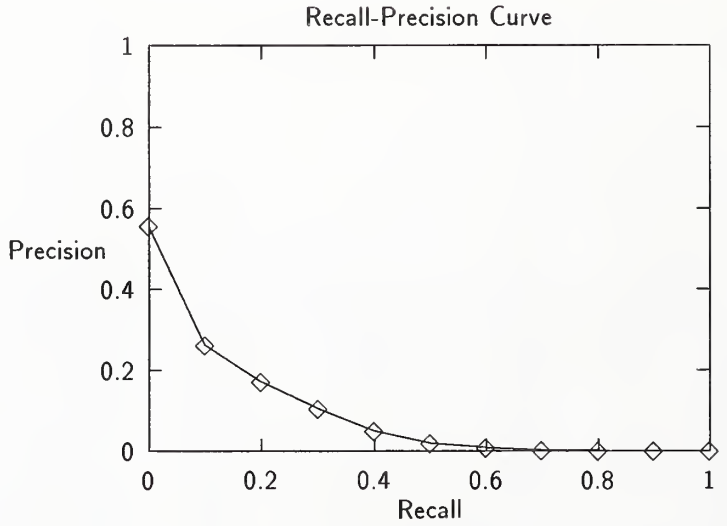
Average precision over all relevant docs	
non-interpolated	0.0823

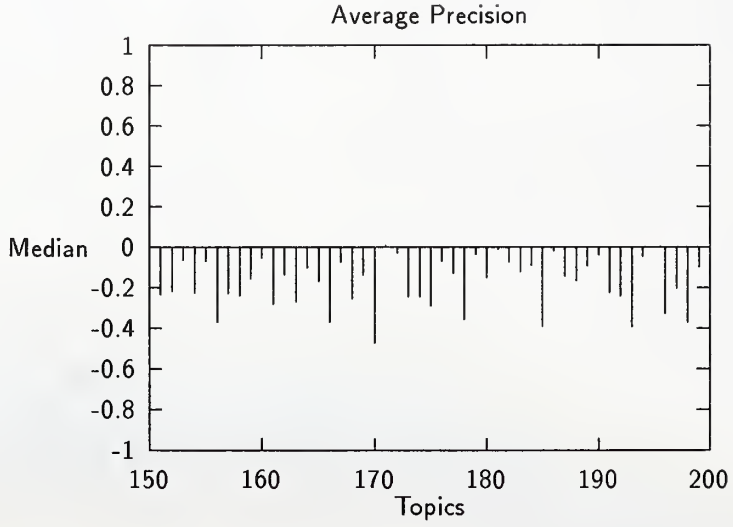
Document Level Averages	
	Precision
At 5 docs	0.3560
At 10 docs	0.3320
At 15 docs	0.3107
At 20 docs	0.2960
At 30 docs	0.2713
At 100 docs	0.1958
At 200 docs	0.1497
At 500 docs	0.0889
At 1000 docs	0.0551

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1459

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00012
0.20	0.00040
0.30	0.00108
0.40	0.00325
0.50	0.01072
0.60	0.03406
0.70	0.15351
0.80	1.00000
0.90	1.00000
1.00	1.00000







Summary Statistics	
Run Number	ASSCTV1–category A, manual
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	6833

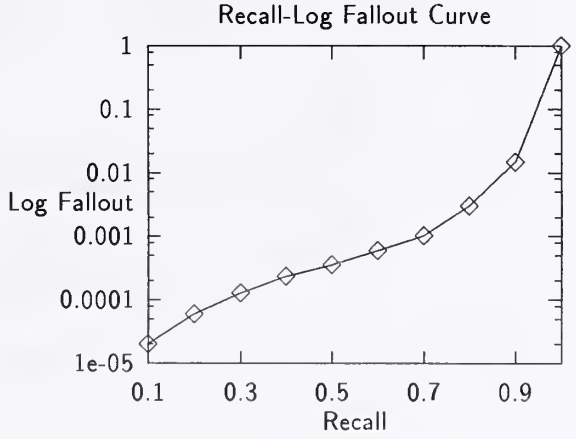
Recall Level Precision Averages	
Recall	Precision
0.00	0.9171
0.10	0.6610
0.20	0.5670
0.30	0.4906
0.40	0.4124
0.50	0.3652
0.60	0.2924
0.70	0.2204
0.80	0.1002
0.90	0.0250
1.00	0.0000

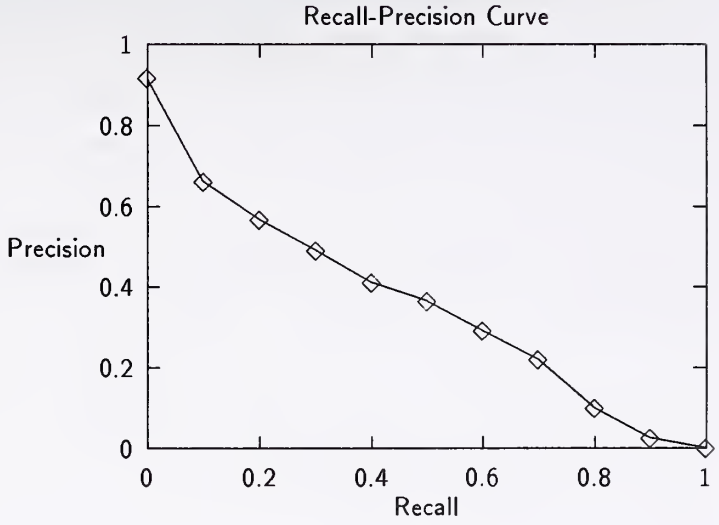
Average precision over all relevant docs	
non-interpolated	0.3504

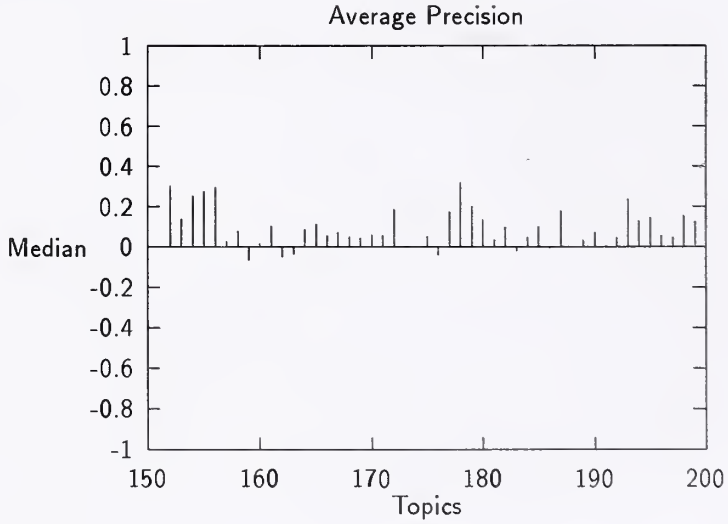
Document Level Averages	
	Precision
At 5 docs	0.7080
At 10 docs	0.6540
At 15 docs	0.6253
At 20 docs	0.5970
At 30 docs	0.5680
At 100 docs	0.4352
At 200 docs	0.3486
At 500 docs	0.2184
At 1000 docs	0.1367

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3948

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00002
0.20	0.00006
0.30	0.00013
0.40	0.00024
0.50	0.00036
0.60	0.00061
0.70	0.00103
0.80	0.00300
0.90	0.01465
1.00	1.00000







Summary Statistics	
Run Number	ASSCTV2–category A, manual
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	6829

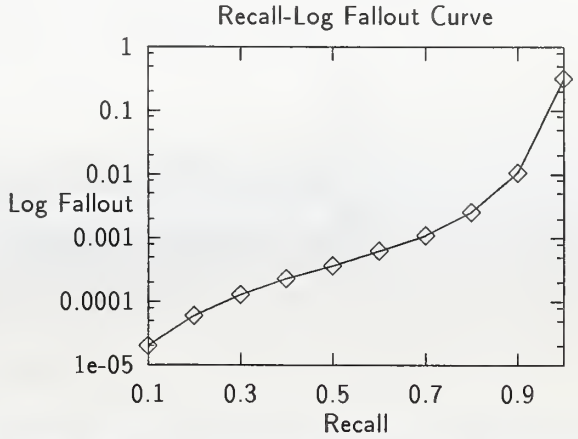
Recall Level Precision Averages	
Recall	Precision
0.00	0.9337
0.10	0.6724
0.20	0.5663
0.30	0.4901
0.40	0.4244
0.50	0.3590
0.60	0.2844
0.70	0.2087
0.80	0.1152
0.90	0.0339
1.00	0.0013

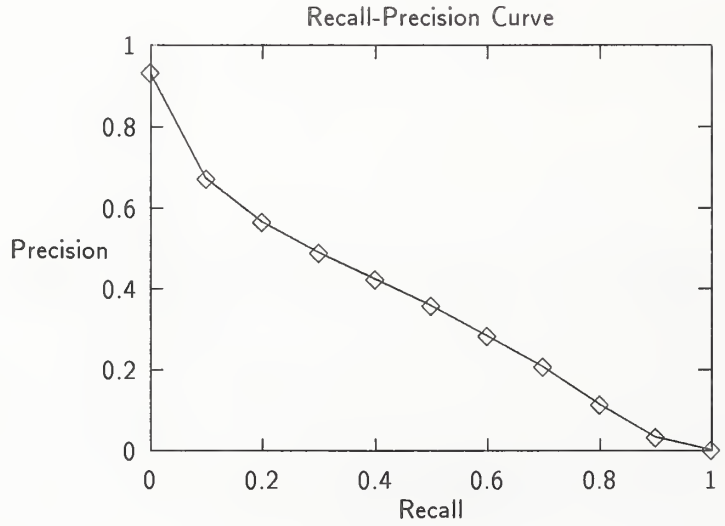
Average precision over all relevant docs	
non-interpolated	0.3539

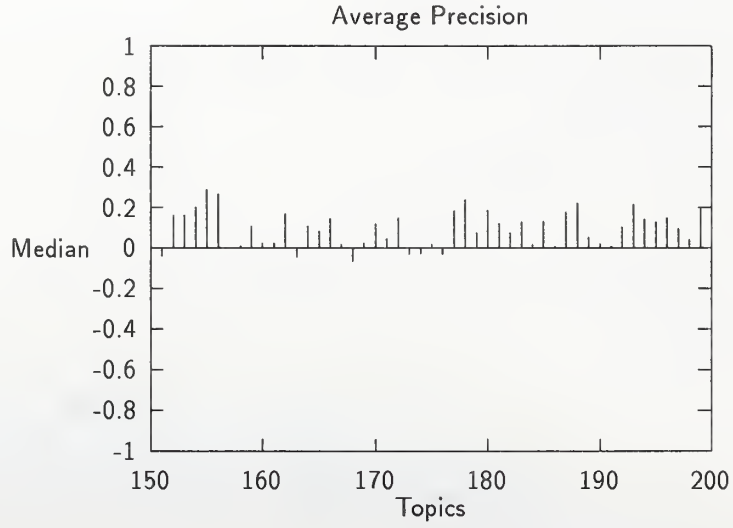
Document Level Averages	
	Precision
At 5 docs	0.7280
At 10 docs	0.6760
At 15 docs	0.6453
At 20 docs	0.6110
At 30 docs	0.5693
At 100 docs	0.4404
At 200 docs	0.3406
At 500 docs	0.2151
At 1000 docs	0.1366

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3999

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00002
0.20	0.00006
0.30	0.00013
0.40	0.00023
0.50	0.00037
0.60	0.00063
0.70	0.00111
0.80	0.00257
0.90	0.01071
1.00	0.32070







Summary Statistics	
Run Number	Brkly6–category A, automatic
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	6014

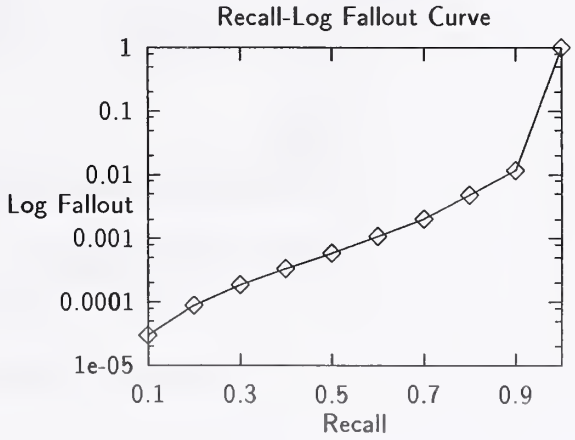
Recall Level Precision Averages	
Recall	Precision
0.00	0.8568
0.10	0.5757
0.20	0.4863
0.30	0.3967
0.40	0.3289
0.50	0.2620
0.60	0.1877
0.70	0.1266
0.80	0.0655
0.90	0.0314
1.00	0.0000

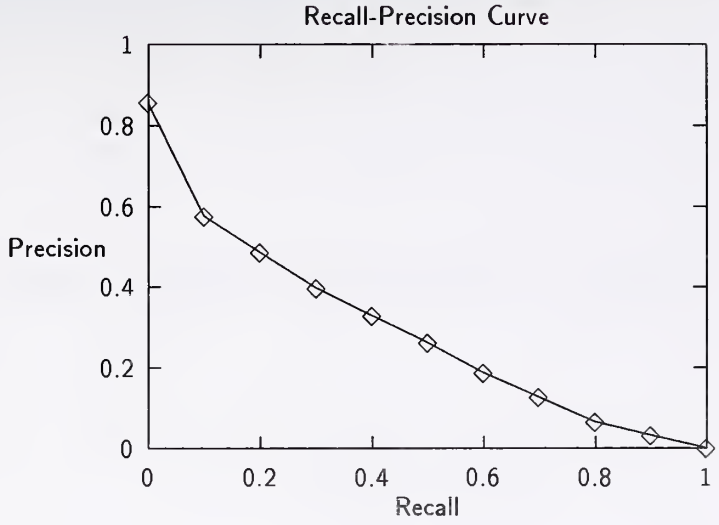
Average precision over all relevant docs	
non-interpolated	0.2775

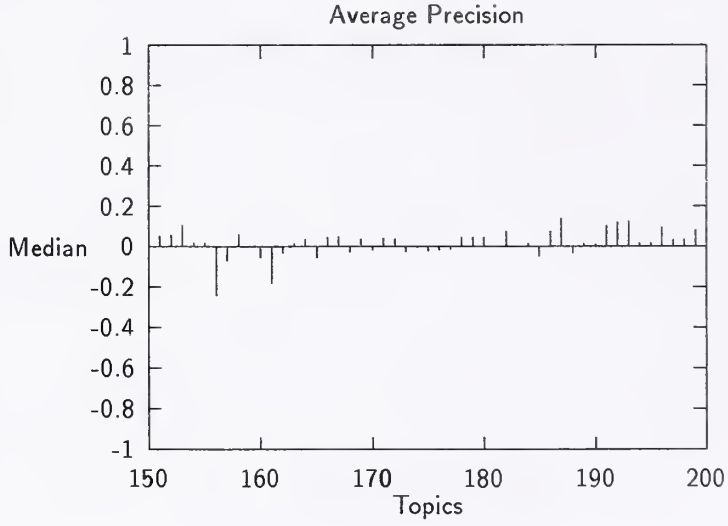
Document Level Averages	
	Precision
At 5 docs	0.6480
At 10 docs	0.6380
At 15 docs	0.5800
At 20 docs	0.5560
At 30 docs	0.5160
At 100 docs	0.3924
At 200 docs	0.2989
At 500 docs	0.1847
At 1000 docs	0.1203

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3498

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00009
0.30	0.00019
0.40	0.00034
0.50	0.00059
0.60	0.00108
0.70	0.00202
0.80	0.00476
0.90	0.01159
1.00	1.00000







Summary Statistics	
Run Number	Brkly7–category A, manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	6757

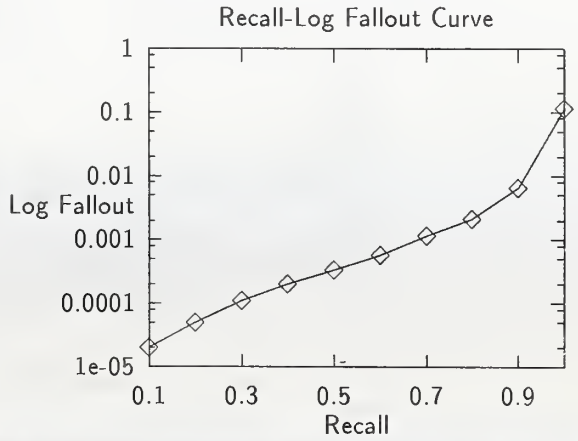
Recall Level Precision Averages	
Recall	Precision
0.00	0.9039
0.10	0.6922
0.20	0.6051
0.30	0.5355
0.40	0.4551
0.50	0.3788
0.60	0.3018
0.70	0.2031
0.80	0.1349
0.90	0.0543
1.00	0.0036

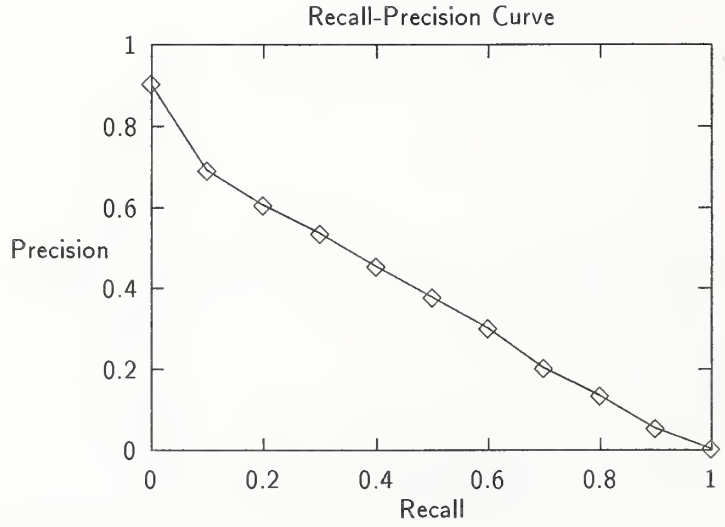
Average precision over all relevant docs	
non-interpolated	0.3714

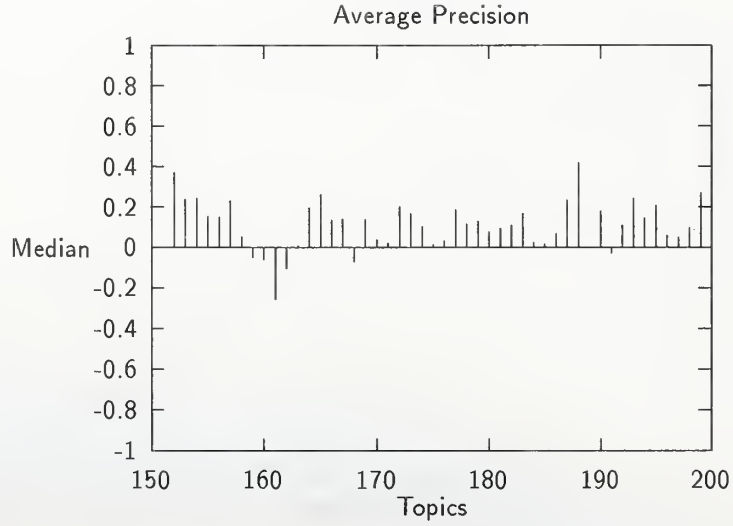
Document Level Averages	
	Precision
At 5 docs	0.7600
At 10 docs	0.7120
At 15 docs	0.6813
At 20 docs	0.6490
At 30 docs	0.6100
At 100 docs	0.4666
At 200 docs	0.3634
At 500 docs	0.2175
At 1000 docs	0.1351

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4152

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00002
0.20	0.00005
0.30	0.00011
0.40	0.00020
0.50	0.00034
0.60	0.00058
0.70	0.00115
0.80	0.00214
0.90	0.00654
1.00	0.11554







Summary Statistics	
Run Number	citri1–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	5791

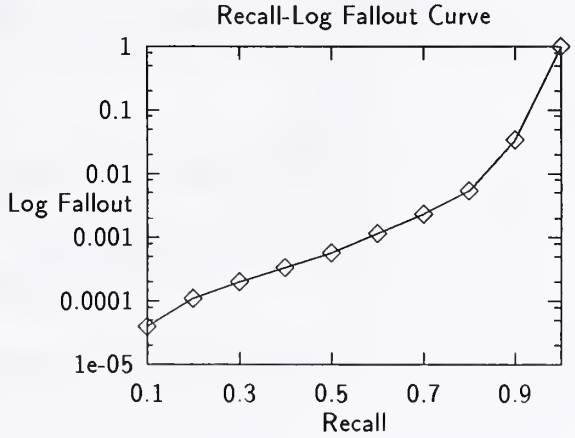
Recall Level Precision Averages	
Recall	Precision
0.00	0.7344
0.10	0.5027
0.20	0.4303
0.30	0.3799
0.40	0.3325
0.50	0.2653
0.60	0.1787
0.70	0.1124
0.80	0.0582
0.90	0.0108
1.00	0.0000

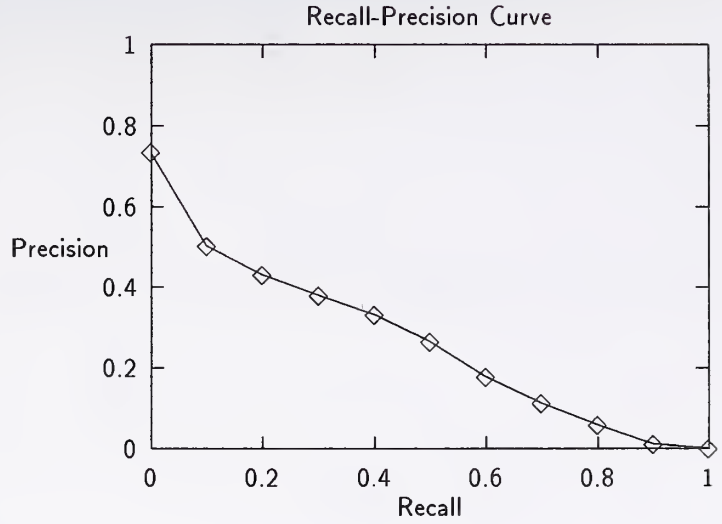
Average precision over all relevant docs	
non-interpolated	0.2531

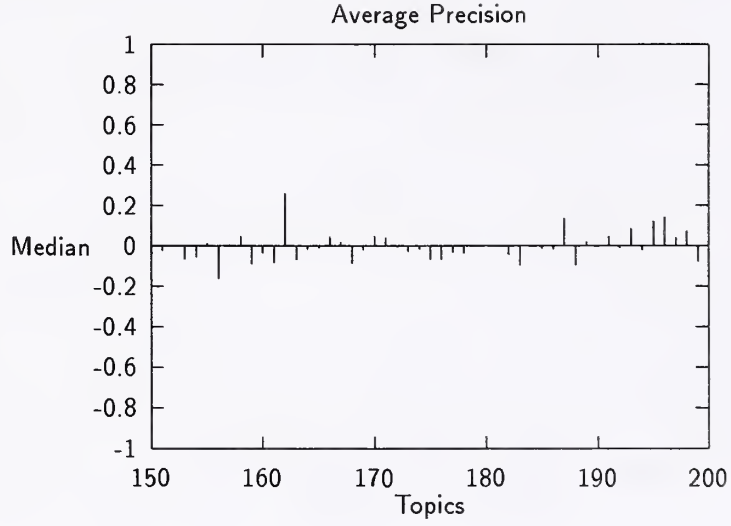
Document Level Averages	
	Precision
At 5 docs	0.5200
At 10 docs	0.5180
At 15 docs	0.5013
At 20 docs	0.4680
At 30 docs	0.4513
At 100 docs	0.3636
At 200 docs	0.2875
At 500 docs	0.1854
At 1000 docs	0.1158

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3228

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00004
0.20	0.00011
0.30	0.00020
0.40	0.00034
0.50	0.00058
0.60	0.00115
0.70	0.00231
0.80	0.00540
0.90	0.03441
1.00	1.00000







Summary Statistics	
Run Number	citri2–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	4504

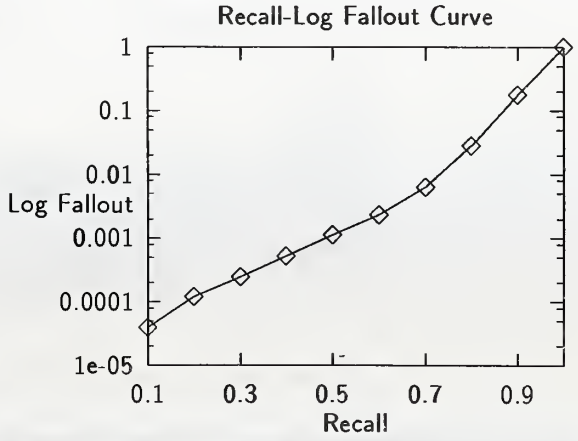
Recall Level Precision Averages	
Recall	Precision
0.00	0.7490
0.10	0.5060
0.20	0.4036
0.30	0.3346
0.40	0.2400
0.50	0.1543
0.60	0.0976
0.70	0.0441
0.80	0.0117
0.90	0.0021
1.00	0.0000

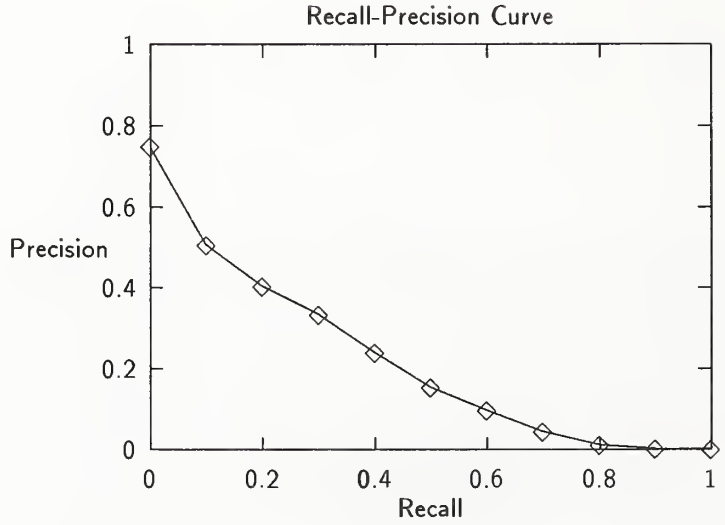
Average precision over all relevant docs	
non-interpolated	0.2068

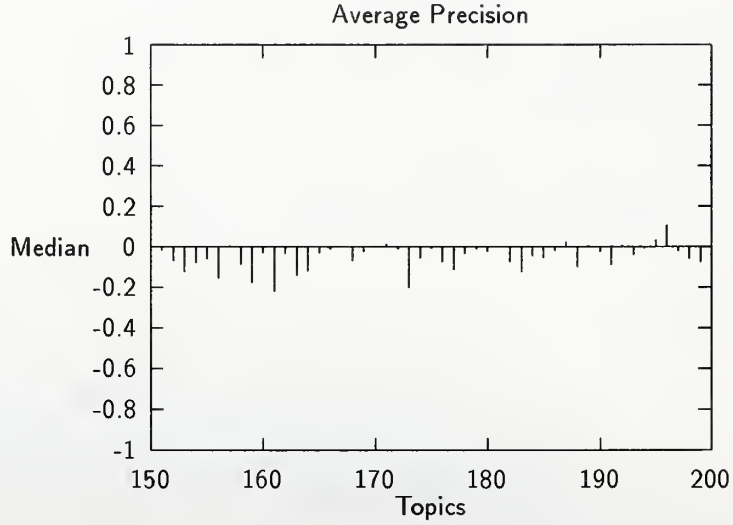
Document Level Averages	
	Precision
At 5 docs	0.5120
At 10 docs	0.5240
At 15 docs	0.4853
At 20 docs	0.4750
At 30 docs	0.4533
At 100 docs	0.3466
At 200 docs	0.2547
At 500 docs	0.1500
At 1000 docs	0.0901

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2911

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00004
0.20	0.00012
0.30	0.00025
0.40	0.00053
0.50	0.00114
0.60	0.00232
0.70	0.00633
0.80	0.02821
0.90	0.17853
1.00	1.00000







Summary Statistics	
Run Number	cityal–category A, automatic
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	7246

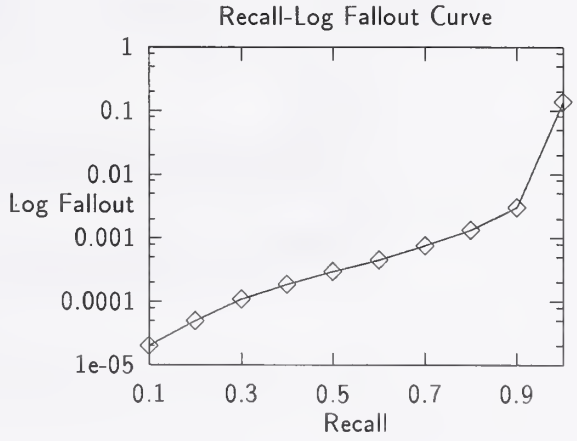
Recall Level Precision Averages	
Recall	Precision
0.00	0.8827
0.10	0.6991
0.20	0.6059
0.30	0.5334
0.40	0.4714
0.50	0.4090
0.60	0.3515
0.70	0.2786
0.80	0.1990
0.90	0.1116
1.00	0.0030

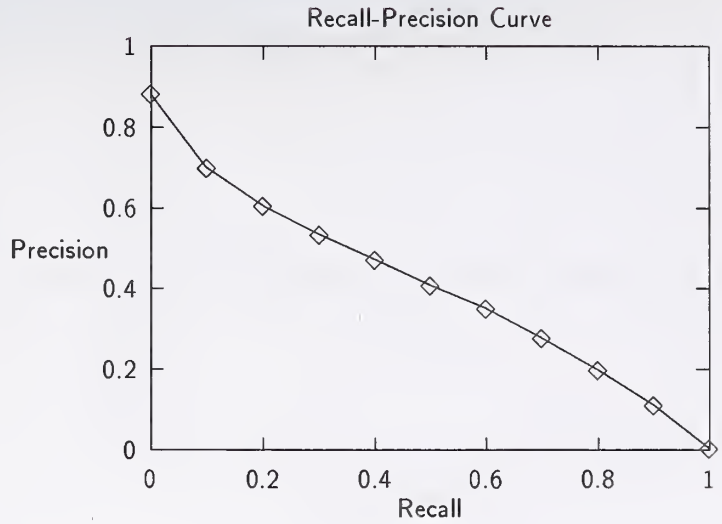
Average precision over all relevant docs	
non-interpolated	0.4012

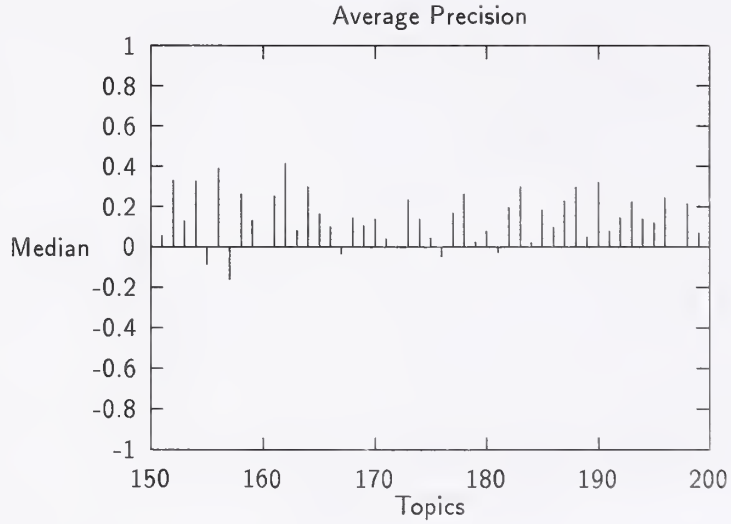
Document Level Averages	
	Precision
At 5 docs	0.7400
At 10 docs	0.7120
At 15 docs	0.6787
At 20 docs	0.6640
At 30 docs	0.6247
At 100 docs	0.4758
At 200 docs	0.3811
At 500 docs	0.2356
At 1000 docs	0.1449

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4217

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00002
0.20	0.00005
0.30	0.00011
0.40	0.00019
0.50	0.00030
0.60	0.00046
0.70	0.00076
0.80	0.00134
0.90	0.00299
1.00	0.13873







Summary Statistics	
Run Number	citya2–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	6678

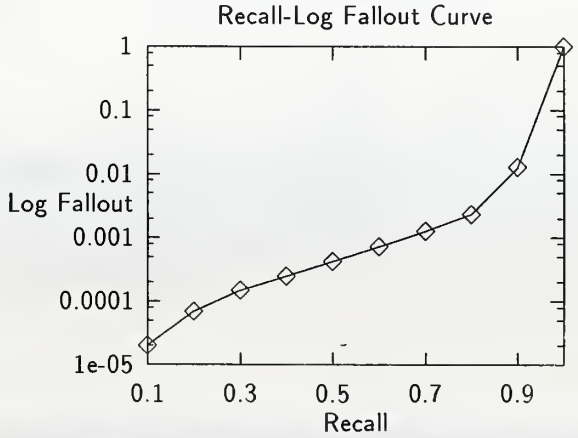
Recall Level Precision Averages	
Recall	Precision
0.00	0.8749
0.10	0.6799
0.20	0.5556
0.30	0.4621
0.40	0.4034
0.50	0.3265
0.60	0.2585
0.70	0.1888
0.80	0.1240
0.90	0.0282
1.00	0.0000

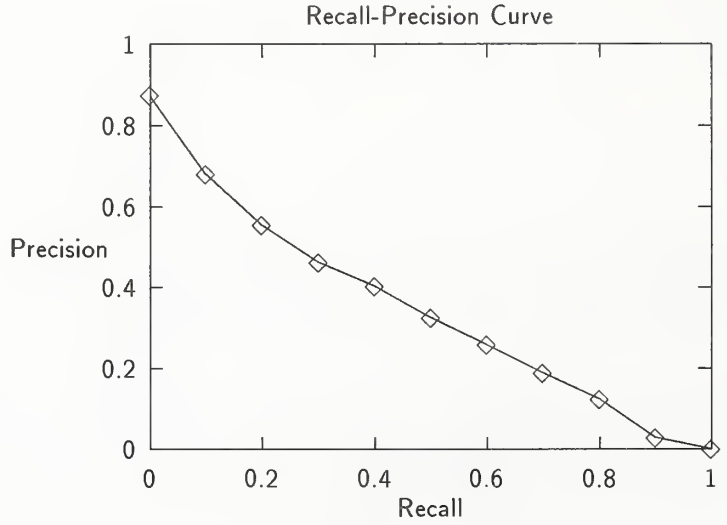
Average precision over all relevant docs	
non-interpolated	0.3373

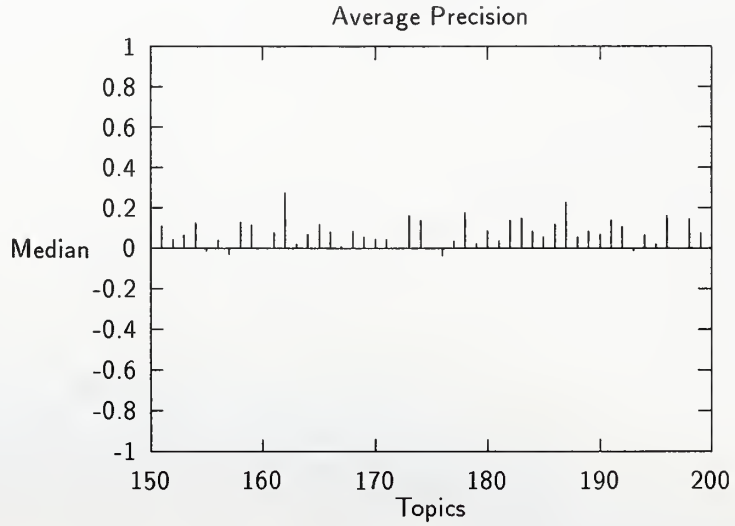
Document Level Averages	
	Precision
At 5 docs	0.7320
At 10 docs	0.6940
At 15 docs	0.6560
At 20 docs	0.6280
At 30 docs	0.5900
At 100 docs	0.4312
At 200 docs	0.3347
At 500 docs	0.2092
At 1000 docs	0.1336

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3815

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00002
0.20	0.00007
0.30	0.00015
0.40	0.00025
0.50	0.00043
0.60	0.00072
0.70	0.00126
0.80	0.00236
0.90	0.01295
1.00	1.00000







Summary Statistics	
Run Number	CLARTA–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	5848

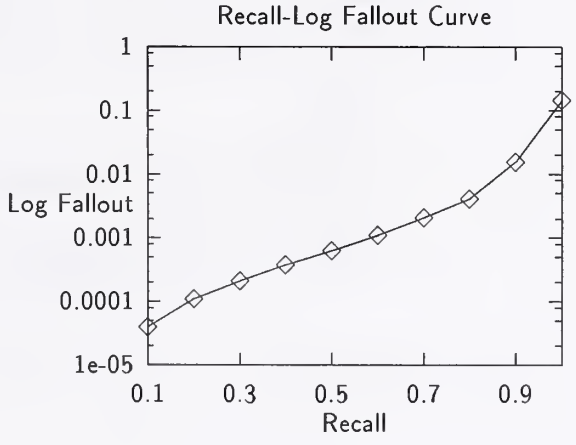
Recall Level Precision Averages	
Recall	Precision
0.00	0.7213
0.10	0.5276
0.20	0.4393
0.30	0.3780
0.40	0.3079
0.50	0.2500
0.60	0.1840
0.70	0.1220
0.80	0.0762
0.90	0.0239
1.00	0.0029

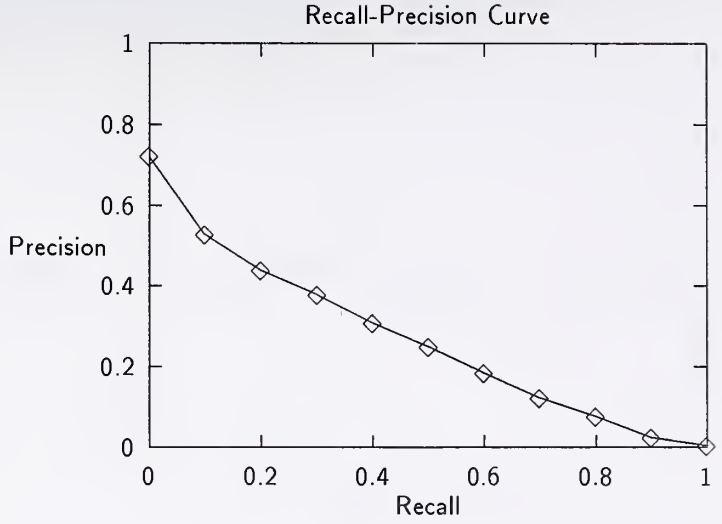
Average precision over all relevant docs	
non-interpolated	0.2581

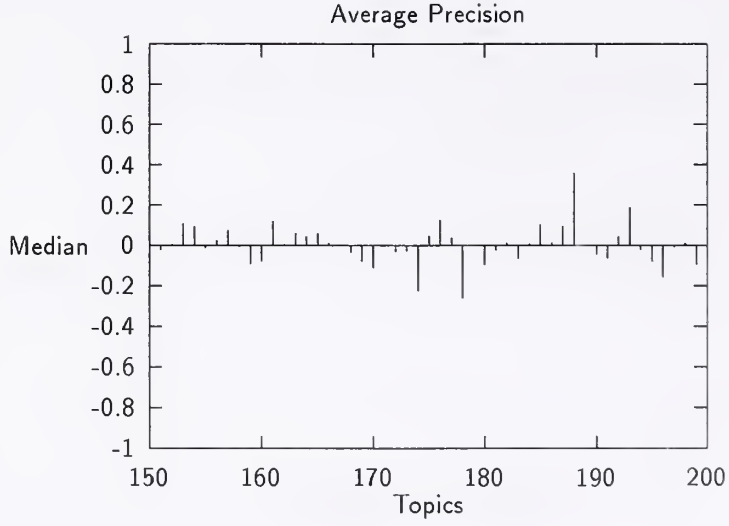
Document Level Averages	
	Precision
At 5 docs	0.5320
At 10 docs	0.5300
At 15 docs	0.5053
At 20 docs	0.4890
At 30 docs	0.4700
At 100 docs	0.3670
At 200 docs	0.2913
At 500 docs	0.1820
At 1000 docs	0.1170

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3184

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00004
0.20	0.00011
0.30	0.00021
0.40	0.00038
0.50	0.00063
0.60	0.00111
0.70	0.00210
0.80	0.00405
0.90	0.01534
1.00	0.14353







Summary Statistics	
Run Number	CLARTM–category A, manual
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	5671

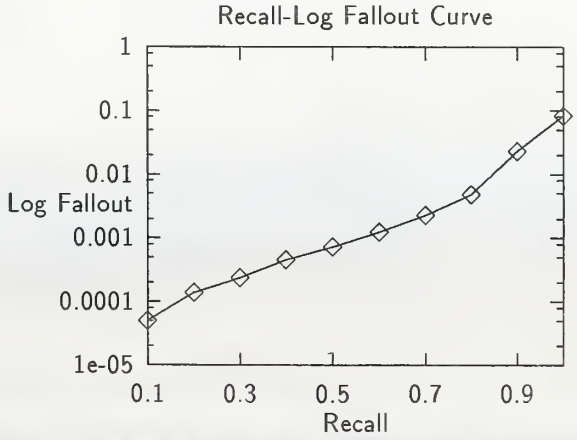
Recall Level Precision Averages	
Recall	Precision
0.00	0.6699
0.10	0.4543
0.20	0.3819
0.30	0.3398
0.40	0.2648
0.50	0.2248
0.60	0.1682
0.70	0.1147
0.80	0.0656
0.90	0.0163
1.00	0.0050

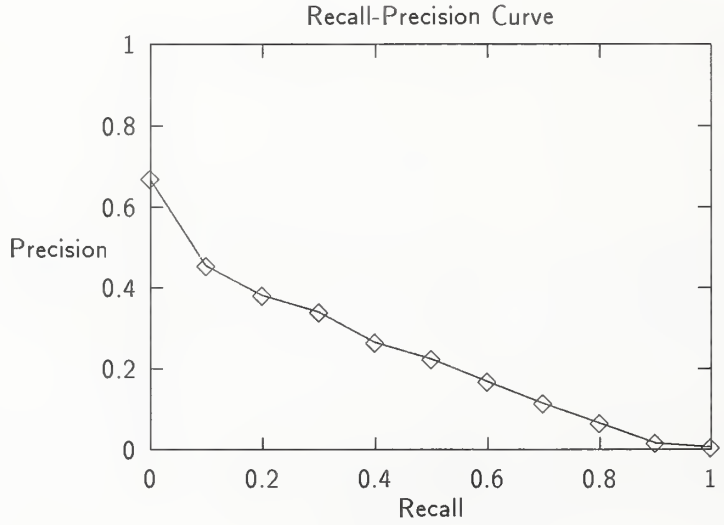
Average precision over all relevant docs	
non-interpolated	0.2249

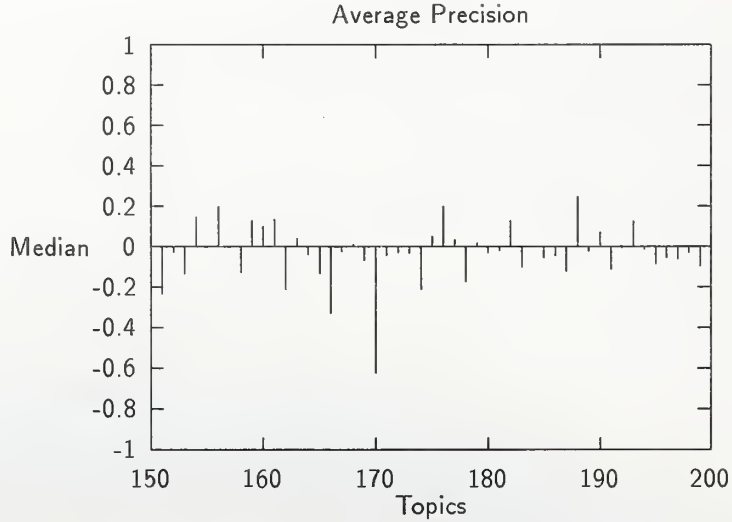
Document Level Averages	
	Precision
At 5 docs	0.4600
At 10 docs	0.4520
At 15 docs	0.4413
At 20 docs	0.4290
At 30 docs	0.4087
At 100 docs	0.3474
At 200 docs	0.2763
At 500 docs	0.1780
At 1000 docs	0.1134

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2841

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00005
0.20	0.00014
0.30	0.00024
0.40	0.00046
0.50	0.00072
0.60	0.00124
0.70	0.00226
0.80	0.00476
0.90	0.02267
1.00	0.08307







Summary Statistics	
Run Number	CrnlEA–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	7267

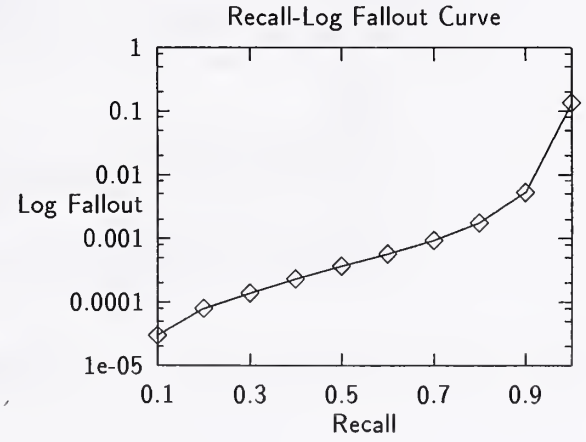
Recall Level Precision Averages	
Recall	Precision
0.00	0.7759
0.10	0.5910
0.20	0.5244
0.30	0.4735
0.40	0.4213
0.50	0.3595
0.60	0.3031
0.70	0.2365
0.80	0.1605
0.90	0.0664
1.00	0.0031

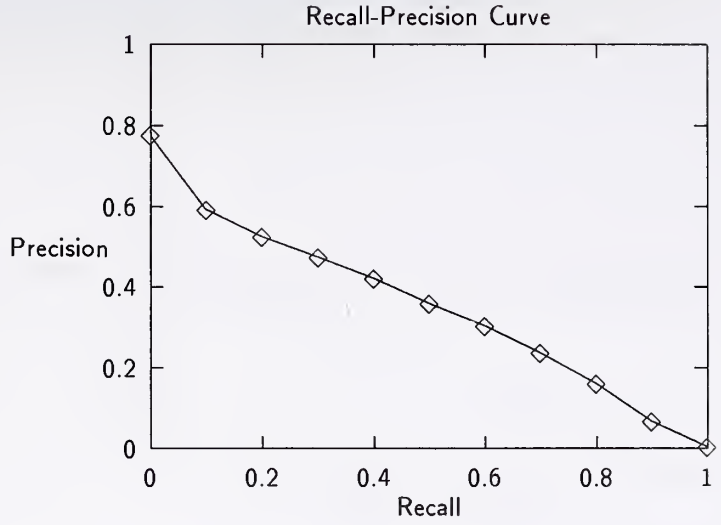
Average precision over all relevant docs	
non-interpolated	0.3419

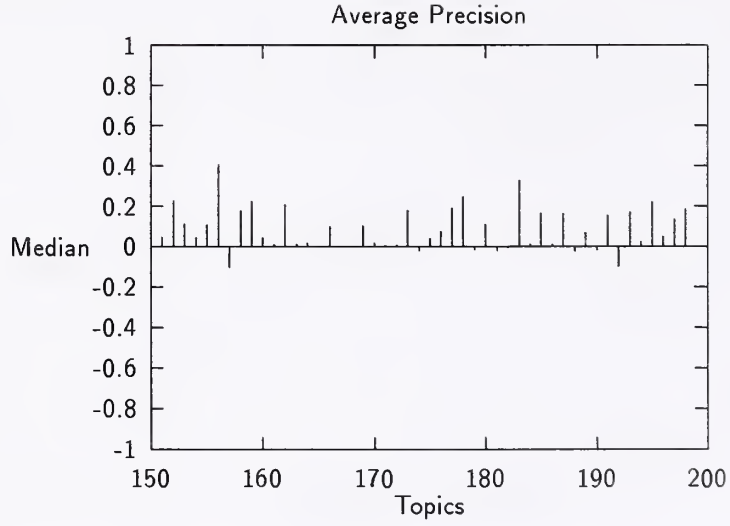
Document Level Averages	
	Precision
At 5 docs	0.5760
At 10 docs	0.5540
At 15 docs	0.5480
At 20 docs	0.5370
At 30 docs	0.5187
At 100 docs	0.4168
At 200 docs	0.3413
At 500 docs	0.2254
At 1000 docs	0.1453

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3890

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00008
0.30	0.00014
0.40	0.00023
0.50	0.00037
0.60	0.00058
0.70	0.00094
0.80	0.00175
0.90	0.00528
1.00	0.13424







Summary Statistics	
Run Number	CrnlLA–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	6808

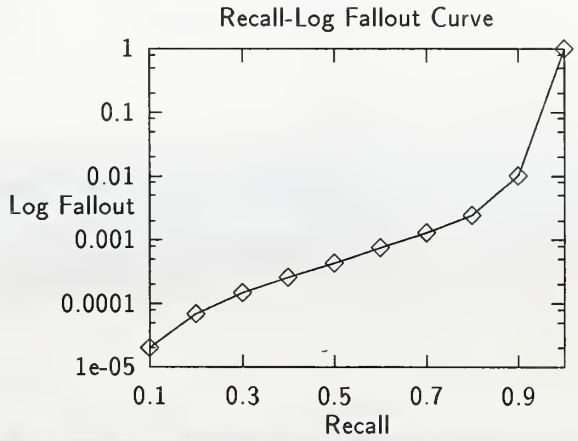
Recall Level Precision Averages	
Recall	Precision
0.00	0.8991
0.10	0.6496
0.20	0.5547
0.30	0.4602
0.40	0.3891
0.50	0.3241
0.60	0.2444
0.70	0.1832
0.80	0.1212
0.90	0.0361
1.00	0.0000

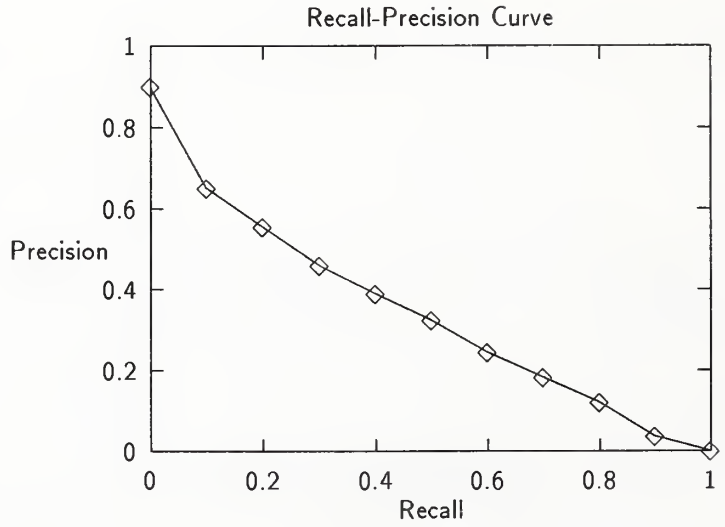
Average precision over all relevant docs	
non-interpolated	0.3302

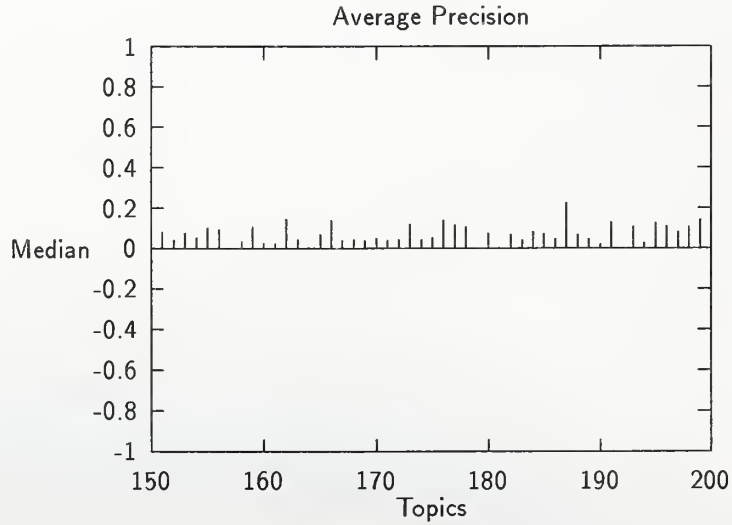
Document Level Averages	
	Precision
At 5 docs	0.6800
At 10 docs	0.6360
At 15 docs	0.6160
At 20 docs	0.5930
At 30 docs	0.5553
At 100 docs	0.4216
At 200 docs	0.3282
At 500 docs	0.2124
At 1000 docs	0.1362

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3768

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00002
0.20	0.00007
0.30	0.00015
0.40	0.00026
0.50	0.00044
0.60	0.00077
0.70	0.00130
0.80	0.00242
0.90	0.01003
1.00	1.00000







Summary Statistics	
Run Number	dortD1–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	6205

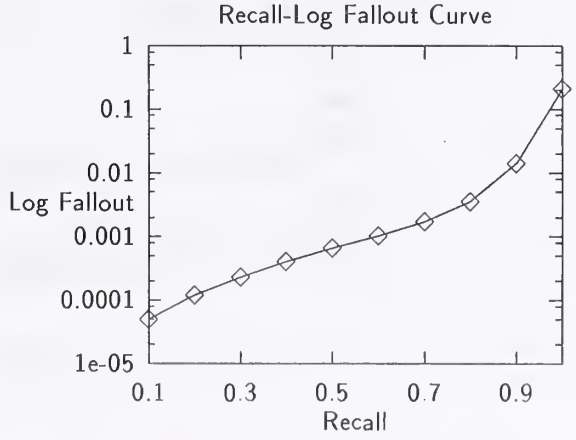
Recall Level Precision Averages	
Recall	Precision
0.00	0.6725
0.10	0.4804
0.20	0.4064
0.30	0.3484
0.40	0.2916
0.50	0.2412
0.60	0.1946
0.70	0.1450
0.80	0.0859
0.90	0.0261
1.00	0.0020

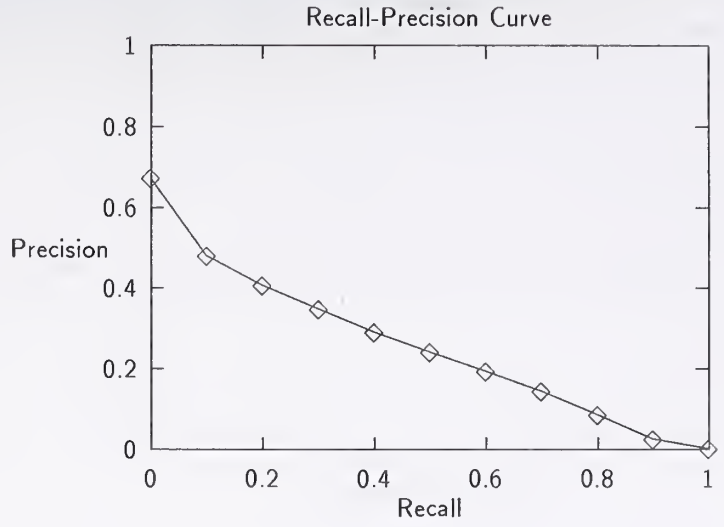
Average precision over all relevant docs	
non-interpolated	0.2443

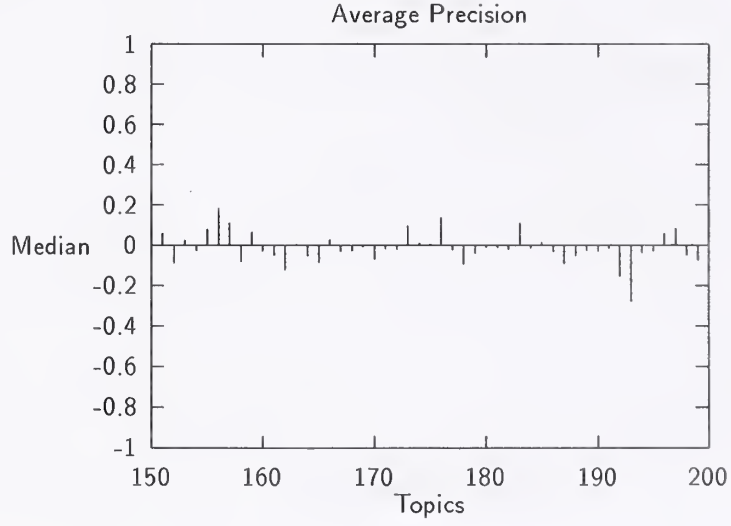
Document Level Averages	
	Precision
At 5 docs	0.4600
At 10 docs	0.4500
At 15 docs	0.4360
At 20 docs	0.4300
At 30 docs	0.4147
At 100 docs	0.3358
At 200 docs	0.2712
At 500 docs	0.1820
At 1000 docs	0.1241

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3128

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00005
0.20	0.00012
0.30	0.00023
0.40	0.00041
0.50	0.00066
0.60	0.00104
0.70	0.00172
0.80	0.00355
0.90	0.01402
1.00	0.20831







Summary Statistics	
Run Number	dortD2–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	6315

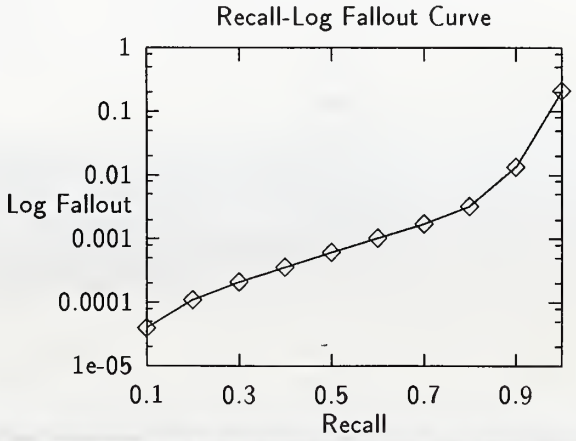
Recall Level Precision Averages	
Recall	Precision
0.00	0.6934
0.10	0.5181
0.20	0.4394
0.30	0.3712
0.40	0.3193
0.50	0.2504
0.60	0.1935
0.70	0.1443
0.80	0.0927
0.90	0.0273
1.00	0.0020

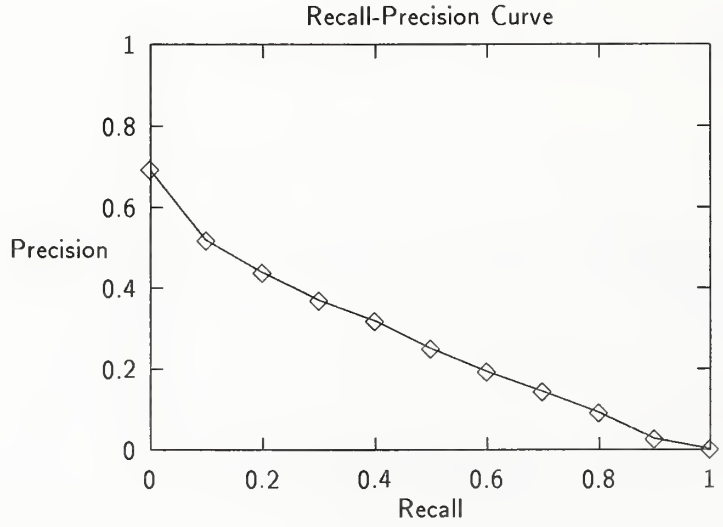
Average precision over all relevant docs	
non-interpolated	0.2577

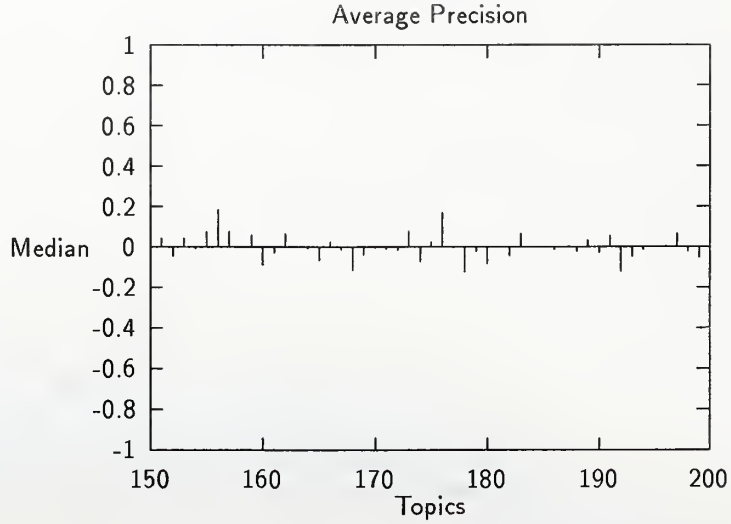
Document Level Averages	
	Precision
At 5 docs	0.4840
At 10 docs	0.4960
At 15 docs	0.4907
At 20 docs	0.4730
At 30 docs	0.4493
At 100 docs	0.3556
At 200 docs	0.2815
At 500 docs	0.1906
At 1000 docs	0.1263

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3221

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00004
0.20	0.00011
0.30	0.00021
0.40	0.00036
0.50	0.00062
0.60	0.00104
0.70	0.00173
0.80	0.00327
0.90	0.01339
1.00	0.20831







Summary Statistics	
Run Number	erima1–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	5325

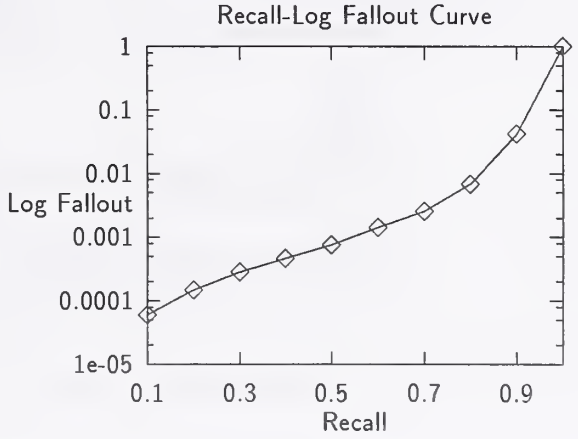
Recall Level Precision Averages	
Recall	Precision
0.00	0.6329
0.10	0.4101
0.20	0.3569
0.30	0.3031
0.40	0.2602
0.50	0.2152
0.60	0.1488
0.70	0.1026
0.80	0.0464
0.90	0.0089
1.00	0.0000

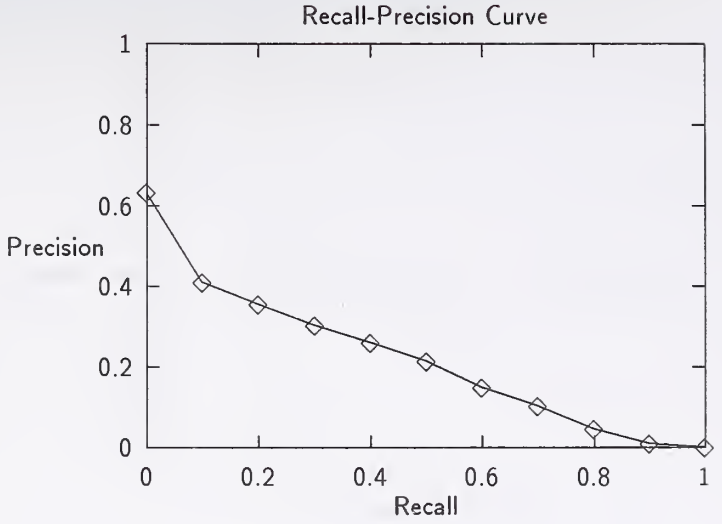
Average precision over all relevant docs	
non-interpolated	0.2061

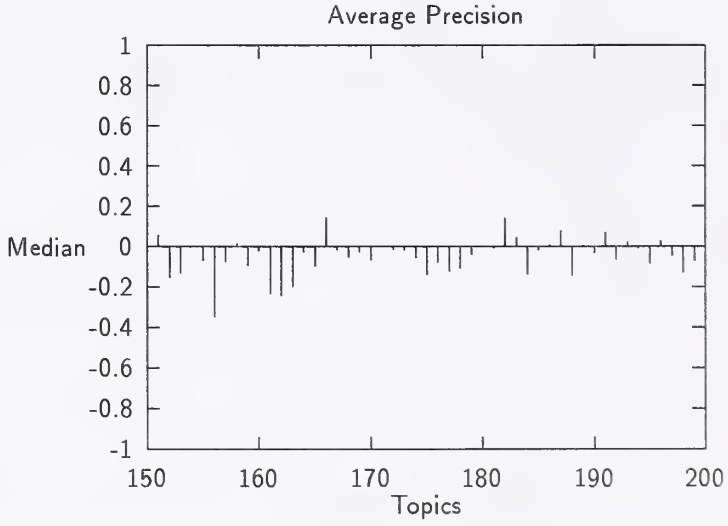
Document Level Averages	
	Precision
At 5 docs	0.4280
At 10 docs	0.4180
At 15 docs	0.4013
At 20 docs	0.3970
At 30 docs	0.3640
At 100 docs	0.2884
At 200 docs	0.2321
At 500 docs	0.1574
At 1000 docs	0.1065

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2668

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00006
0.20	0.00015
0.30	0.00029
0.40	0.00047
0.50	0.00076
0.60	0.00143
0.70	0.00256
0.80	0.00686
0.90	0.04184
1.00	1.00000







Summary Statistics	
Run Number	ETH001–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	6202

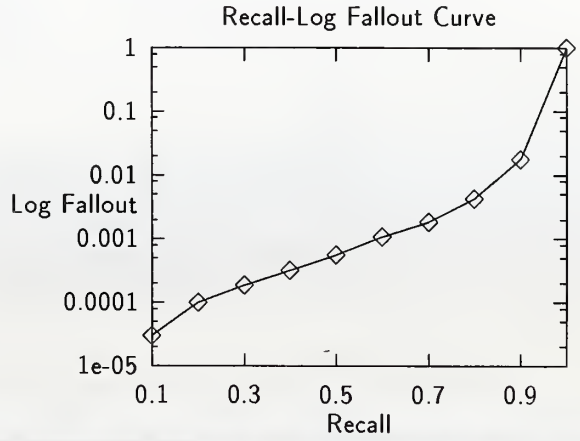
Recall Level Precision Averages	
Recall	Precision
0.00	0.7914
0.10	0.5677
0.20	0.4672
0.30	0.3980
0.40	0.3410
0.50	0.2691
0.60	0.1892
0.70	0.1351
0.80	0.0723
0.90	0.0206
1.00	0.0000

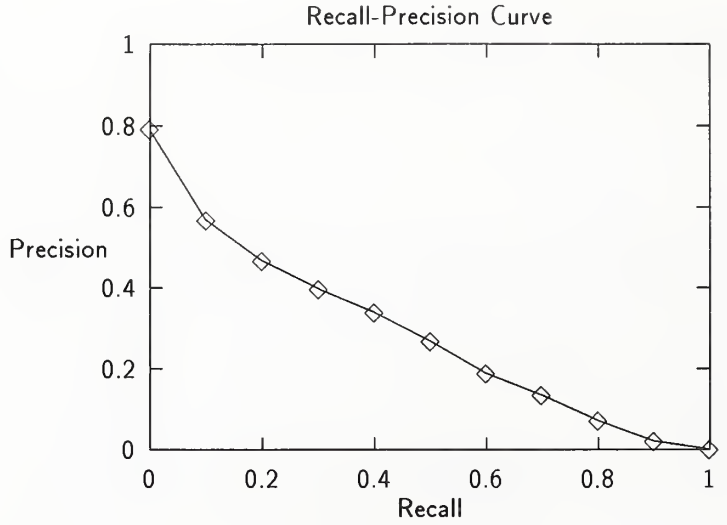
Average precision over all relevant docs	
non-interpolated	0.2737

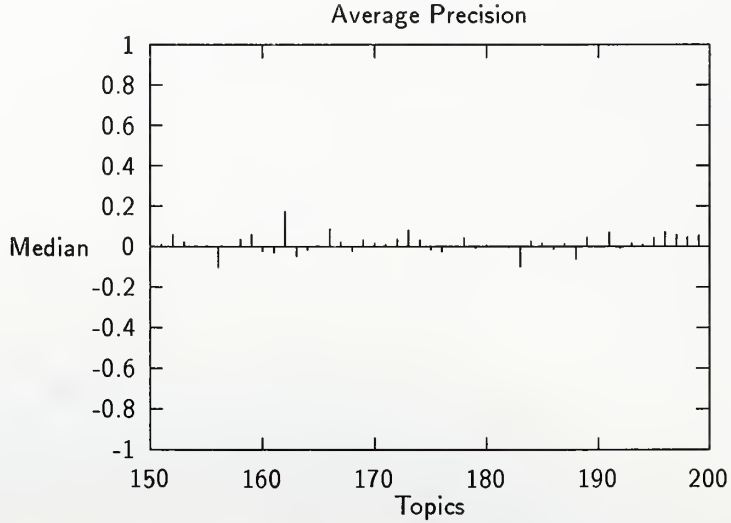
Document Level Averages	
	Precision
At 5 docs	0.5600
At 10 docs	0.5660
At 15 docs	0.5413
At 20 docs	0.5200
At 30 docs	0.4947
At 100 docs	0.3676
At 200 docs	0.2937
At 500 docs	0.1893
At 1000 docs	0.1240

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3374

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00010
0.30	0.00019
0.40	0.00032
0.50	0.00057
0.60	0.00107
0.70	0.00187
0.80	0.00429
0.90	0.01786
1.00	1.00000







Summary Statistics	
Run Number	ETH002–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	6237

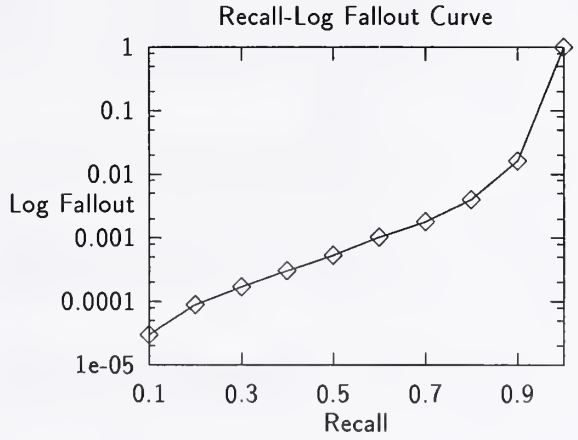
Recall Level Precision Averages	
Recall	Precision
0.00	0.8327
0.10	0.6098
0.20	0.4949
0.30	0.4181
0.40	0.3525
0.50	0.2771
0.60	0.1955
0.70	0.1392
0.80	0.0771
0.90	0.0227
1.00	0.0000

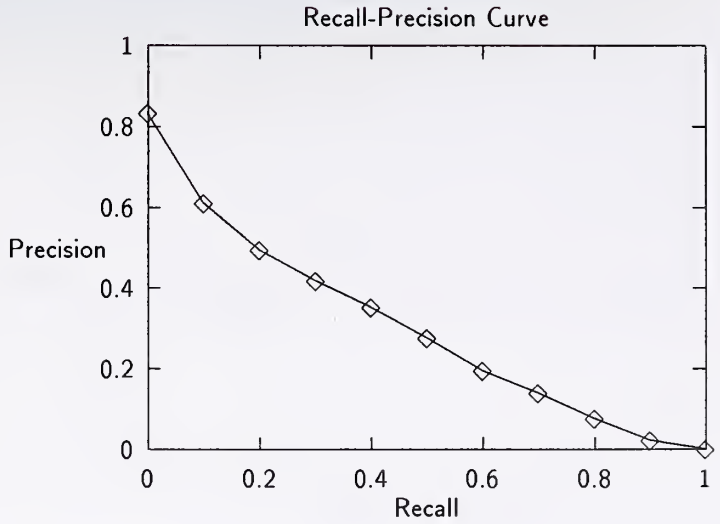
Average precision over all relevant docs	
non-interpolated	0.2916

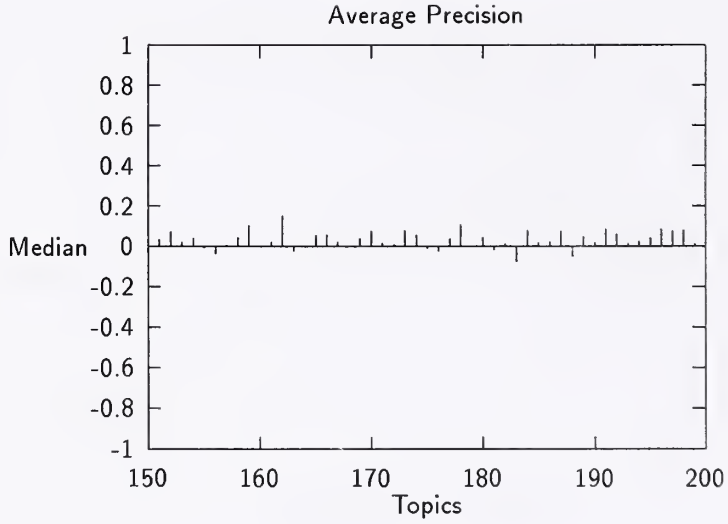
Document Level Averages	
	Precision
At 5 docs	0.6880
At 10 docs	0.6360
At 15 docs	0.5973
At 20 docs	0.5810
At 30 docs	0.5393
At 100 docs	0.3954
At 200 docs	0.3059
At 500 docs	0.1927
At 1000 docs	0.1247

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3475

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00009
0.30	0.00017
0.40	0.00031
0.50	0.00054
0.60	0.00103
0.70	0.00181
0.80	0.00400
0.90	0.01618
1.00	1.00000







Summary Statistics	
Run Number	INQ101–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	6895

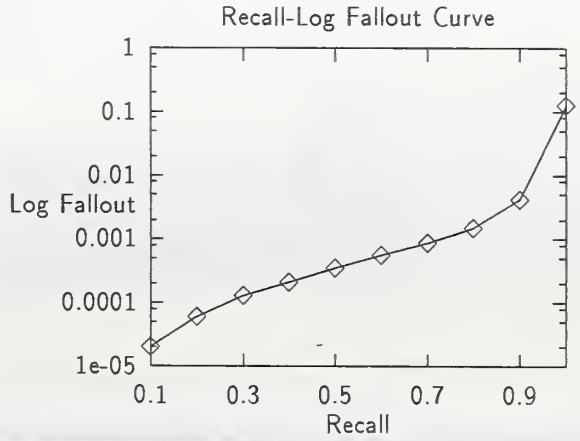
Recall Level Precision Averages	
Recall	Precision
0.00	0.8381
0.10	0.6370
0.20	0.5654
0.30	0.4994
0.40	0.4438
0.50	0.3705
0.60	0.3093
0.70	0.2519
0.80	0.1818
0.90	0.0824
1.00	0.0033

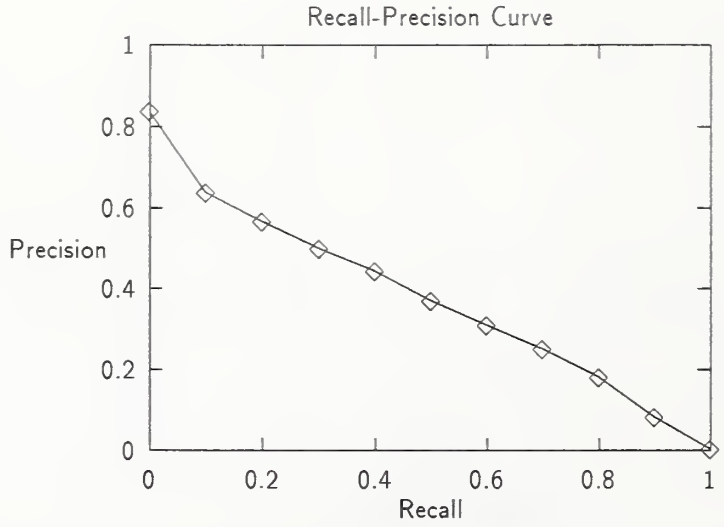
Average precision over all relevant docs	
non-interpolated	0.3659

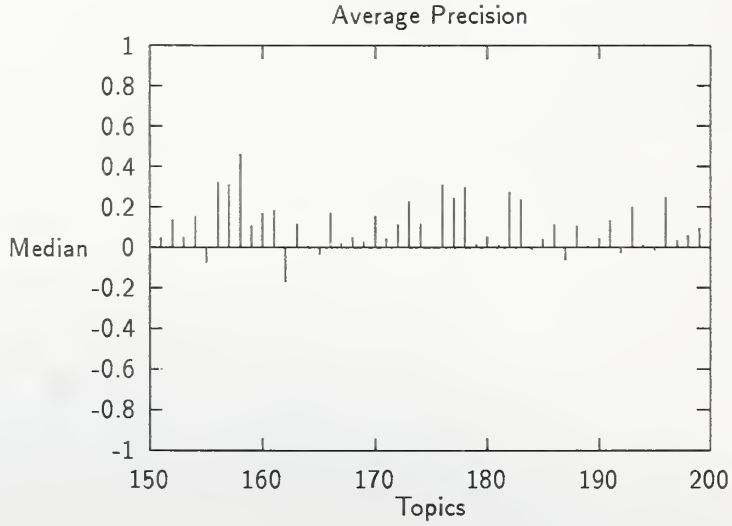
Document Level Averages	
	Precision
At 5 docs	0.6440
At 10 docs	0.6580
At 15 docs	0.6213
At 20 docs	0.5920
At 30 docs	0.5647
At 100 docs	0.4398
At 200 docs	0.3525
At 500 docs	0.2212
At 1000 docs	0.1379

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4088

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00002
0.20	0.00006
0.30	0.00013
0.40	0.00021
0.50	0.00035
0.60	0.00056
0.70	0.00087
0.80	0.00150
0.90	0.00418
1.00	0.12608







Summary Statistics	
Run Number	INQ102–category A, manual
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	7305

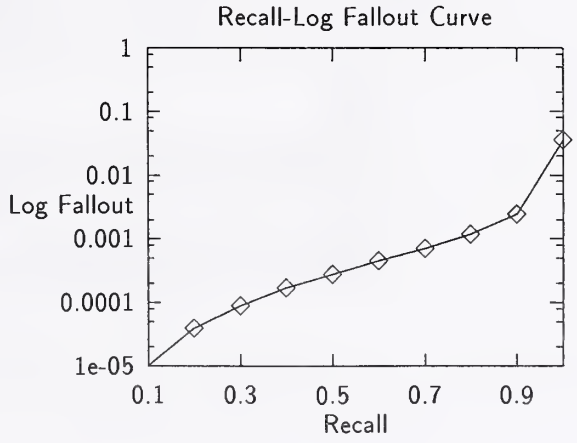
Recall Level Precision Averages	
Recall	Precision
0.00	0.8992
0.10	0.7514
0.20	0.6584
0.30	0.5724
0.40	0.4982
0.50	0.4272
0.60	0.3521
0.70	0.2915
0.80	0.2173
0.90	0.1336
1.00	0.0115

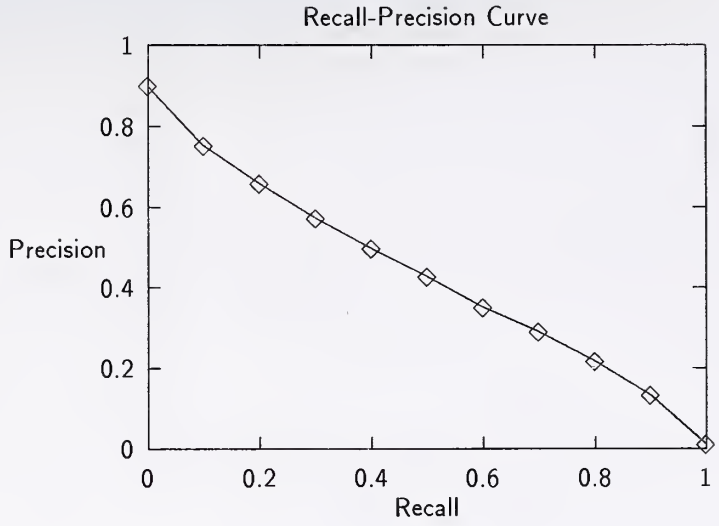
Average precision over all relevant docs	
non-interpolated	0.4226

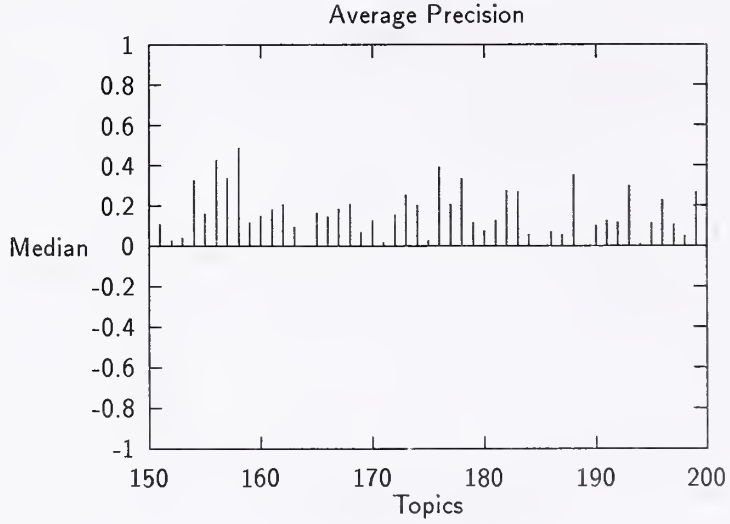
Document Level Averages	
	Precision
At 5 docs	0.7440
At 10 docs	0.7220
At 15 docs	0.6867
At 20 docs	0.6740
At 30 docs	0.6267
At 100 docs	0.4902
At 200 docs	0.3848
At 500 docs	0.2401
At 1000 docs	0.1461

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4524

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00001
0.20	0.00004
0.30	0.00009
0.40	0.00017
0.50	0.00028
0.60	0.00046
0.70	0.00071
0.80	0.00120
0.90	0.00244
1.00	0.03588







Summary Statistics	
Run Number	lsia0mf–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	6045

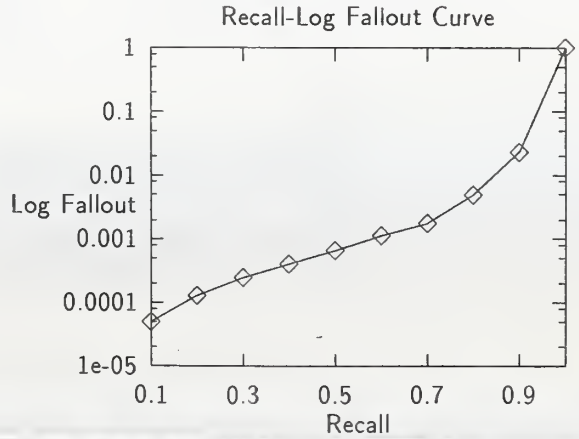
Recall Level Precision Averages	
Recall	Precision
0.00	0.6680
0.10	0.4571
0.20	0.3877
0.30	0.3373
0.40	0.2873
0.50	0.2398
0.60	0.1801
0.70	0.1409
0.80	0.0634
0.90	0.0161
1.00	0.0000

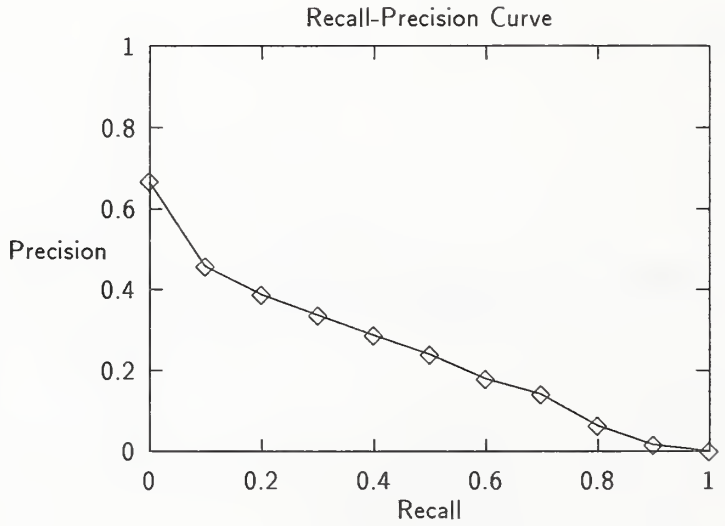
Average precision over all relevant docs	
non-interpolated	0.2325

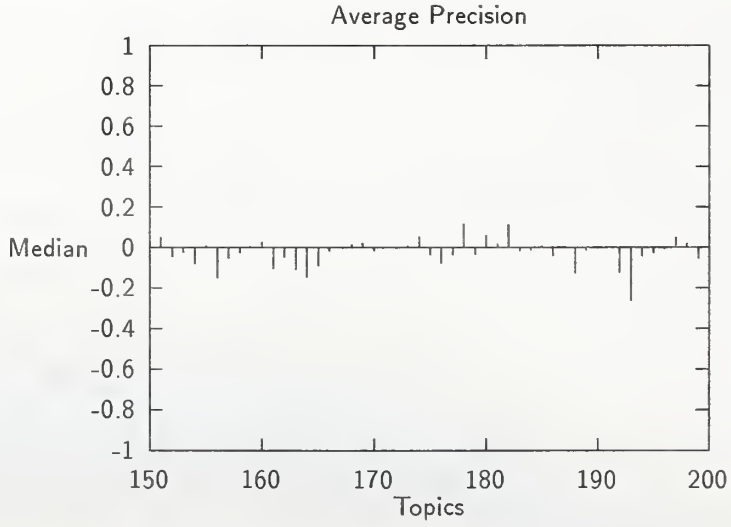
Document Level Averages	
	Precision
At 5 docs	0.4160
At 10 docs	0.4340
At 15 docs	0.4160
At 20 docs	0.4090
At 30 docs	0.3913
At 100 docs	0.3130
At 200 docs	0.2586
At 500 docs	0.1789
At 1000 docs	0.1209

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3030

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00005
0.20	0.00013
0.30	0.00025
0.40	0.00041
0.50	0.00066
0.60	0.00114
0.70	0.00178
0.80	0.00493
0.90	0.02296
1.00	1.00000







Summary Statistics	
Run Number	lsia0mw20f–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	6090

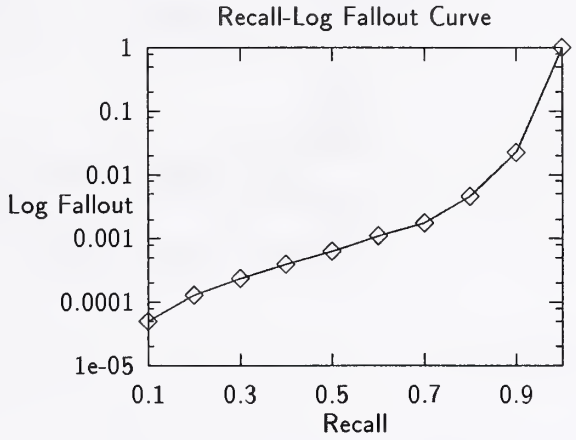
Recall Level Precision Averages	
Recall	Precision
0.00	0.6872
0.10	0.4732
0.20	0.3997
0.30	0.3452
0.40	0.2933
0.50	0.2486
0.60	0.1821
0.70	0.1422
0.80	0.0680
0.90	0.0164
1.00	0.0000

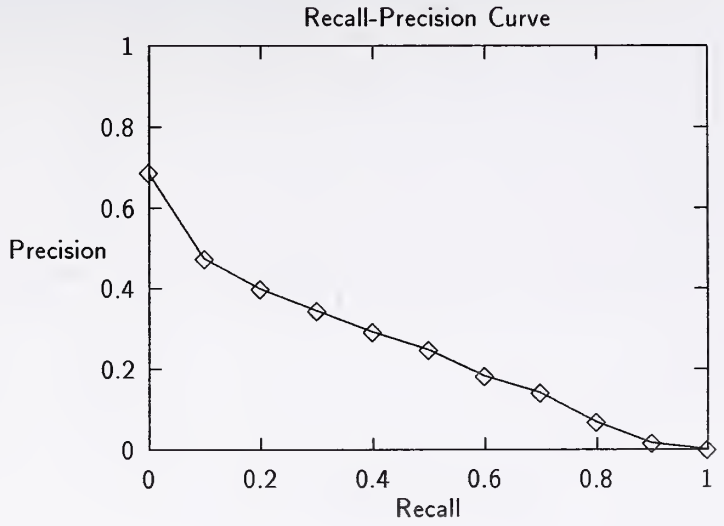
Average precision over all relevant docs	
non-interpolated	0.2393

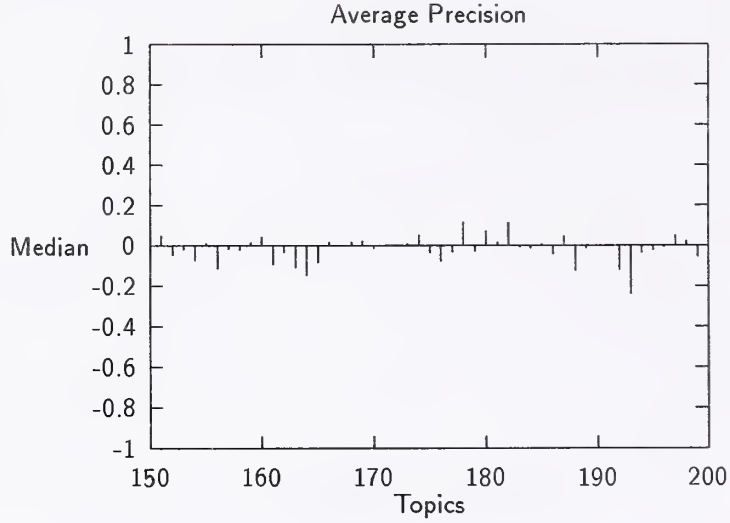
Document Level Averages	
	Precision
At 5 docs	0.4480
At 10 docs	0.4520
At 15 docs	0.4347
At 20 docs	0.4270
At 30 docs	0.4040
At 100 docs	0.3246
At 200 docs	0.2659
At 500 docs	0.1811
At 1000 docs	0.1218

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3071

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00005
0.20	0.00013
0.30	0.00024
0.40	0.00040
0.50	0.00063
0.60	0.00112
0.70	0.00176
0.80	0.00458
0.90	0.02253
1.00	1.00000







Summary Statistics	
Run Number	nyuir1–category A, automatic
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	5978

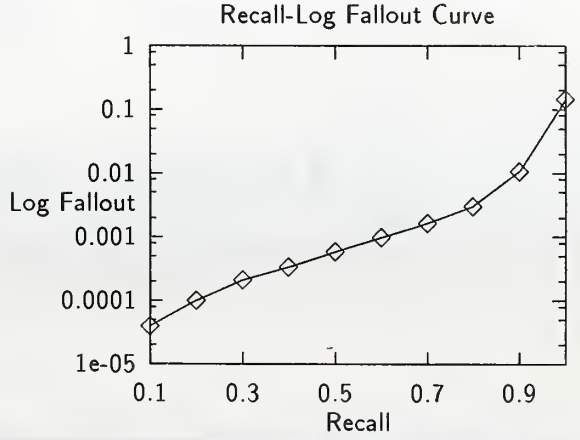
Recall Level Precision Averages	
Recall	Precision
0.00	0.7653
0.10	0.5420
0.20	0.4465
0.30	0.3763
0.40	0.3271
0.50	0.2608
0.60	0.2031
0.70	0.1522
0.80	0.0990
0.90	0.0339
1.00	0.0029

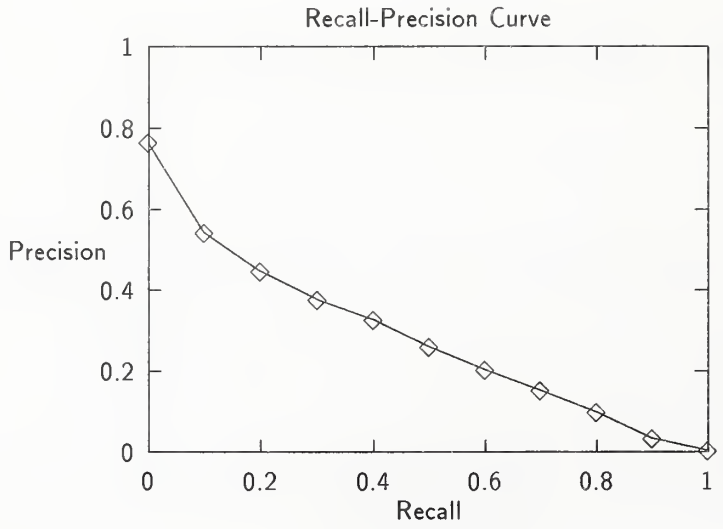
Average precision over all relevant docs	
non-interpolated	0.2722

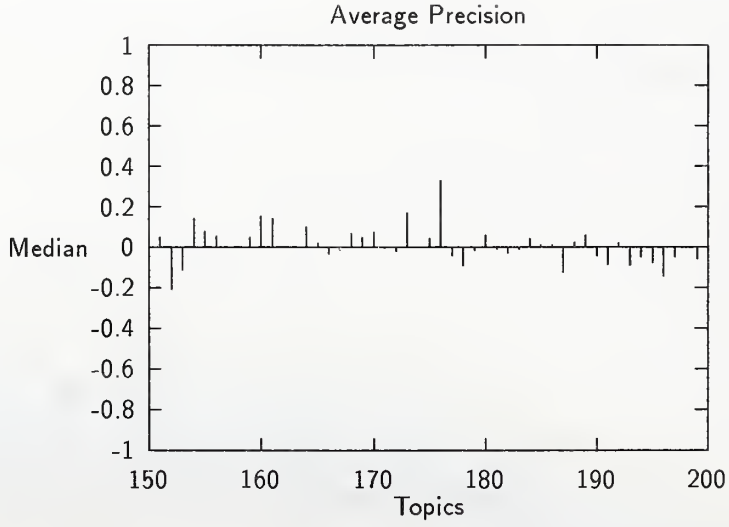
Document Level Averages	
	Precision
At 5 docs	0.5960
At 10 docs	0.5480
At 15 docs	0.5280
At 20 docs	0.5060
At 30 docs	0.4793
At 100 docs	0.3650
At 200 docs	0.2902
At 500 docs	0.1832
At 1000 docs	0.1196

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3232

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00004
0.20	0.00010
0.30	0.00021
0.40	0.00034
0.50	0.00059
0.60	0.00098
0.70	0.00163
0.80	0.00304
0.90	0.01071
1.00	0.14353







Summary Statistics	
Run Number	nyuir2–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	5978

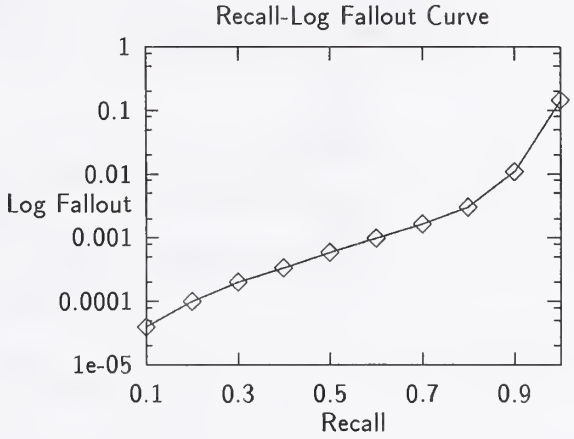
Recall Level Precision Averages	
Recall	Precision
0.00	0.7639
0.10	0.5429
0.20	0.4523
0.30	0.3814
0.40	0.3281
0.50	0.2613
0.60	0.2033
0.70	0.1519
0.80	0.0989
0.90	0.0332
1.00	0.0029

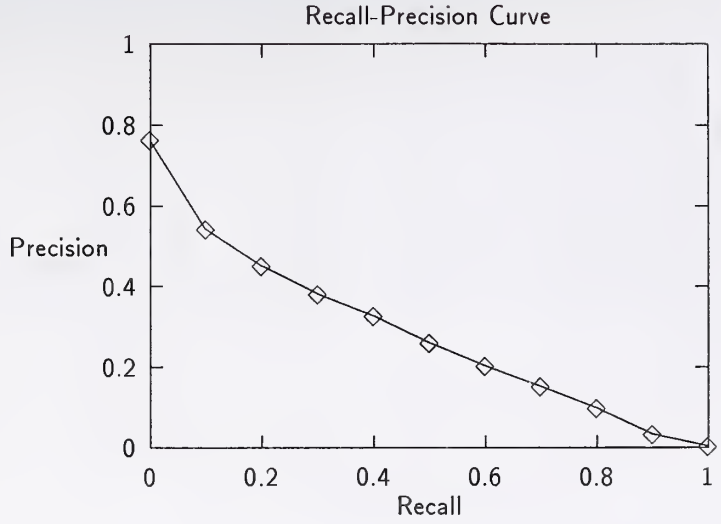
Average precision over all relevant docs	
non-interpolated	0.2735

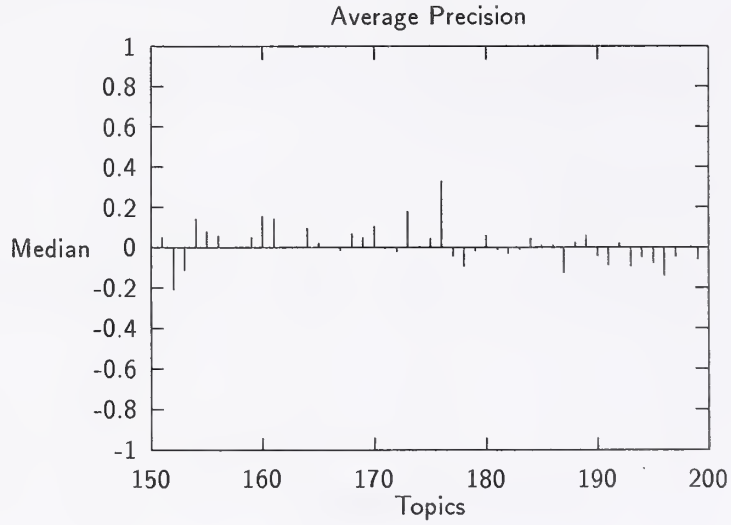
Document Level Averages	
	Precision
At 5 docs	0.5880
At 10 docs	0.5580
At 15 docs	0.5253
At 20 docs	0.5070
At 30 docs	0.4827
At 100 docs	0.3644
At 200 docs	0.2907
At 500 docs	0.1832
At 1000 docs	0.1196

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3231

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00004
0.20	0.00010
0.30	0.00020
0.40	0.00034
0.50	0.00059
0.60	0.00098
0.70	0.00163
0.80	0.00304
0.90	0.01094
1.00	0.14353







Summary Statistics	
Run Number	padre1–category A, manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	30240
Relevant:	9805
Rel_ret:	3285

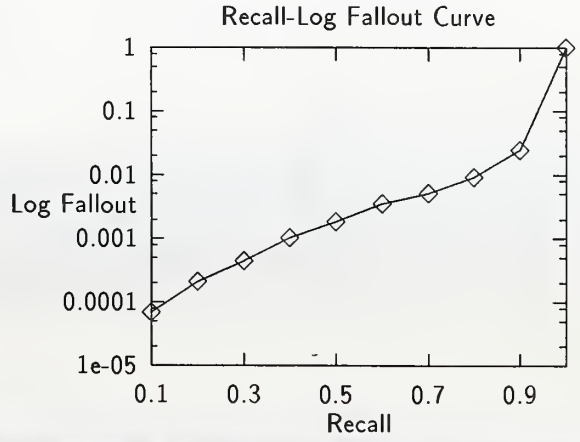
Recall Level Precision Averages	
Recall	Precision
0.00	0.6623
0.10	0.3787
0.20	0.2820
0.30	0.2171
0.40	0.1380
0.50	0.1026
0.60	0.0664
0.70	0.0541
0.80	0.0348
0.90	0.0152
1.00	0.0000

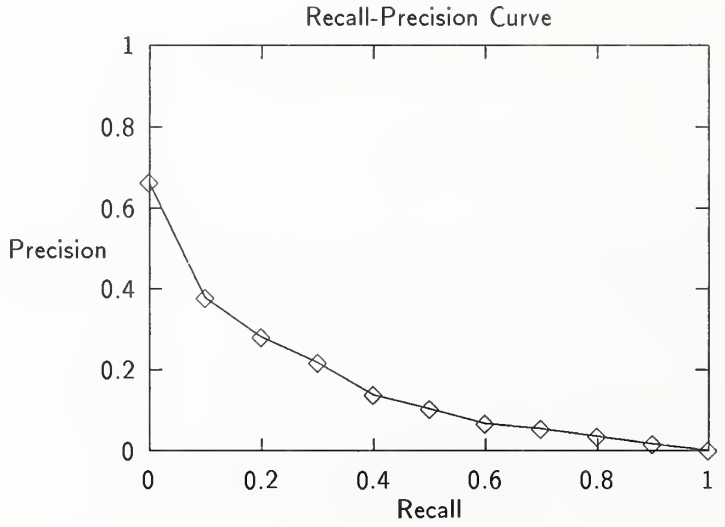
Average precision over all relevant docs	
non-interpolated	0.1448

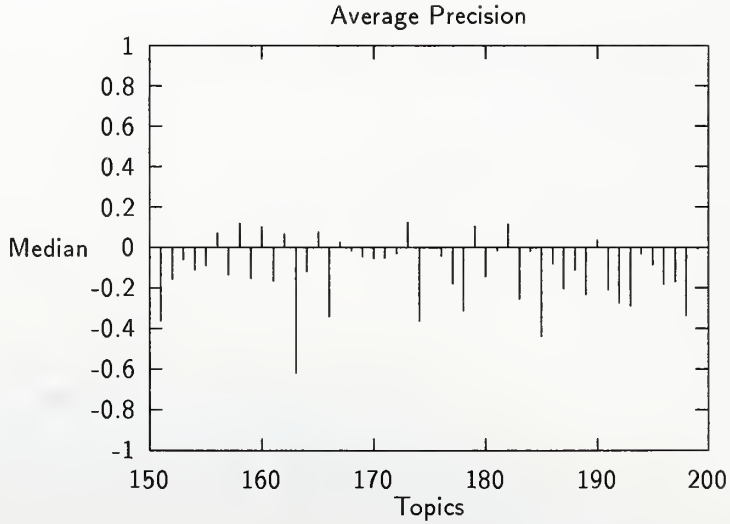
Document Level Averages	
	Precision
At 5 docs	0.4320
At 10 docs	0.3680
At 15 docs	0.3547
At 20 docs	0.3440
At 30 docs	0.3353
At 100 docs	0.2544
At 200 docs	0.1927
At 500 docs	0.1105
At 1000 docs	0.0657

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2177

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00007
0.20	0.00021
0.30	0.00045
0.40	0.00104
0.50	0.00183
0.60	0.00352
0.70	0.00511
0.80	0.00926
0.90	0.02434
1.00	1.00000







Summary Statistics	
Run Number	padre2–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	4759

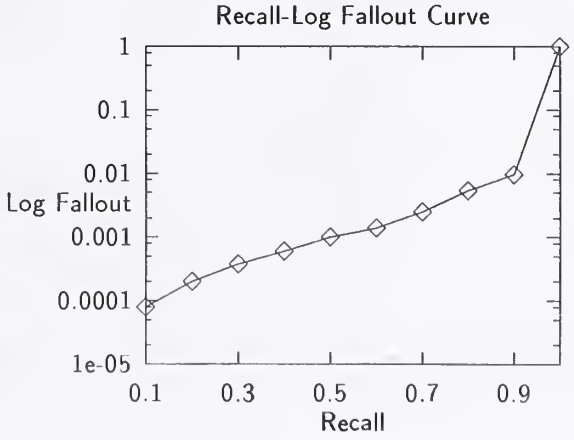
Recall Level Precision Averages	
Recall	Precision
0.00	0.5696
0.10	0.3388
0.20	0.2957
0.30	0.2469
0.40	0.2158
0.50	0.1731
0.60	0.1517
0.70	0.1034
0.80	0.0580
0.90	0.0373
1.00	0.0000

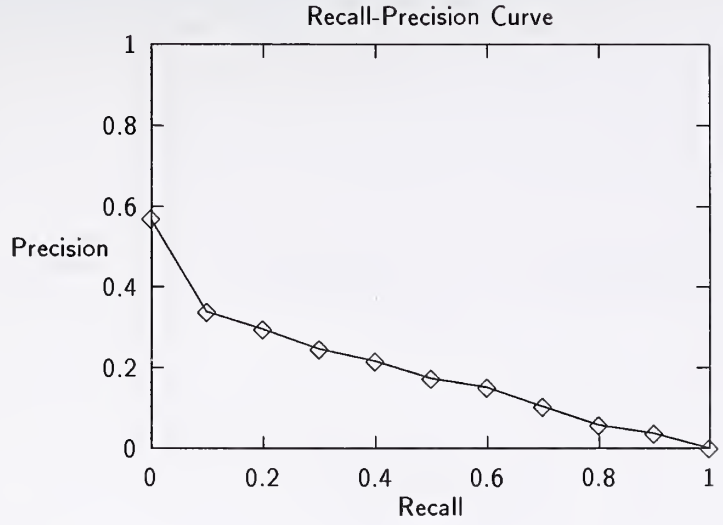
Average precision over all relevant docs	
non-interpolated	0.1752

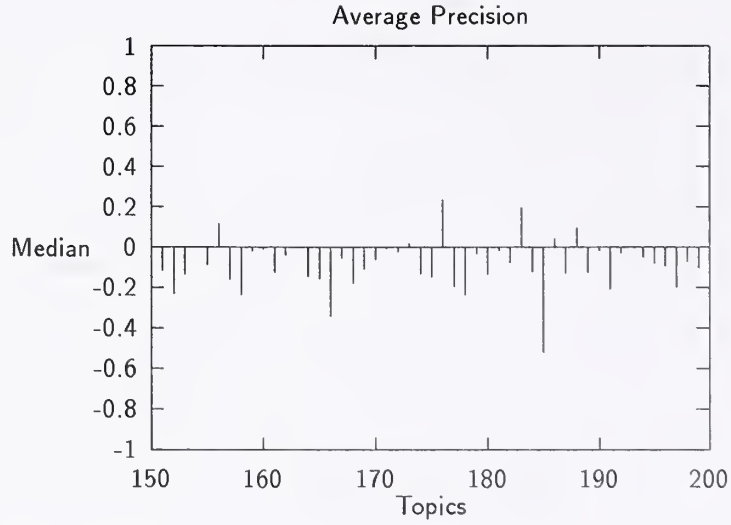
Document Level Averages	
	Precision
At 5 docs	0.3560
At 10 docs	0.3700
At 15 docs	0.3547
At 20 docs	0.3490
At 30 docs	0.3293
At 100 docs	0.2736
At 200 docs	0.2124
At 500 docs	0.1456
At 1000 docs	0.0952

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2279

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00008
0.20	0.00020
0.30	0.00038
0.40	0.00061
0.50	0.00100
0.60	0.00140
0.70	0.00253
0.80	0.00542
0.90	0.00970
1.00	1.00000







Summary Statistics	
Run Number	pircs1–category A, automatic
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	6017

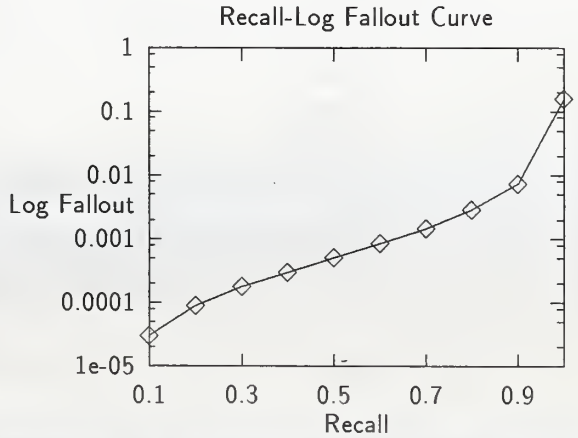
Recall Level Precision Averages	
Recall	Precision
0.00	0.8129
0.10	0.5917
0.20	0.4814
0.30	0.4133
0.40	0.3569
0.50	0.2921
0.60	0.2247
0.70	0.1673
0.80	0.1028
0.90	0.0488
1.00	0.0026

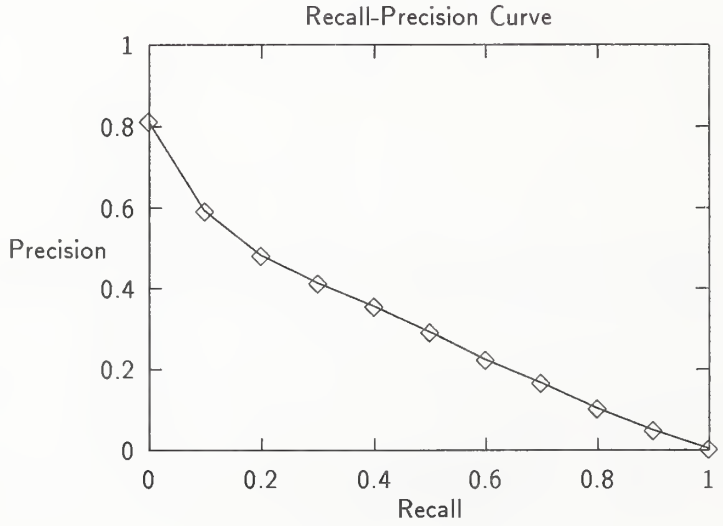
Average precision over all relevant docs	
non-interpolated	0.3001

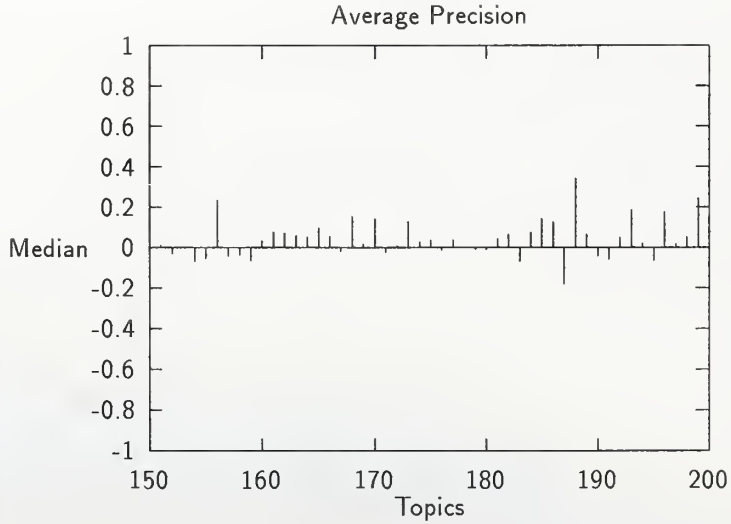
Document Level Averages	
	Precision
At 5 docs	0.6600
At 10 docs	0.6160
At 15 docs	0.5947
At 20 docs	0.5670
At 30 docs	0.5187
At 100 docs	0.3792
At 200 docs	0.2955
At 500 docs	0.1852
At 1000 docs	0.1203

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3484

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00009
0.30	0.00018
0.40	0.00030
0.50	0.00051
0.60	0.00086
0.70	0.00145
0.80	0.00291
0.90	0.00732
1.00	0.16014







Summary Statistics	
Run Number	pircs2–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	6145

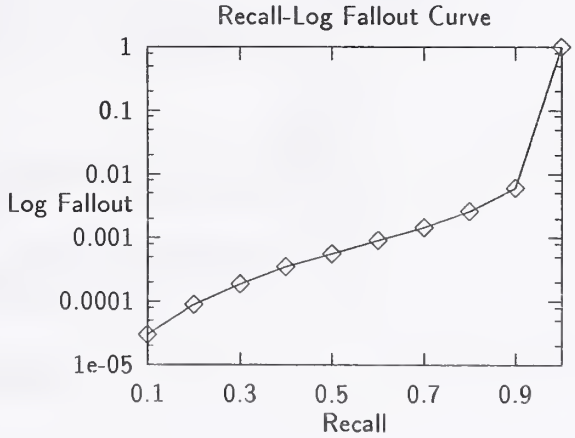
Recall Level Precision Averages	
Recall	Precision
0.00	0.8124
0.10	0.5809
0.20	0.4726
0.30	0.3956
0.40	0.3257
0.50	0.2675
0.60	0.2157
0.70	0.1667
0.80	0.1126
0.90	0.0586
1.00	0.0000

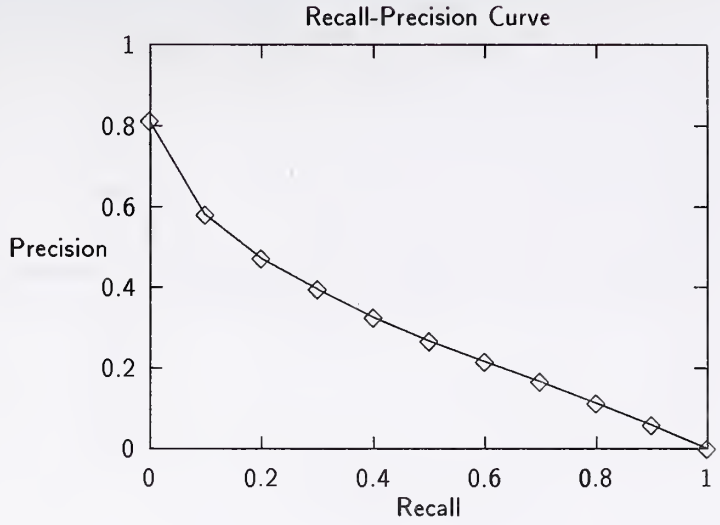
Average precision over all relevant docs	
non-interpolated	0.2913

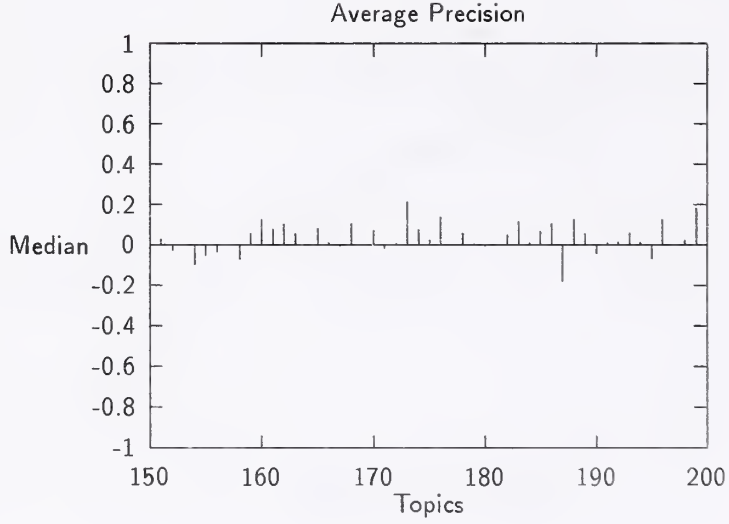
Document Level Averages	
	Precision
At 5 docs	0.6600
At 10 docs	0.6100
At 15 docs	0.5560
At 20 docs	0.5240
At 30 docs	0.4780
At 100 docs	0.3774
At 200 docs	0.2969
At 500 docs	0.1889
At 1000 docs	0.1229

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3402

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00009
0.30	0.00019
0.40	0.00035
0.50	0.00057
0.60	0.00091
0.70	0.00146
0.80	0.00263
0.90	0.00604
1.00	1.00000







Summary Statistics	
Run Number	rutfua1–category A, manual
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	5157

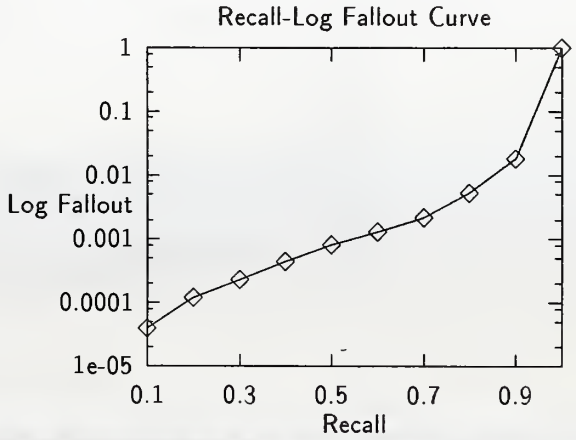
Recall Level Precision Averages	
Recall	Precision
0.00	0.7568
0.10	0.5334
0.20	0.4201
0.30	0.3529
0.40	0.2716
0.50	0.2052
0.60	0.1628
0.70	0.1186
0.80	0.0597
0.90	0.0205
1.00	0.0000

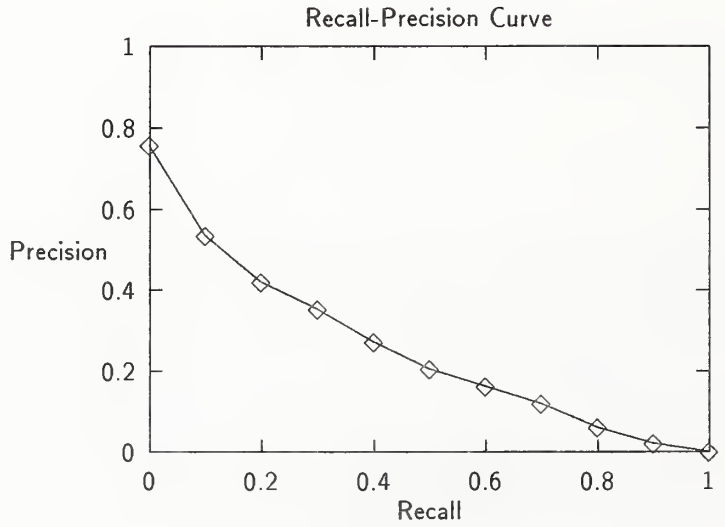
Average precision over all relevant docs	
non-interpolated	0.2393

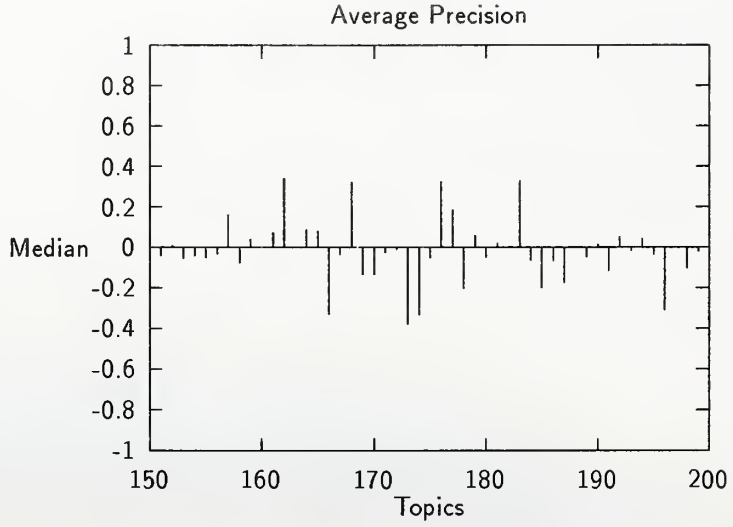
Document Level Averages	
	Precision
At 5 docs	0.5360
At 10 docs	0.5360
At 15 docs	0.5253
At 20 docs	0.5160
At 30 docs	0.4867
At 100 docs	0.3700
At 200 docs	0.2849
At 500 docs	0.1664
At 1000 docs	0.1031

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3163

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00004
0.20	0.00012
0.30	0.00023
0.40	0.00045
0.50	0.00081
0.60	0.00129
0.70	0.00217
0.80	0.00526
0.90	0.01795
1.00	1.00000







Summary Statistics	
Run Number	rutfua2–category A, manual
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	5183

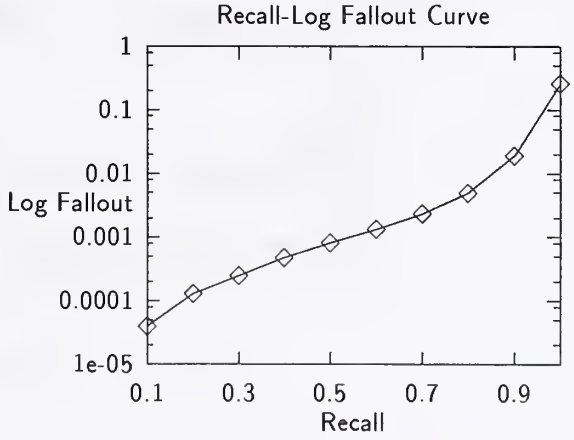
Recall Level Precision Averages	
Recall	Precision
0.00	0.7406
0.10	0.4852
0.20	0.3884
0.30	0.3310
0.40	0.2565
0.50	0.2032
0.60	0.1595
0.70	0.1106
0.80	0.0637
0.90	0.0193
1.00	0.0016

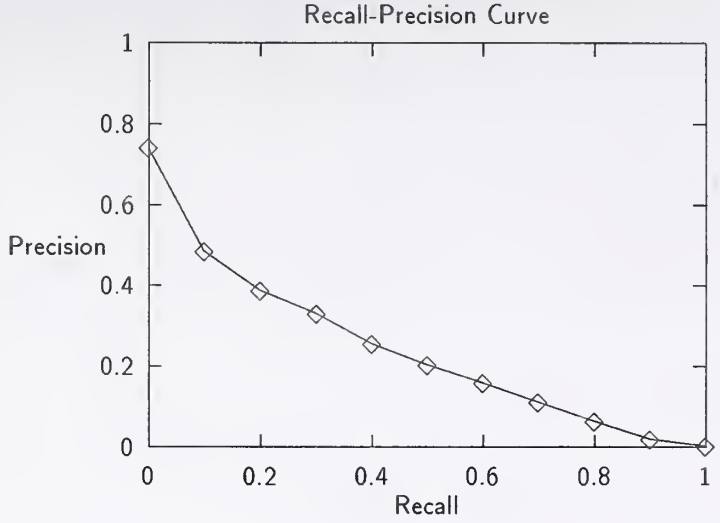
Average precision over all relevant docs	
non-interpolated	0.2254

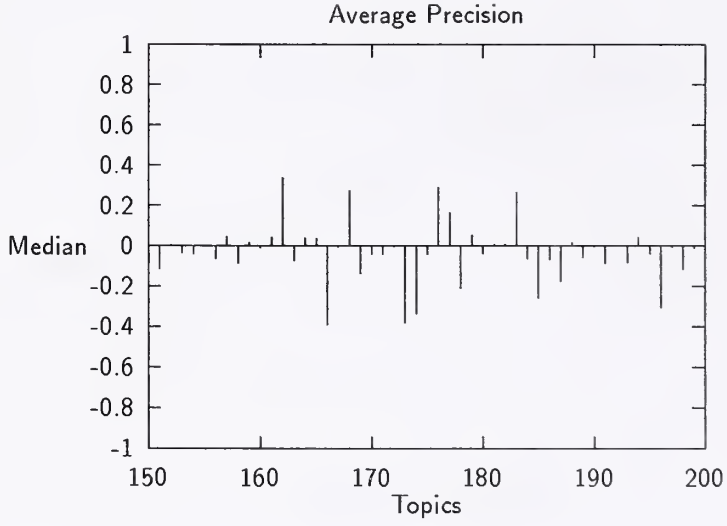
Document Level Averages	
	Precision
At 5 docs	0.5240
At 10 docs	0.4760
At 15 docs	0.4707
At 20 docs	0.4530
At 30 docs	0.4493
At 100 docs	0.3584
At 200 docs	0.2779
At 500 docs	0.1664
At 1000 docs	0.1037

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3091

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00004
0.20	0.00013
0.30	0.00025
0.40	0.00048
0.50	0.00082
0.60	0.00132
0.70	0.00235
0.80	0.00491
0.90	0.01909
1.00	0.26049







Summary Statistics	
Run Number	siems1–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	5381

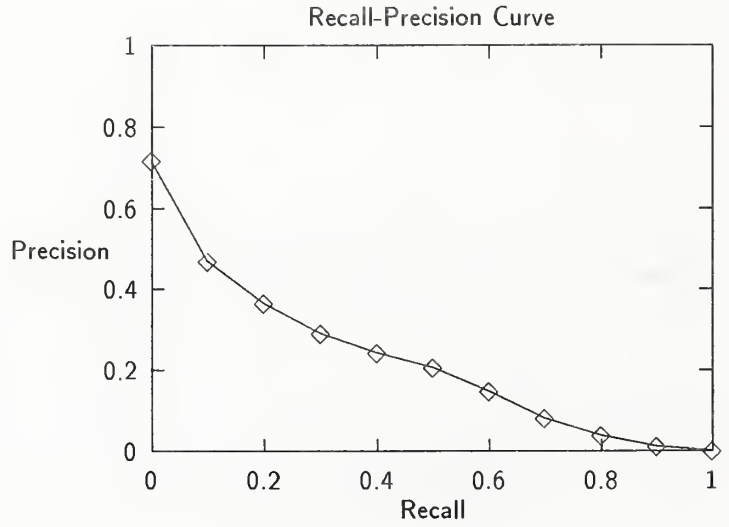
Recall Level Precision Averages	
Recall	Precision
0.00	0.7168
0.10	0.4697
0.20	0.3659
0.30	0.2915
0.40	0.2434
0.50	0.2060
0.60	0.1484
0.70	0.0804
0.80	0.0381
0.90	0.0112
1.00	0.0000

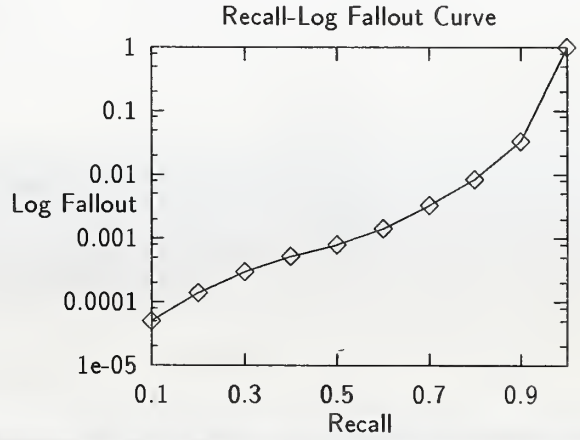
Average precision over all relevant docs	
non-interpolated	0.2088

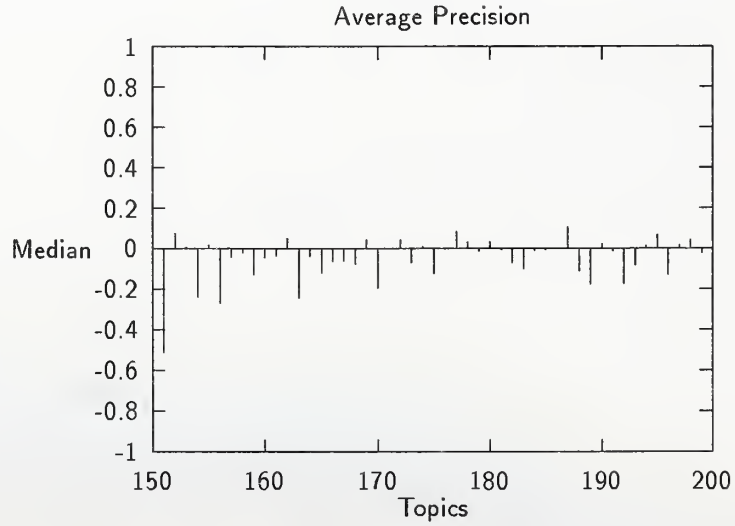
Document Level Averages	
	Precision
At 5 docs	0.4840
At 10 docs	0.4720
At 15 docs	0.4453
At 20 docs	0.4420
At 30 docs	0.4187
At 100 docs	0.3268
At 200 docs	0.2607
At 500 docs	0.1688
At 1000 docs	0.1076

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2822

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00005
0.20	0.00014
0.30	0.00030
0.40	0.00052
0.50	0.00080
0.60	0.00144
0.70	0.00334
0.80	0.00843
0.90	0.03317
1.00	1.00000







Summary Statistics	
Run Number	siems2–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	5981

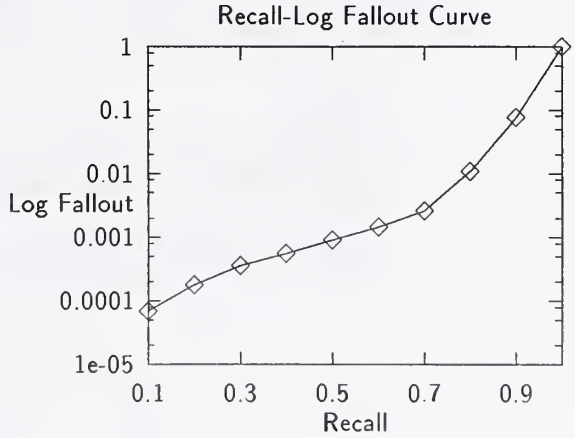
Recall Level Precision Averages	
Recall	Precision
0.00	0.7753
0.10	0.3884
0.20	0.3154
0.30	0.2603
0.40	0.2261
0.50	0.1854
0.60	0.1447
0.70	0.0997
0.80	0.0299
0.90	0.0049
1.00	0.0000

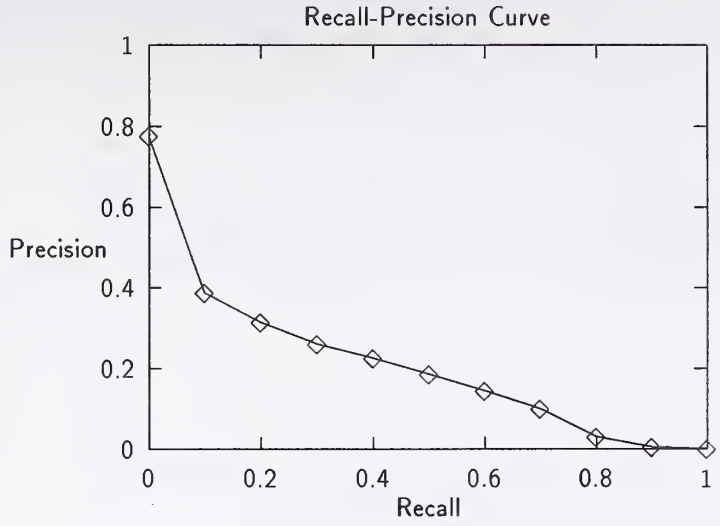
Average precision over all relevant docs	
non-interpolated	0.1873

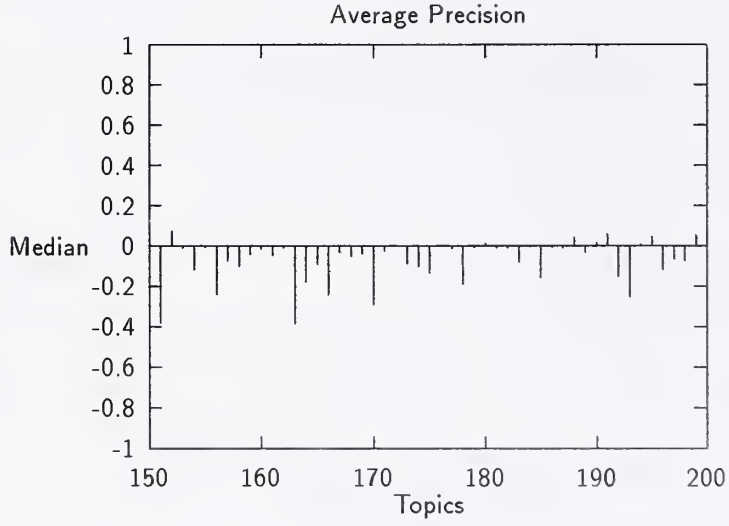
Document Level Averages	
	Precision
At 5 docs	0.4520
At 10 docs	0.3940
At 15 docs	0.3853
At 20 docs	0.3680
At 30 docs	0.3467
At 100 docs	0.2820
At 200 docs	0.2434
At 500 docs	0.1788
At 1000 docs	0.1196

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2635

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00007
0.20	0.00018
0.30	0.00036
0.40	0.00057
0.50	0.00092
0.60	0.00148
0.70	0.00264
0.80	0.01084
0.90	0.07630
1.00	1.00000







Summary Statistics	
Run Number	TOPIC3–category A, manual
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	2350

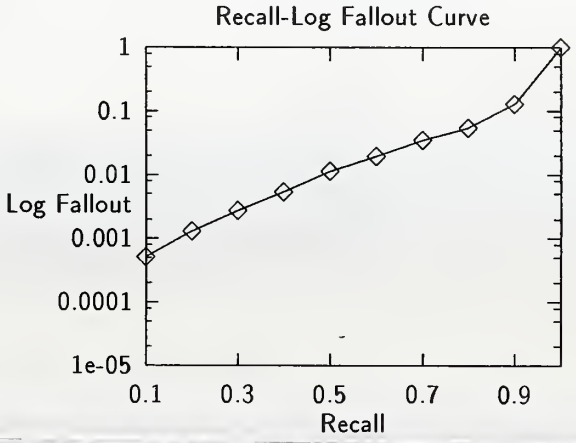
Recall Level Precision Averages	
Recall	Precision
0.00	0.2016
0.10	0.0751
0.20	0.0603
0.30	0.0440
0.40	0.0297
0.50	0.0180
0.60	0.0128
0.70	0.0084
0.80	0.0061
0.90	0.0029
1.00	0.0000

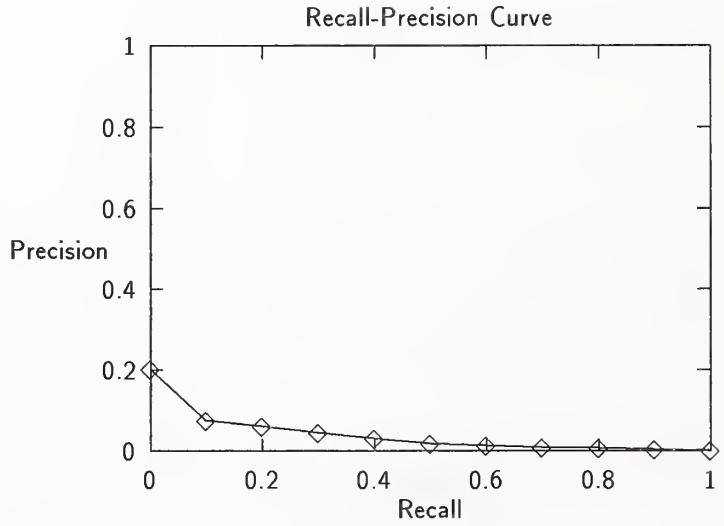
Average precision over all relevant docs	
non-interpolated	0.0286

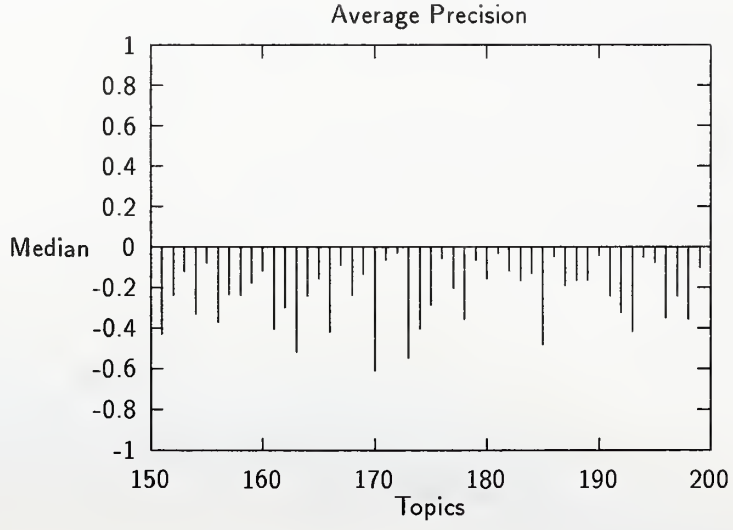
Document Level Averages	
	Precision
At 5 docs	0.0760
At 10 docs	0.0820
At 15 docs	0.0747
At 20 docs	0.0620
At 30 docs	0.0673
At 100 docs	0.0564
At 200 docs	0.0587
At 500 docs	0.0560
At 1000 docs	0.0470

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0610

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00051
0.20	0.00130
0.30	0.00272
0.40	0.00546
0.50	0.01139
0.60	0.01932
0.70	0.03450
0.80	0.05441
0.90	0.12918
1.00	1.00000







Summary Statistics	
Run Number	TOPIC4–category A, feedback
Number of Topics	50

Total number of documents over all topics	
Retrieved:	28709
Relevant:	9805
Rel_ret:	4358

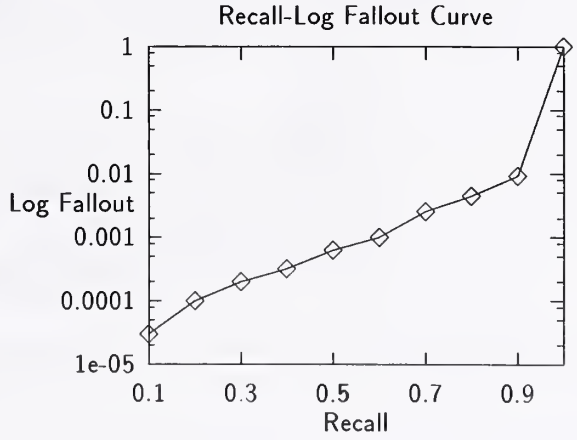
Recall Level Precision Averages	
Recall	Precision
0.00	0.8531
0.10	0.5769
0.20	0.4561
0.30	0.3842
0.40	0.3430
0.50	0.2498
0.60	0.1994
0.70	0.1029
0.80	0.0689
0.90	0.0391
1.00	0.0000

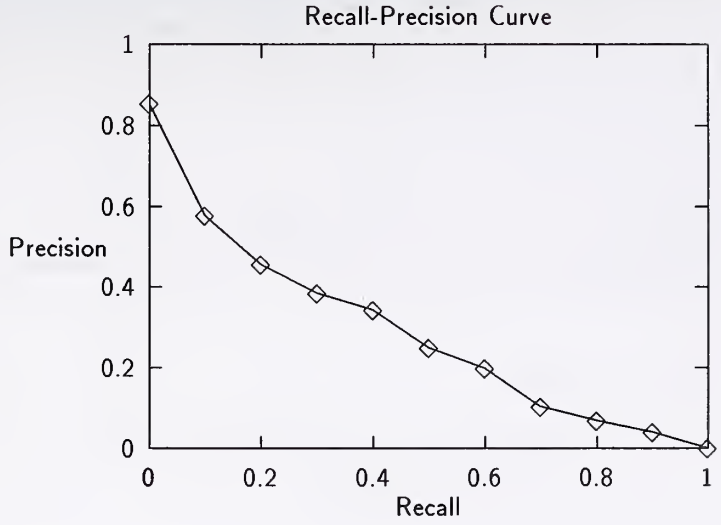
Average precision over all relevant docs	
non-interpolated	0.2689

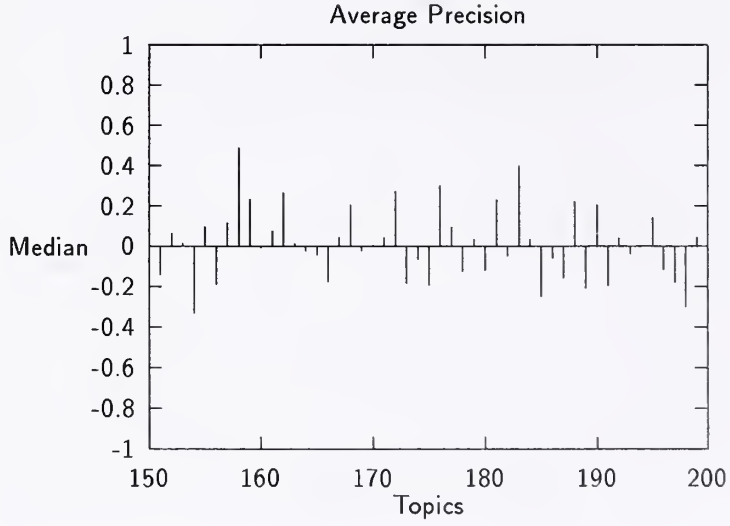
Document Level Averages	
	Precision
At 5 docs	0.6360
At 10 docs	0.5960
At 15 docs	0.5587
At 20 docs	0.5470
At 30 docs	0.5180
At 100 docs	0.3998
At 200 docs	0.2972
At 500 docs	0.1547
At 1000 docs	0.0872

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3510

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00010
0.30	0.00020
0.40	0.00032
0.50	0.00063
0.60	0.00101
0.70	0.00255
0.80	0.00451
0.90	0.00923
1.00	1.00000







Summary Statistics	
Run Number	virtu1–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	3093

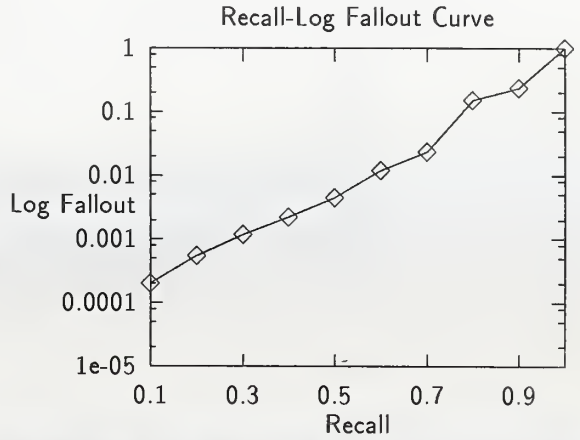
Recall Level Precision Averages	
Recall	Precision
0.00	0.3339
0.10	0.1714
0.20	0.1316
0.30	0.0959
0.40	0.0692
0.50	0.0444
0.60	0.0204
0.70	0.0123
0.80	0.0022
0.90	0.0016
1.00	0.0000

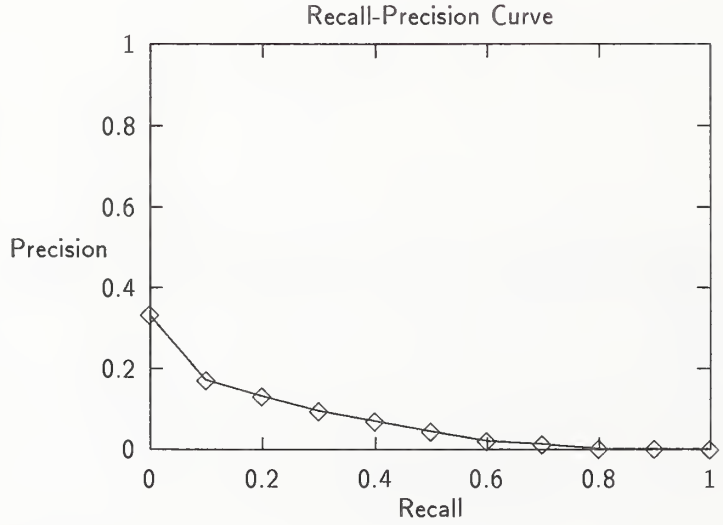
Average precision over all relevant docs	
non-interpolated	0.0624

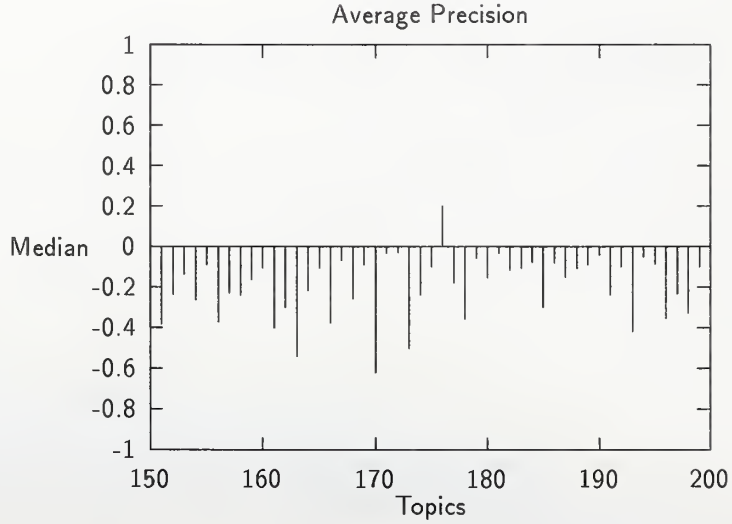
Document Level Averages	
	Precision
At 5 docs	0.1480
At 10 docs	0.1620
At 15 docs	0.1600
At 20 docs	0.1630
At 30 docs	0.1593
At 100 docs	0.1356
At 200 docs	0.1180
At 500 docs	0.0834
At 1000 docs	0.0619

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1170

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00020
0.20	0.00055
0.30	0.00118
0.40	0.00225
0.50	0.00449
0.60	0.01203
0.70	0.02347
0.80	0.15147
0.90	0.23444
1.00	1.00000







Summary Statistics	
Run Number	VTc2s2–category A, manual
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	6219

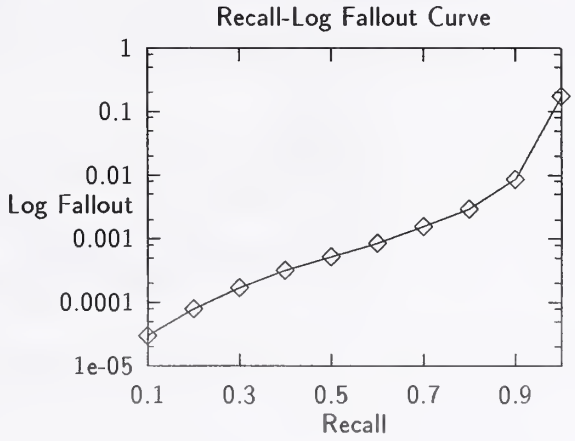
Recall Level Precision Averages	
Recall	Precision
0.00	0.8544
0.10	0.6069
0.20	0.5148
0.30	0.4176
0.40	0.3457
0.50	0.2875
0.60	0.2280
0.70	0.1578
0.80	0.1011
0.90	0.0416
1.00	0.0024

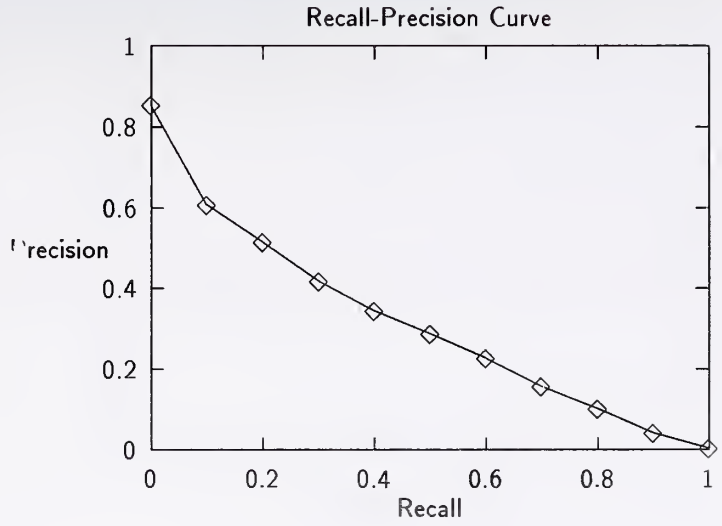
Average precision over all relevant docs	
non-interpolated	0.3021

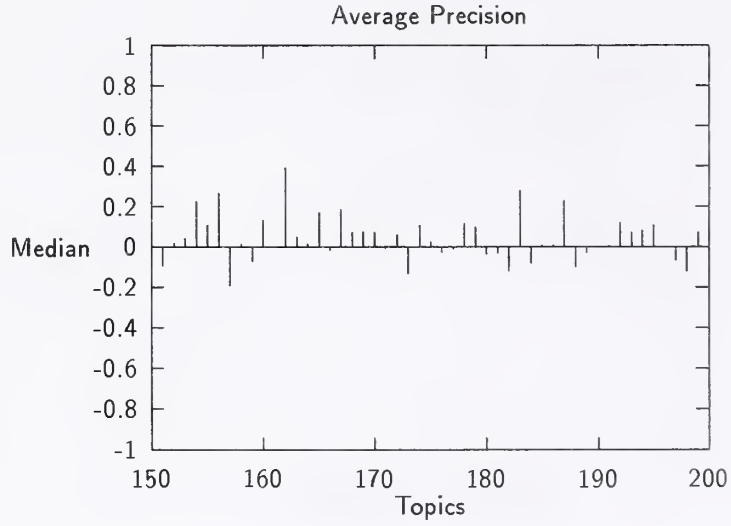
Document Level Averages	
	Precision
At 5 docs	0.6280
At 10 docs	0.6220
At 15 docs	0.5960
At 20 docs	0.5740
At 30 docs	0.5400
At 100 docs	0.4044
At 200 docs	0.3169
At 500 docs	0.1975
At 1000 docs	0.1244

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3538

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00008
0.30	0.00017
0.40	0.00032
0.50	0.00052
0.60	0.00085
0.70	0.00156
0.80	0.00297
0.90	0.00866
1.00	0.17352







Summary Statistics	
Run Number	VTc5s2–category A, manual
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	6077

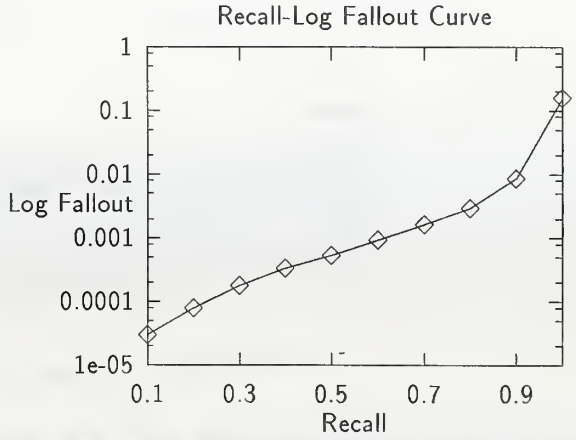
Recall Level Precision Averages	
Recall	Precision
0.00	0.8326
0.10	0.5909
0.20	0.4963
0.30	0.4081
0.40	0.3299
0.50	0.2774
0.60	0.2081
0.70	0.1503
0.80	0.1024
0.90	0.0416
1.00	0.0026

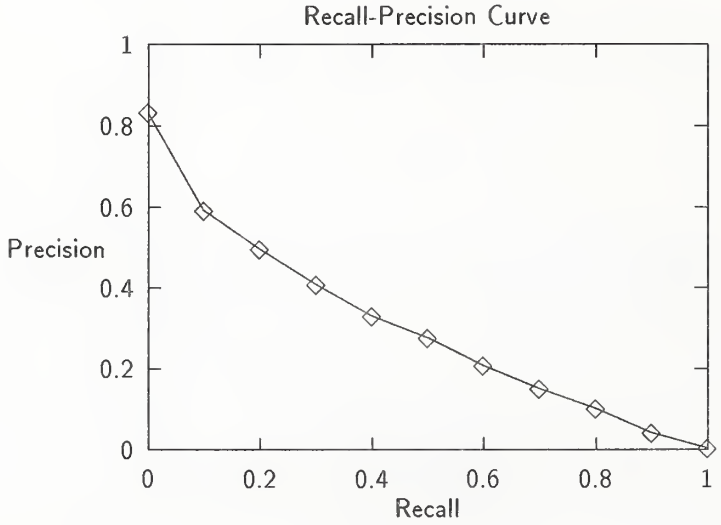
Average precision over all relevant docs	
non-interpolated	0.2914

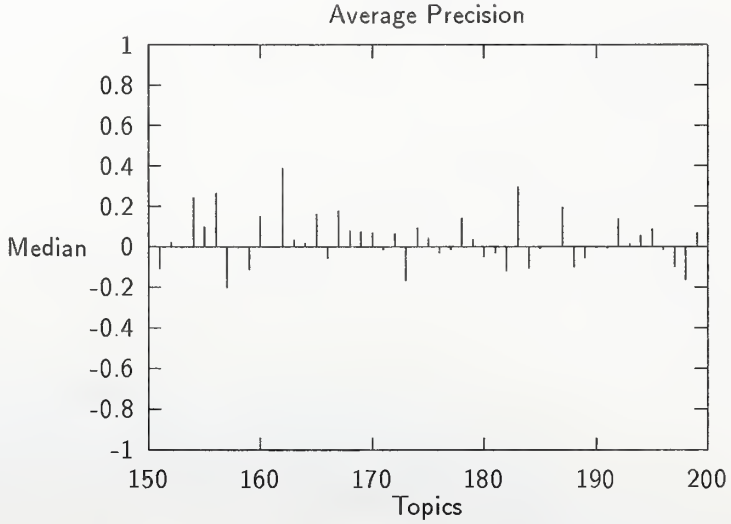
Document Level Averages	
	Precision
At 5 docs	0.6240
At 10 docs	0.5980
At 15 docs	0.5747
At 20 docs	0.5460
At 30 docs	0.5200
At 100 docs	0.3964
At 200 docs	0.3173
At 500 docs	0.1926
At 1000 docs	0.1215

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3404

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00008
0.30	0.00018
0.40	0.00034
0.50	0.00054
0.60	0.00095
0.70	0.00165
0.80	0.00293
0.90	0.00866
1.00	0.16014







Summary Statistics	
Run Number	westp1–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	6237

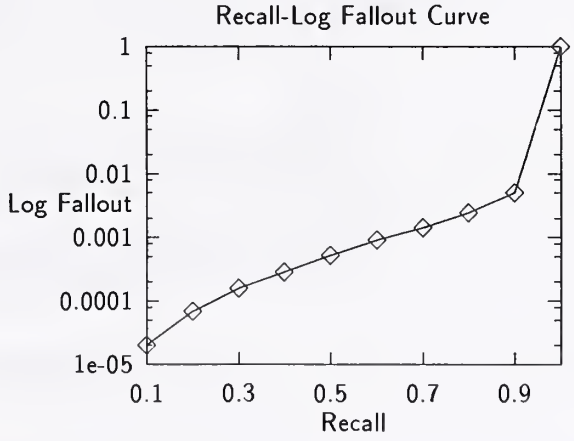
Recall Level Precision Averages	
Recall	Precision
0.00	0.8483
0.10	0.6367
0.20	0.5405
0.30	0.4415
0.40	0.3688
0.50	0.2864
0.60	0.2126
0.70	0.1709
0.80	0.1190
0.90	0.0697
1.00	0.0000

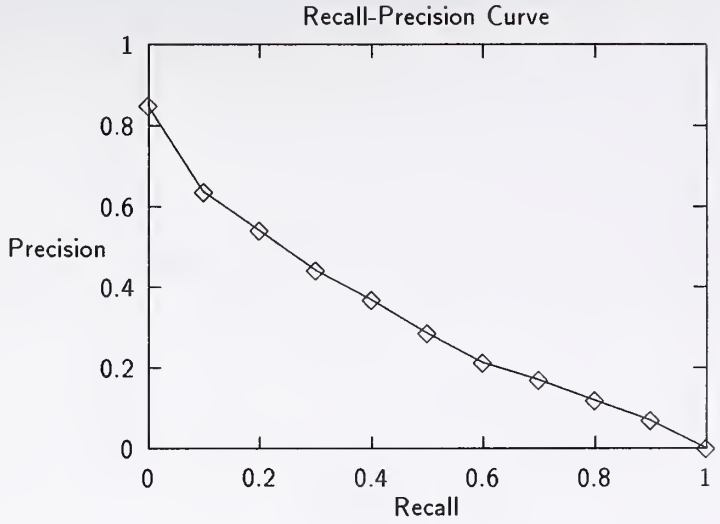
Average precision over all relevant docs	
non-interpolated	0.3157

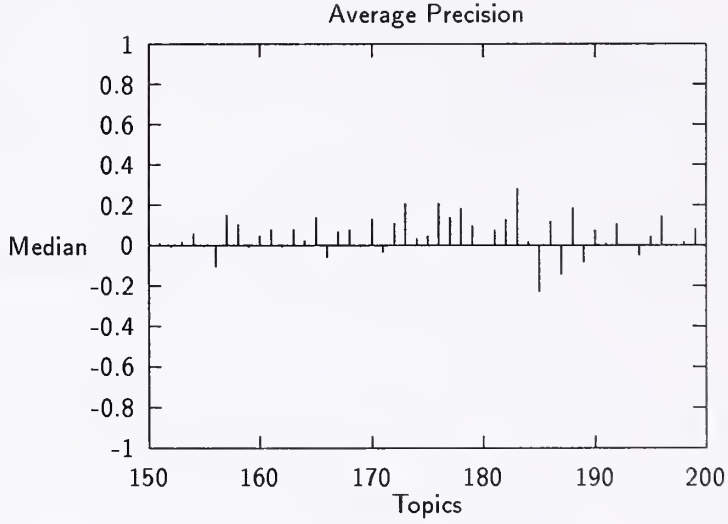
Document Level Averages	
	Precision
At 5 docs	0.6800
At 10 docs	0.6220
At 15 docs	0.6040
At 20 docs	0.5830
At 30 docs	0.5420
At 100 docs	0.4092
At 200 docs	0.3148
At 500 docs	0.1992
At 1000 docs	0.1247

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3780

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00002
0.20	0.00007
0.30	0.00016
0.40	0.00029
0.50	0.00052
0.60	0.00093
0.70	0.00142
0.80	0.00247
0.90	0.00501
1.00	1.00000







Summary Statistics	
Run Number	xerox3–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	5542

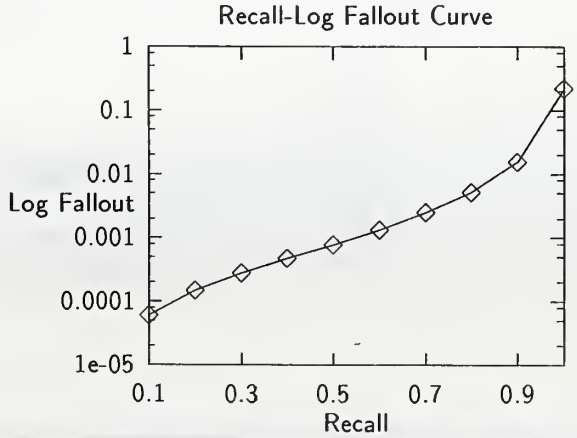
Recall Level Precision Averages	
Recall	Precision
0.00	0.6290
0.10	0.4253
0.20	0.3533
0.30	0.3086
0.40	0.2570
0.50	0.2103
0.60	0.1567
0.70	0.1055
0.80	0.0603
0.90	0.0240
1.00	0.0019

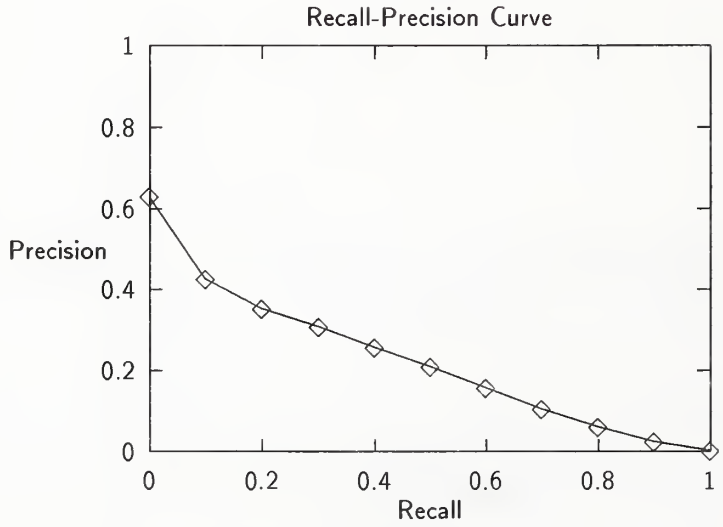
Average precision over all relevant docs	
non-interpolated	0.2092

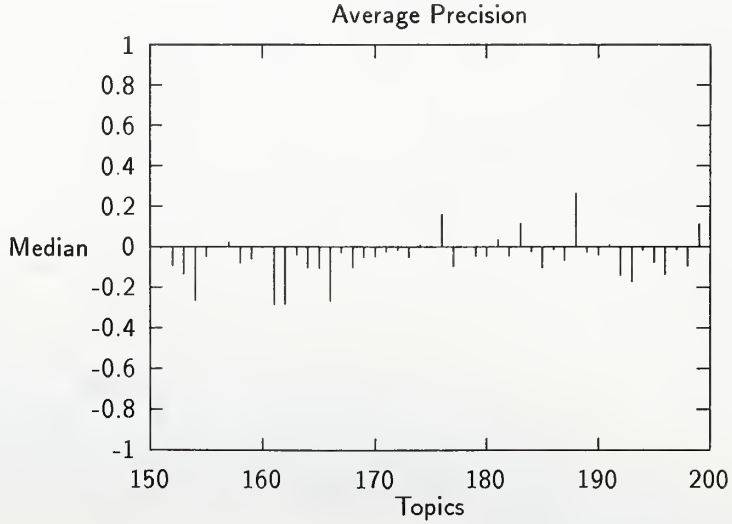
Document Level Averages	
	Precision
At 5 docs	0.3760
At 10 docs	0.3960
At 15 docs	0.3907
At 20 docs	0.3840
At 30 docs	0.3780
At 100 docs	0.2996
At 200 docs	0.2466
At 500 docs	0.1614
At 1000 docs	0.1108

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2765

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00006
0.20	0.00015
0.30	0.00028
0.40	0.00048
0.50	0.00078
0.60	0.00135
0.70	0.00248
0.80	0.00520
0.90	0.01528
1.00	0.21929







Summary Statistics	
Run Number	xerox4–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9805
Rel_ret:	5306

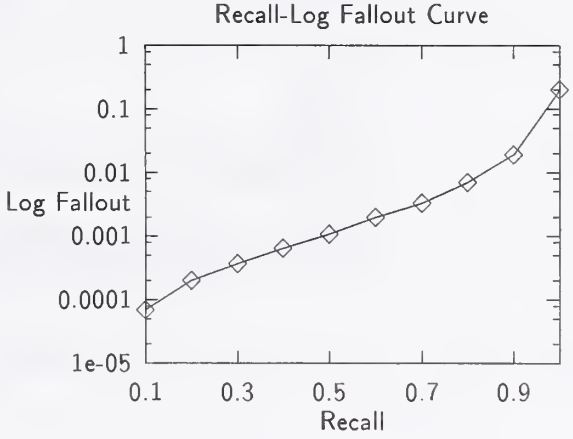
Recall Level Precision Averages	
Recall	Precision
0.00	0.5914
0.10	0.3601
0.20	0.2941
0.30	0.2549
0.40	0.2055
0.50	0.1608
0.60	0.1114
0.70	0.0805
0.80	0.0454
0.90	0.0194
1.00	0.0021

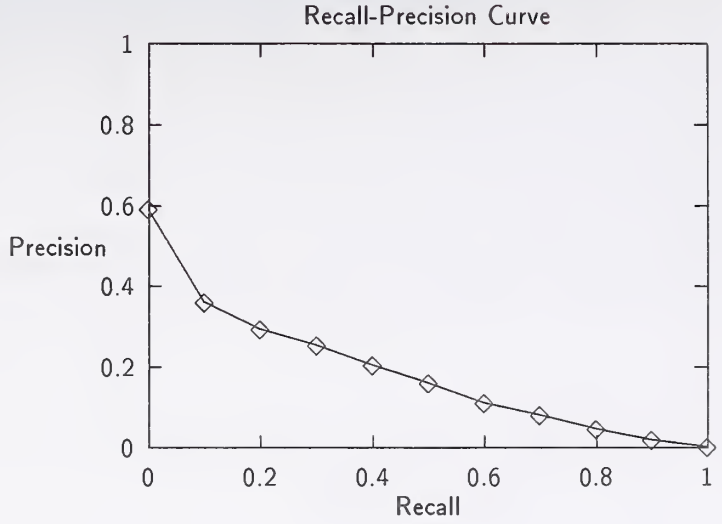
Average precision over all relevant docs	
non-interpolated	0.1693

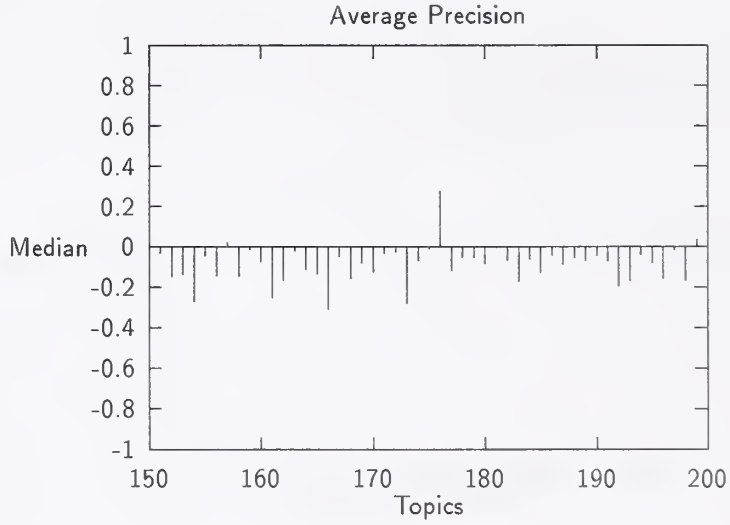
Document Level Averages	
	Precision
At 5 docs	0.3200
At 10 docs	0.3320
At 15 docs	0.3160
At 20 docs	0.3160
At 30 docs	0.3140
At 100 docs	0.2560
At 200 docs	0.2092
At 500 docs	0.1450
At 1000 docs	0.1061

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2396

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00007
0.20	0.00020
0.30	0.00037
0.40	0.00065
0.50	0.00109
0.60	0.00200
0.70	0.00334
0.80	0.00702
0.90	0.01899
1.00	0.19837







Summary Statistics	
Run Number	DCUNL1–category B, manual
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	3913
Rel_ret:	2694

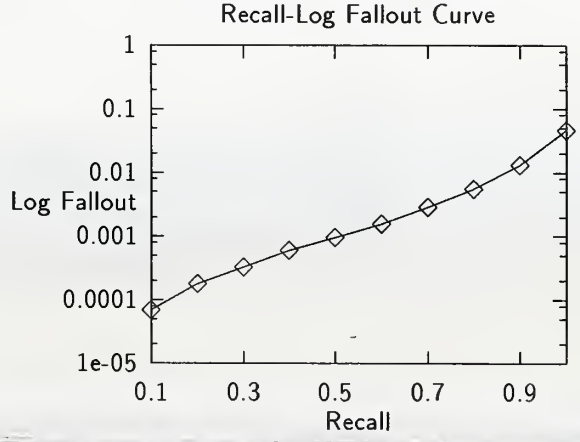
Recall Level Precision Averages	
Recall	Precision
0.00	0.6431
0.10	0.3973
0.20	0.3383
0.30	0.2931
0.40	0.2321
0.50	0.1907
0.60	0.1479
0.70	0.0999
0.80	0.0610
0.90	0.0299
1.00	0.0097

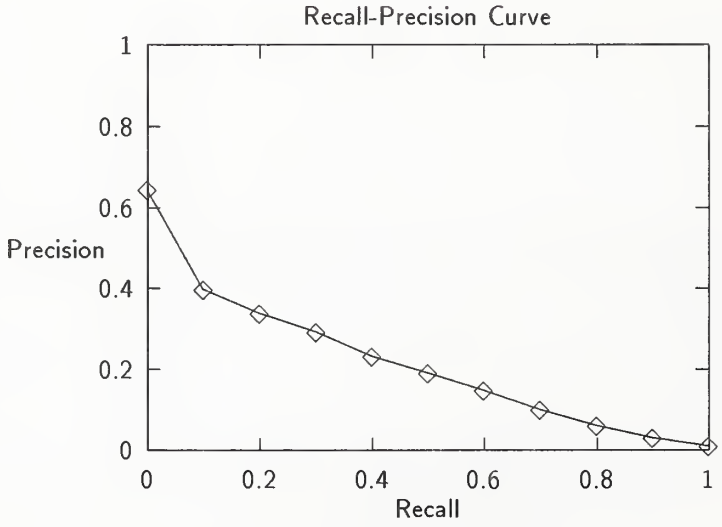
Average precision over all relevant docs	
non-interpolated	0.1996

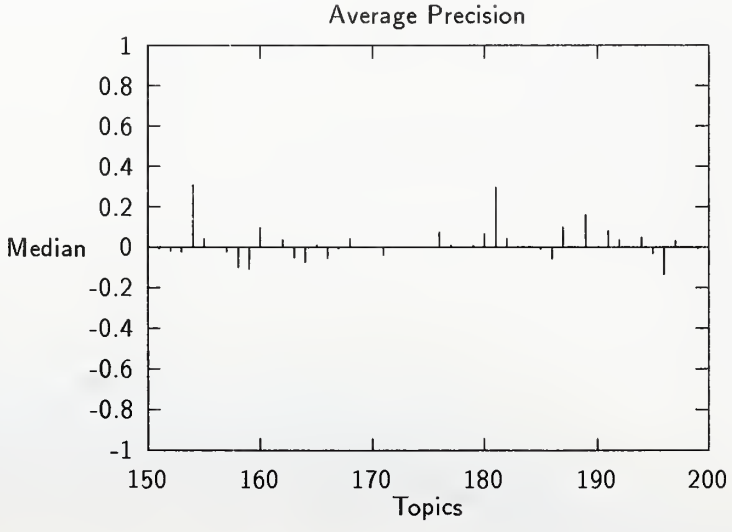
Document Level Averages	
	Precision
At 5 docs	0.4120
At 10 docs	0.3500
At 15 docs	0.3227
At 20 docs	0.3190
At 30 docs	0.2933
At 100 docs	0.2062
At 200 docs	0.1553
At 500 docs	0.0891
At 1000 docs	0.0539

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2393

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00007
0.20	0.00018
0.30	0.00033
0.40	0.00060
0.50	0.00096
0.60	0.00156
0.70	0.00285
0.80	0.00557
0.90	0.01320
1.00	0.04614







Summary Statistics	
Run Number	DCUNL2–category B, manual
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	3913
Rel_ret:	2719

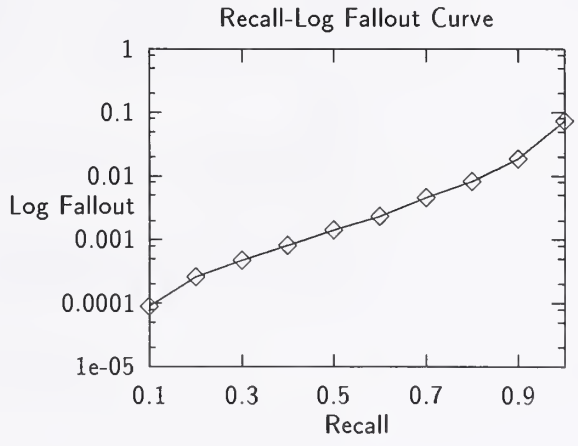
Recall Level Precision Averages	
Recall	Precision
0.00	0.5453
0.10	0.3362
0.20	0.2587
0.30	0.2193
0.40	0.1799
0.50	0.1367
0.60	0.1039
0.70	0.0651
0.80	0.0425
0.90	0.0212
1.00	0.0061

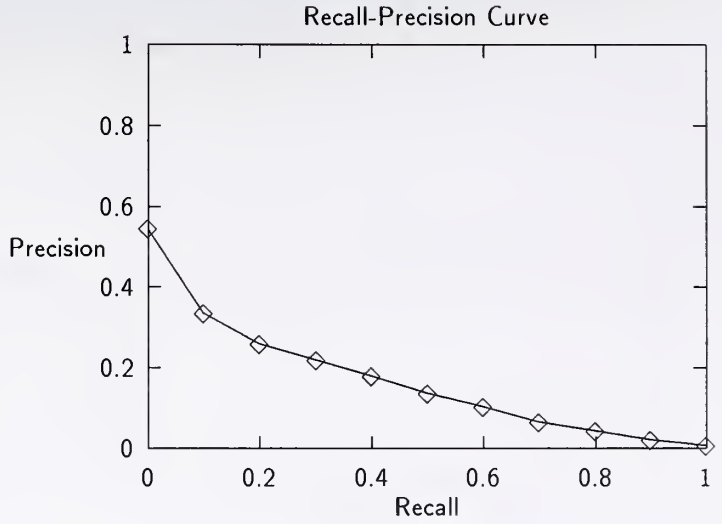
Average precision over all relevant docs	
non-interpolated	0.1514

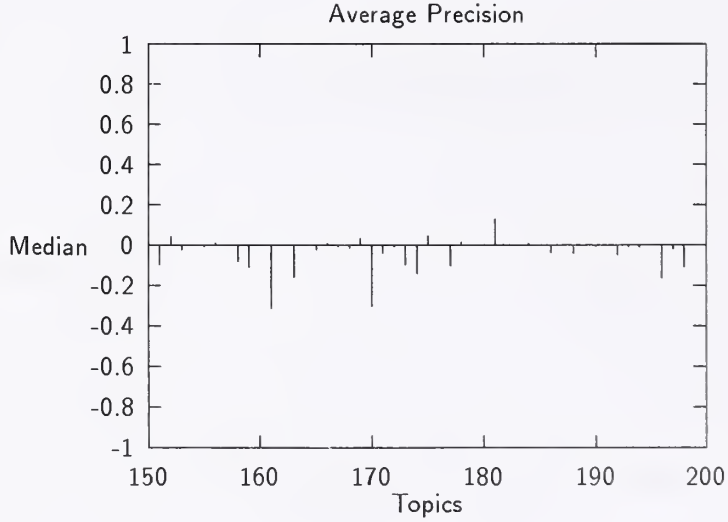
Document Level Averages	
	Precision
At 5 docs	0.3400
At 10 docs	0.2960
At 15 docs	0.2627
At 20 docs	0.2620
At 30 docs	0.2433
At 100 docs	0.1668
At 200 docs	0.1266
At 500 docs	0.0811
At 1000 docs	0.0544

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2040

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00009
0.20	0.00026
0.30	0.00048
0.40	0.00082
0.50	0.00143
0.60	0.00234
0.70	0.00454
0.80	0.00815
0.90	0.01878
1.00	0.07363







Summary Statistics	
Run Number	dgros1–category B, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	3913
Rel_ret:	1256

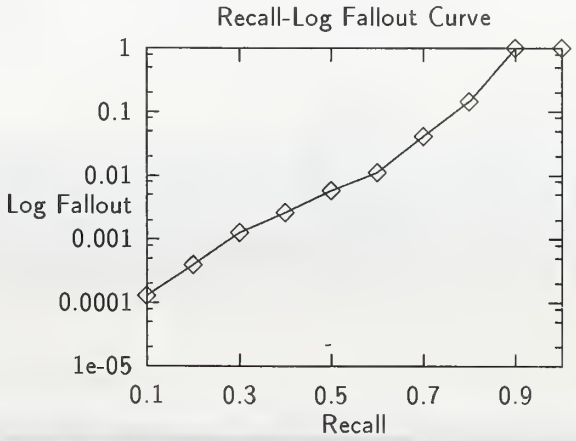
Recall Level Precision Averages	
Recall	Precision
0.00	0.5327
0.10	0.2578
0.20	0.1831
0.30	0.0965
0.40	0.0650
0.50	0.0377
0.60	0.0239
0.70	0.0077
0.80	0.0025
0.90	0.0000
1.00	0.0000

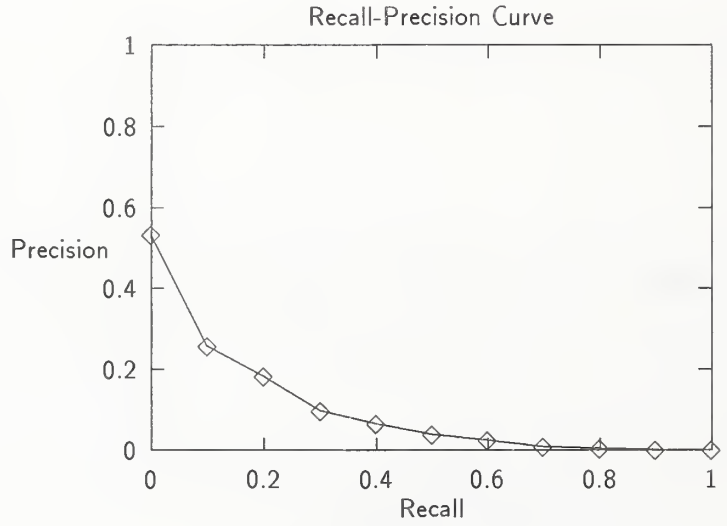
Average precision over all relevant docs	
non-interpolated	0.0860

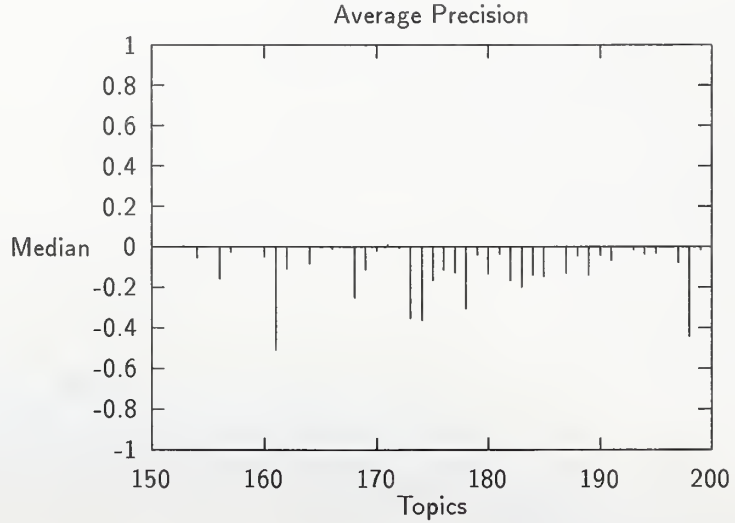
Document Level Averages	
	Precision
At 5 docs	0.2920
At 10 docs	0.2480
At 15 docs	0.2267
At 20 docs	0.2180
At 30 docs	0.1893
At 100 docs	0.1094
At 200 docs	0.0772
At 500 docs	0.0409
At 1000 docs	0.0251

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1356

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00013
0.20	0.00040
0.30	0.00127
0.40	0.00260
0.50	0.00577
0.60	0.01107
0.70	0.04077
0.80	0.14425
0.90	1.00000
1.00	1.00000







Summary Statistics	
Run Number	expst2–category B, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	3913
Rel_ret:	2218

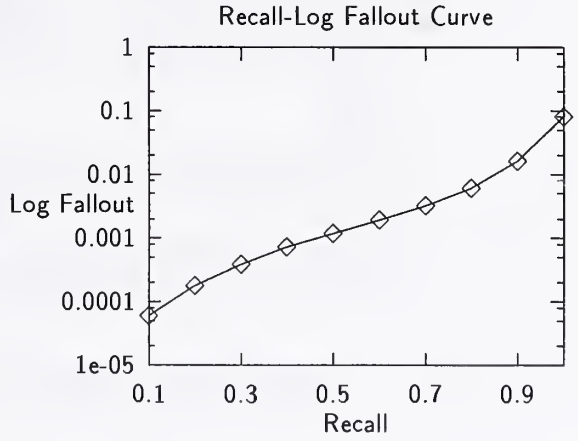
Recall Level Precision Averages	
Recall	Precision
0.00	0.6168
0.10	0.4363
0.20	0.3397
0.30	0.2603
0.40	0.2003
0.50	0.1586
0.60	0.1239
0.70	0.0887
0.80	0.0557
0.90	0.0248
1.00	0.0056

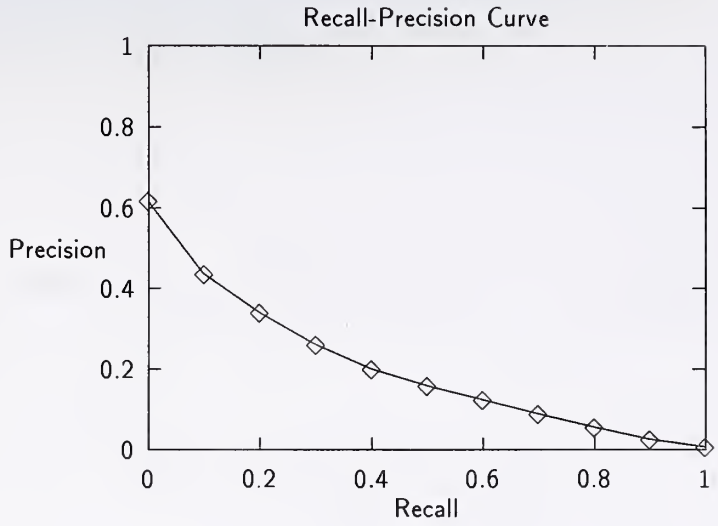
Average precision over all relevant docs	
non-interpolated	0.1910

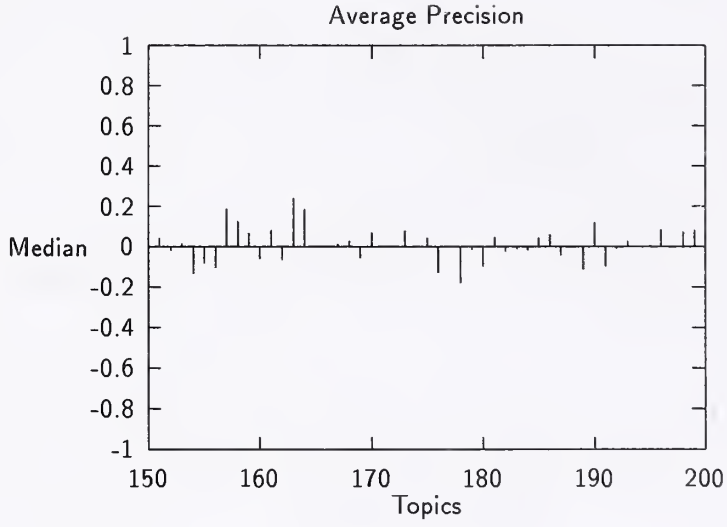
Document Level Averages	
	Precision
At 5 docs	0.4160
At 10 docs	0.3680
At 15 docs	0.3320
At 20 docs	0.3170
At 30 docs	0.2887
At 100 docs	0.1942
At 200 docs	0.1344
At 500 docs	0.0739
At 1000 docs	0.0444

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2319

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00006
0.20	0.00018
0.30	0.00039
0.40	0.00072
0.50	0.00120
0.60	0.00192
0.70	0.00325
0.80	0.00613
0.90	0.01599
1.00	0.08025







Summary Statistics	
Run Number	UniNE1–category B, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	3913
Rel_ret:	3191

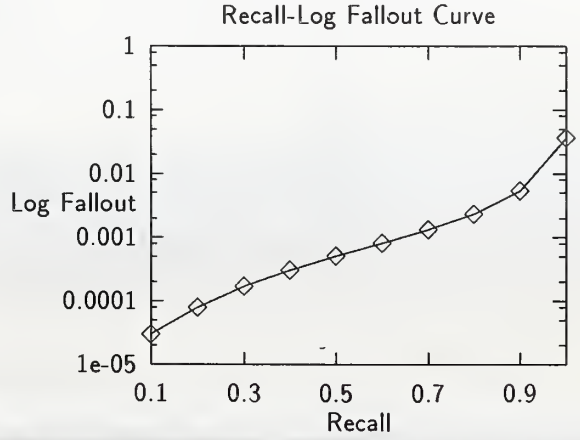
Recall Level Precision Averages	
Recall	Precision
0.00	0.8025
0.10	0.6297
0.20	0.5423
0.30	0.4392
0.40	0.3681
0.50	0.3052
0.60	0.2513
0.70	0.1914
0.80	0.1340
0.90	0.0705
1.00	0.0123

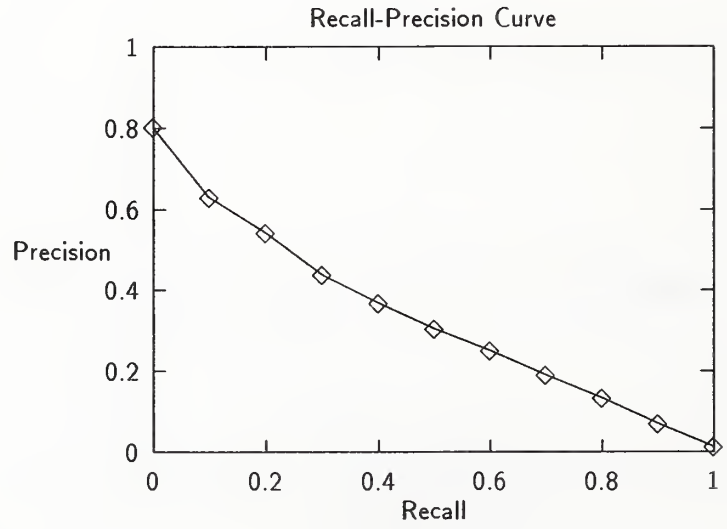
Average precision over all relevant docs	
non-interpolated	0.3190

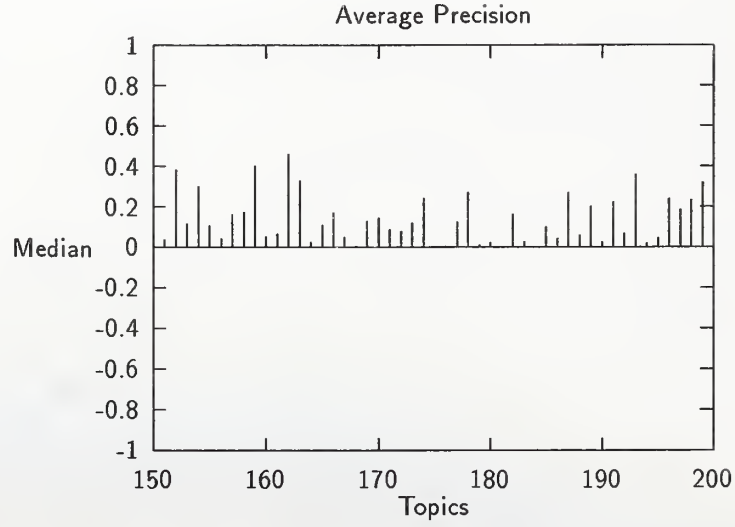
Document Level Averages	
	Precision
At 5 docs	0.5200
At 10 docs	0.4740
At 15 docs	0.4560
At 20 docs	0.4320
At 30 docs	0.4040
At 100 docs	0.2806
At 200 docs	0.2043
At 500 docs	0.1110
At 1000 docs	0.0638

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3442

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00008
0.30	0.00017
0.40	0.00031
0.50	0.00051
0.60	0.00081
0.70	0.00134
0.80	0.00234
0.90	0.00536
1.00	0.03629







Summary Statistics	
Run Number	UniNE2–category B, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	3913
Rel_ret:	3191

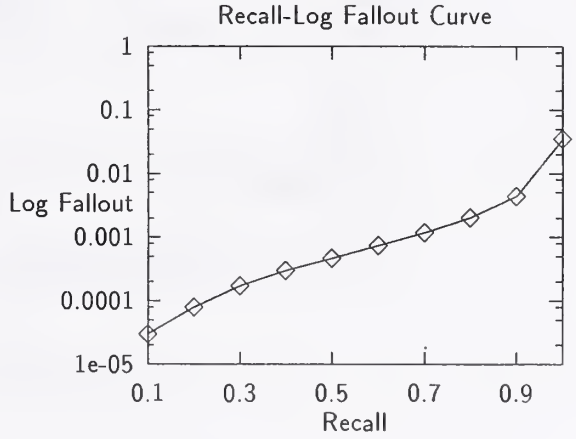
Recall Level Precision Averages	
Recall	Precision
0.00	0.8107
0.10	0.6272
0.20	0.5389
0.30	0.4428
0.40	0.3799
0.50	0.3251
0.60	0.2688
0.70	0.2082
0.80	0.1520
0.90	0.0845
1.00	0.0129

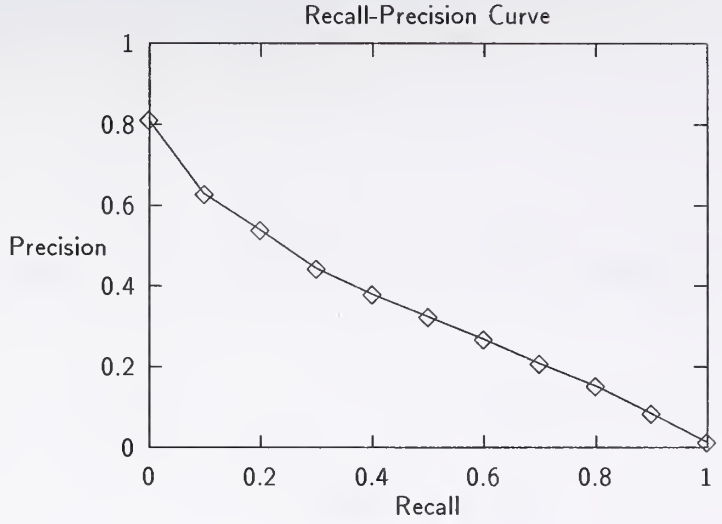
Average precision over all relevant docs	
non-interpolated	0.3279

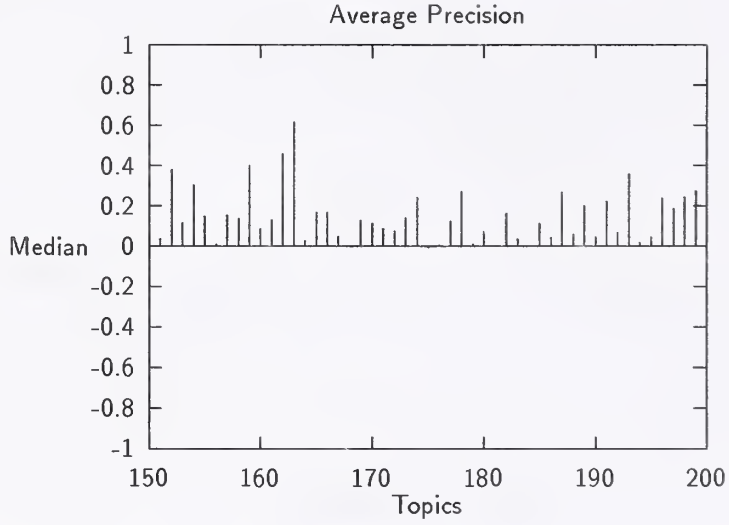
Document Level Averages	
	Precision
At 5 docs	0.5200
At 10 docs	0.4900
At 15 docs	0.4587
At 20 docs	0.4360
At 30 docs	0.4067
At 100 docs	0.2836
At 200 docs	0.2063
At 500 docs	0.1116
At 1000 docs	0.0638

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3502

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00008
0.30	0.00017
0.40	0.00030
0.50	0.00047
0.60	0.00074
0.70	0.00120
0.80	0.00202
0.90	0.00441
1.00	0.03458







Summary Statistics	
Run Number	ACQNT2–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	5679

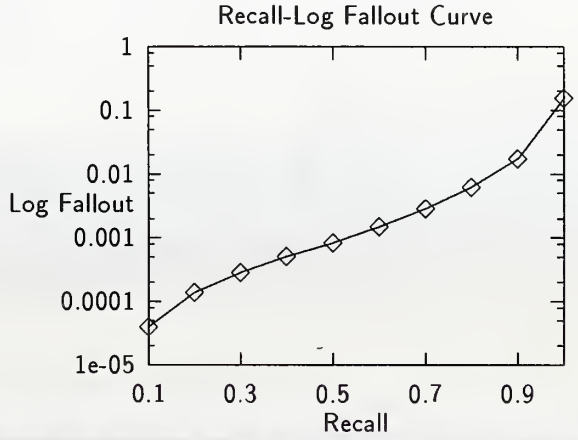
Recall Level Precision Averages	
Recall	Precision
0.00	0.7743
0.10	0.5672
0.20	0.4468
0.30	0.3648
0.40	0.3053
0.50	0.2483
0.60	0.1829
0.70	0.1194
0.80	0.0669
0.90	0.0281
1.00	0.0036

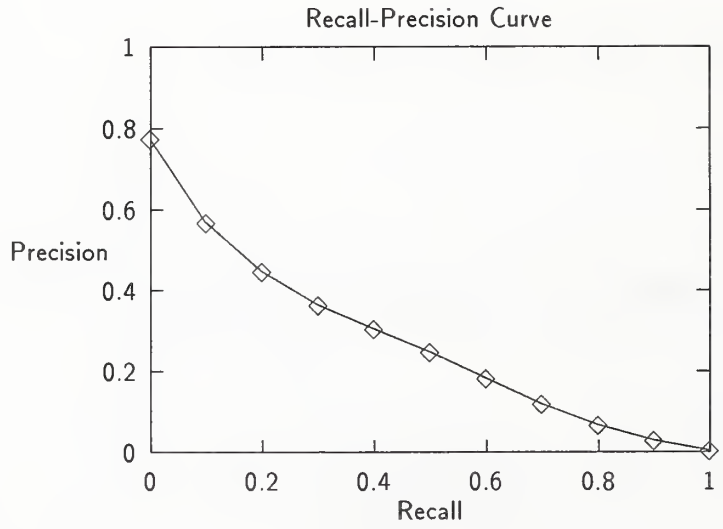
Average precision over all relevant docs	
non-interpolated	0.2641

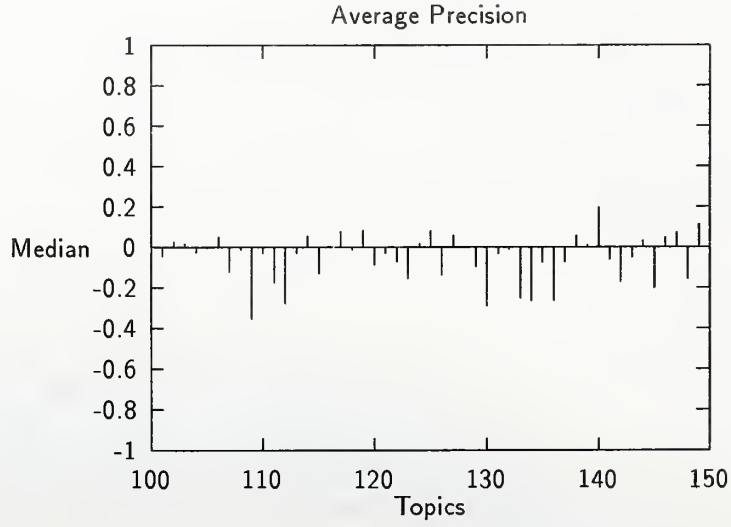
Document Level Averages	
	Precision
At 5 docs	0.6360
At 10 docs	0.5760
At 15 docs	0.5427
At 20 docs	0.5330
At 30 docs	0.4960
At 100 docs	0.3726
At 200 docs	0.2948
At 500 docs	0.1841
At 1000 docs	0.1136

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3147

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00004
0.20	0.00014
0.30	0.00029
0.40	0.00051
0.50	0.00084
0.60	0.00149
0.70	0.00287
0.80	0.00621
0.90	0.01732
1.00	0.15403







Summary Statistics	
Run Number	Brkly8–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	6828

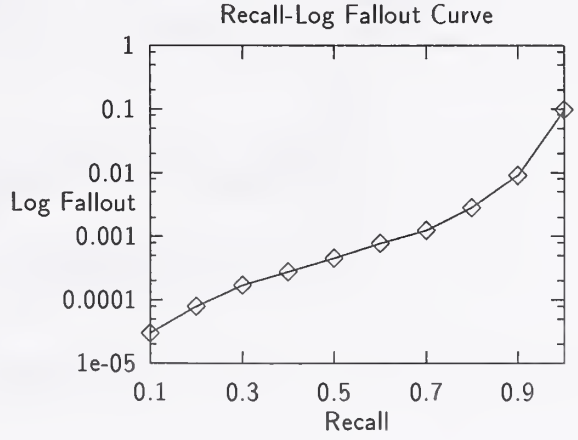
Recall Level Precision Averages	
Recall	Precision
0.00	0.8382
0.10	0.6775
0.20	0.5837
0.30	0.4938
0.40	0.4386
0.50	0.3774
0.60	0.2975
0.70	0.2388
0.80	0.1373
0.90	0.0524
1.00	0.0057

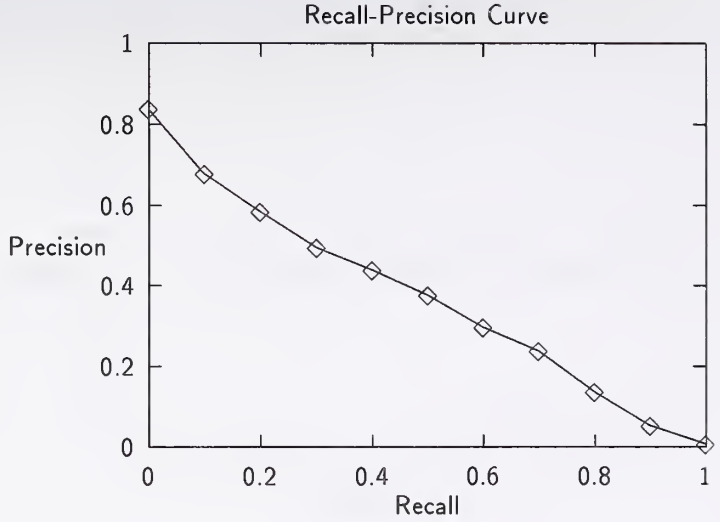
Average precision over all relevant docs	
non-interpolated	0.3643

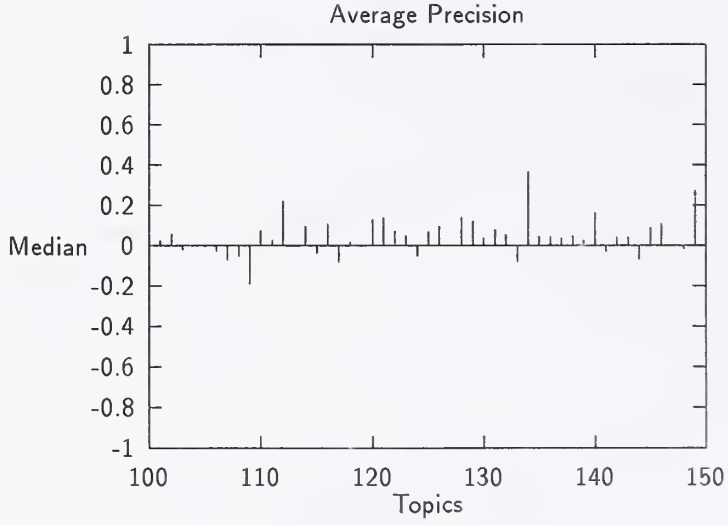
Document Level Averages	
	Precision
At 5 docs	0.6960
At 10 docs	0.6520
At 15 docs	0.6413
At 20 docs	0.6130
At 30 docs	0.5760
At 100 docs	0.4426
At 200 docs	0.3607
At 500 docs	0.2226
At 1000 docs	0.1366

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4004

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00008
0.30	0.00017
0.40	0.00028
0.50	0.00046
0.60	0.00079
0.70	0.00124
0.80	0.00280
0.90	0.00906
1.00	0.09708







Summary Statistics	
Run Number	cityi1–category A, feedback
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	6059

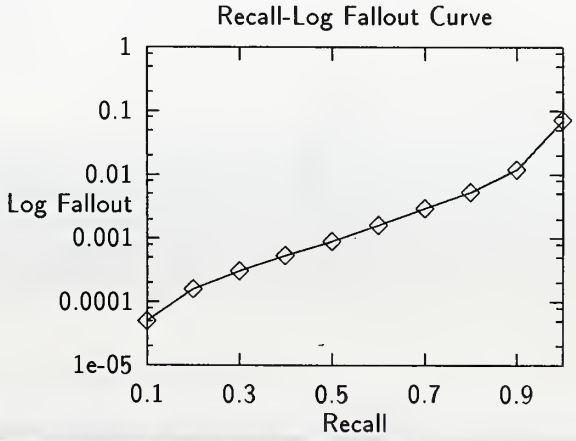
Recall Level Precision Averages	
Recall	Precision
0.00	0.7521
0.10	0.5151
0.20	0.4116
0.30	0.3469
0.40	0.2909
0.50	0.2385
0.60	0.1722
0.70	0.1172
0.80	0.0771
0.90	0.0407
1.00	0.0077

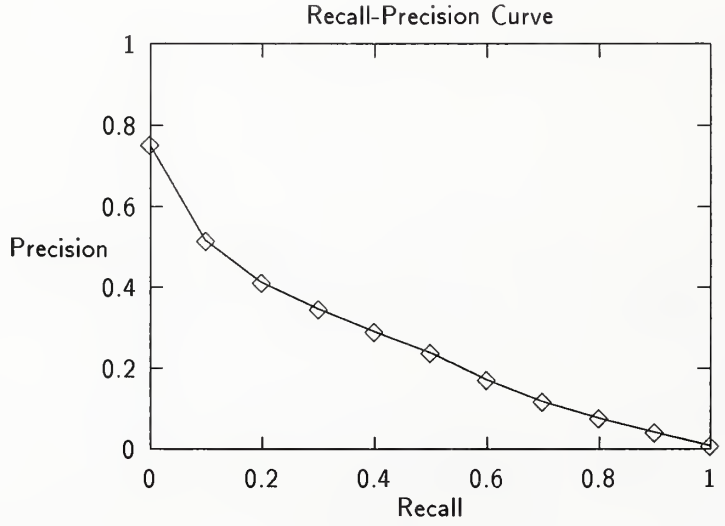
Average precision over all relevant docs	
non-interpolated	0.2498

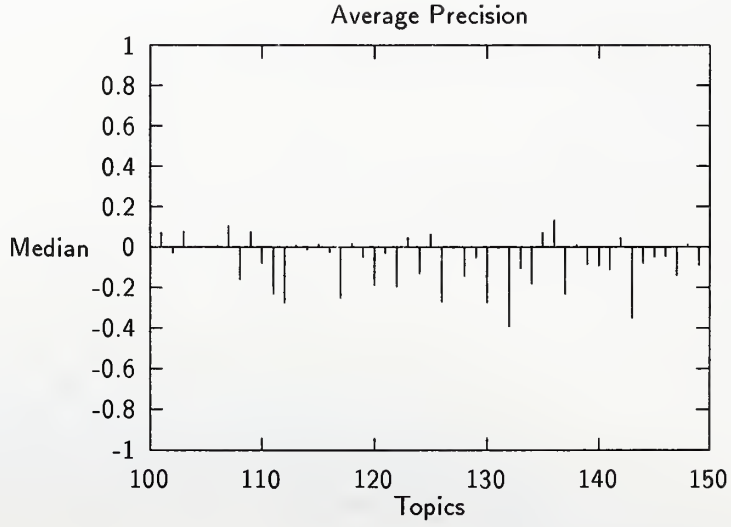
Document Level Averages	
	Precision
At 5 docs	0.5600
At 10 docs	0.5080
At 15 docs	0.4987
At 20 docs	0.4840
At 30 docs	0.4453
At 100 docs	0.3446
At 200 docs	0.2840
At 500 docs	0.1826
At 1000 docs	0.1212

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3024

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00005
0.20	0.00016
0.30	0.00031
0.40	0.00054
0.50	0.00089
0.60	0.00161
0.70	0.00293
0.80	0.00533
0.90	0.01181
1.00	0.07172







Summary Statistics	
Run Number	cityr1–category A, automatic
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	7158

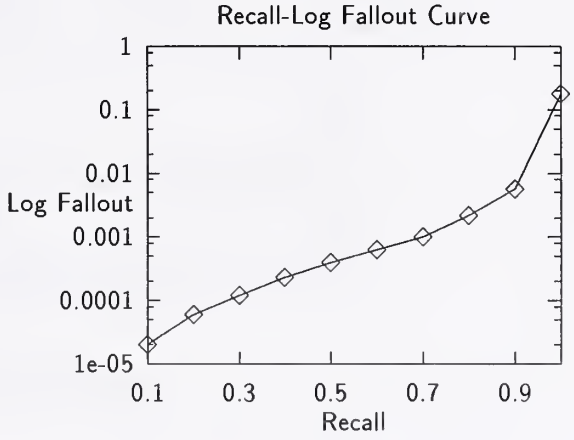
Recall Level Precision Averages	
Recall	Precision
0.00	0.8395
0.10	0.7288
0.20	0.6332
0.30	0.5772
0.40	0.4922
0.50	0.4118
0.60	0.3481
0.70	0.2763
0.80	0.1703
0.90	0.0818
1.00	0.0031

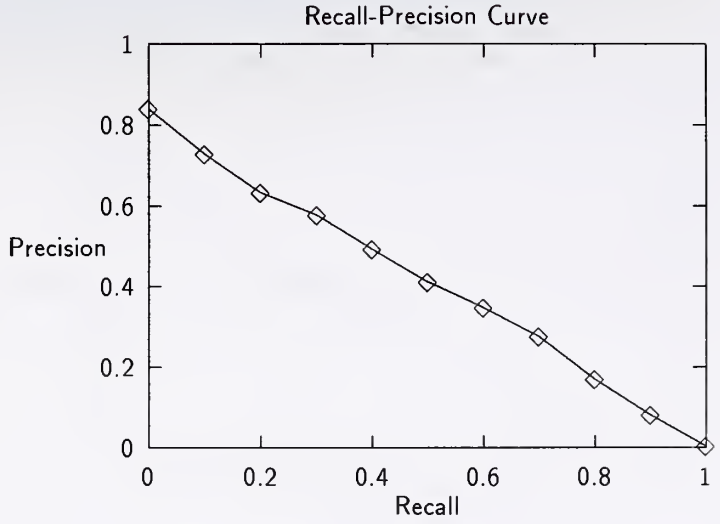
Average precision over all relevant docs	
non-interpolated	0.4068

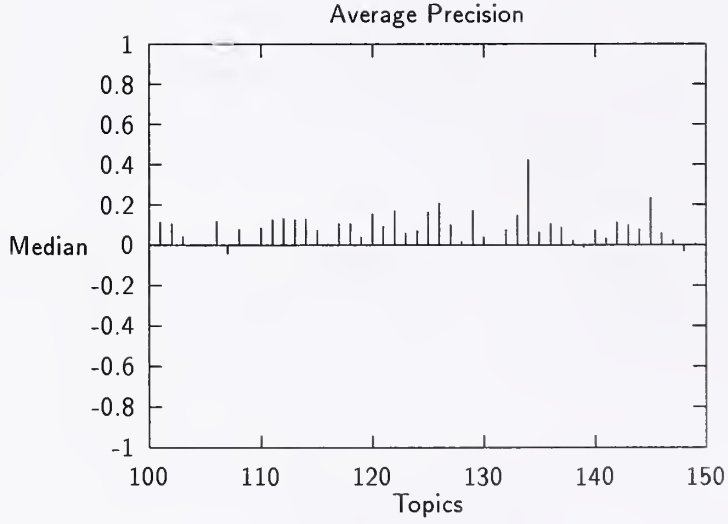
Document Level Averages	
	Precision
At 5 docs	0.7160
At 10 docs	0.6880
At 15 docs	0.6533
At 20 docs	0.6430
At 30 docs	0.6120
At 100 docs	0.4748
At 200 docs	0.3813
At 500 docs	0.2332
At 1000 docs	0.1432

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4352

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00002
0.20	0.00006
0.30	0.00012
0.40	0.00023
0.50	0.00040
0.60	0.00063
0.70	0.00102
0.80	0.00217
0.90	0.00562
1.00	0.17897







Summary Statistics	
Run Number	cityr2–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	6985

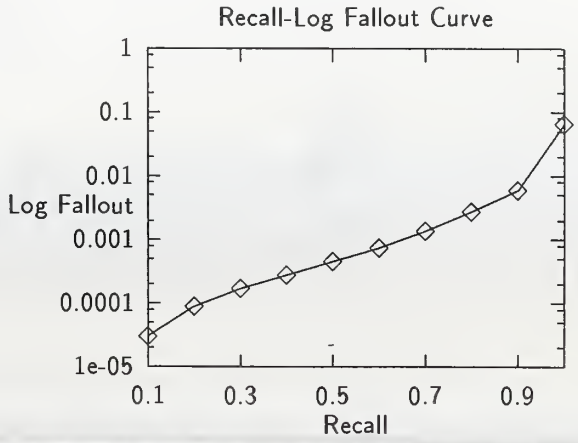
Recall Level Precision Averages	
Recall	Precision
0.00	0.8348
0.10	0.6551
0.20	0.5656
0.30	0.4992
0.40	0.4461
0.50	0.3765
0.60	0.3083
0.70	0.2222
0.80	0.1407
0.90	0.0770
1.00	0.0084

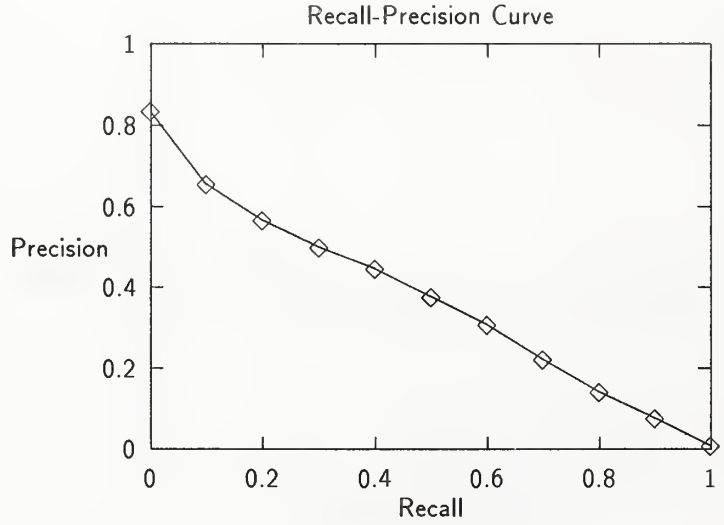
Average precision over all relevant docs	
non-interpolated	0.3621

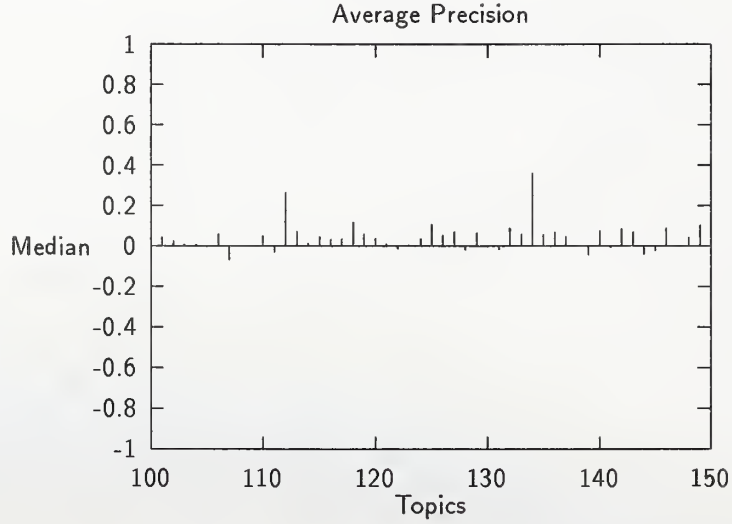
Document Level Averages	
	Precision
At 5 docs	0.6480
At 10 docs	0.6340
At 15 docs	0.6053
At 20 docs	0.5810
At 30 docs	0.5533
At 100 docs	0.4512
At 200 docs	0.3599
At 500 docs	0.2245
At 1000 docs	0.1397

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3920

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00009
0.30	0.00017
0.40	0.00028
0.50	0.00046
0.60	0.00075
0.70	0.00136
0.80	0.00272
0.90	0.00600
1.00	0.06570







Summary Statistics	
Run Number	CLARTA–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	6317

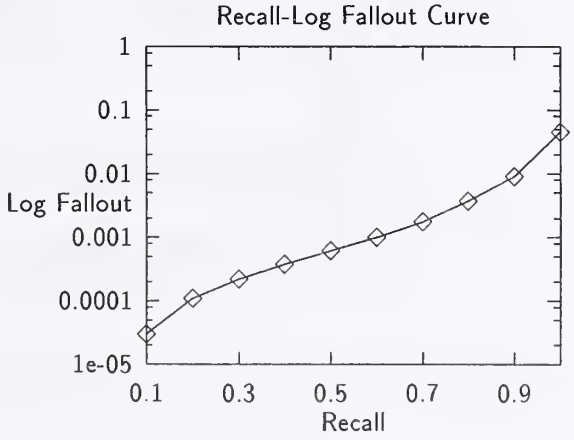
Recall Level Precision Averages	
Recall	Precision
0.00	0.8096
0.10	0.6192
0.20	0.5125
0.30	0.4340
0.40	0.3683
0.50	0.3097
0.60	0.2463
0.70	0.1795
0.80	0.1071
0.90	0.0525
1.00	0.0123

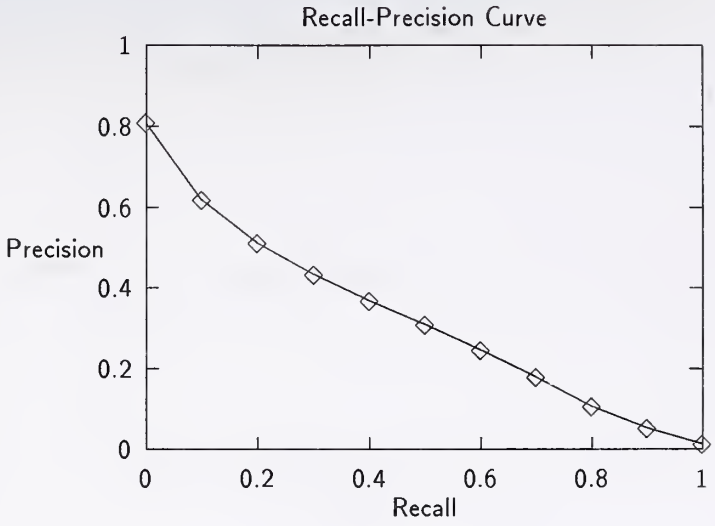
Average precision over all relevant docs	
non-interpolated	0.3140

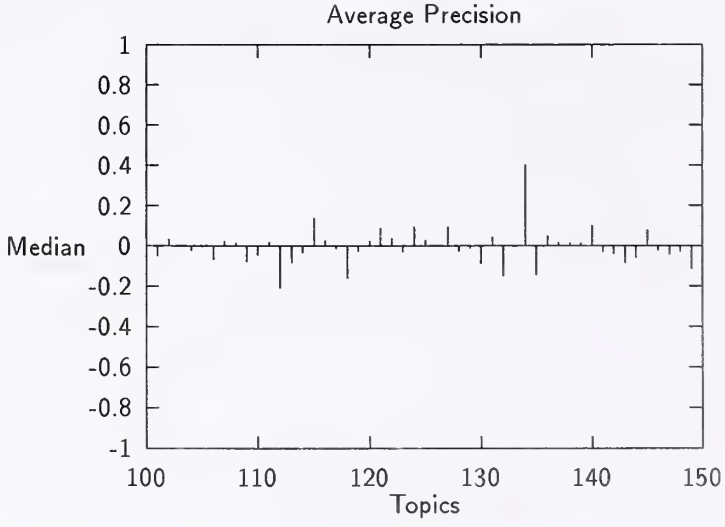
Document Level Averages	
	Precision
At 5 docs	0.6360
At 10 docs	0.6160
At 15 docs	0.5813
At 20 docs	0.5680
At 30 docs	0.5240
At 100 docs	0.3990
At 200 docs	0.3212
At 500 docs	0.2030
At 1000 docs	0.1263

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3612

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00011
0.30	0.00022
0.40	0.00038
0.50	0.00062
0.60	0.00102
0.70	0.00178
0.80	0.00371
0.90	0.00904
1.00	0.04469







Summary Statistics	
Run Number	crnlQR–category A, automatic
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	7215

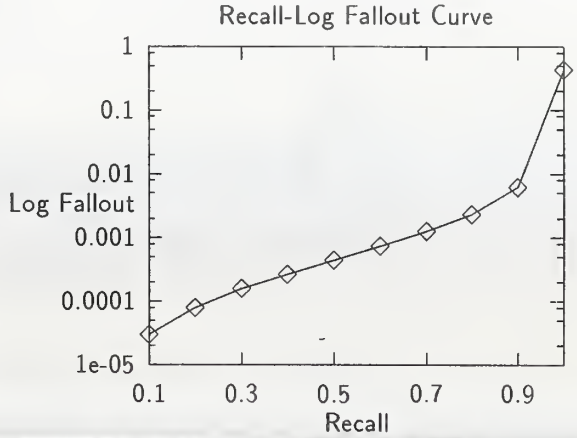
Recall Level Precision Averages	
Recall	Precision
0.00	0.8278
0.10	0.6888
0.20	0.5864
0.30	0.5171
0.40	0.4516
0.50	0.3835
0.60	0.3109
0.70	0.2350
0.80	0.1608
0.90	0.0744
1.00	0.0013

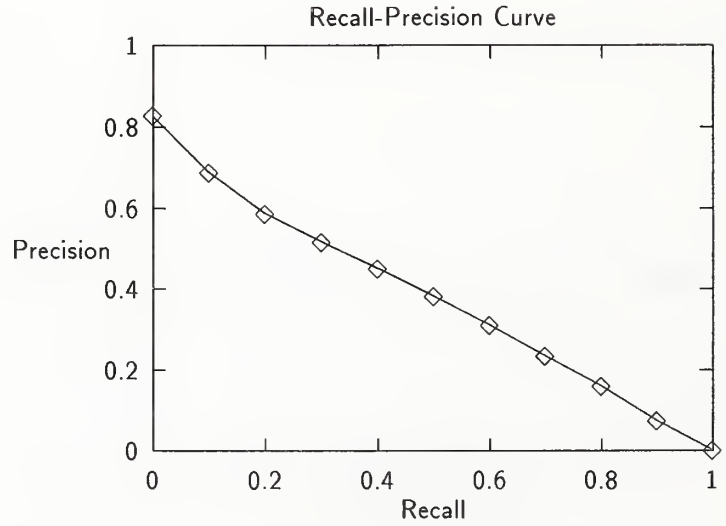
Average precision over all relevant docs	
non-interpolated	0.3725

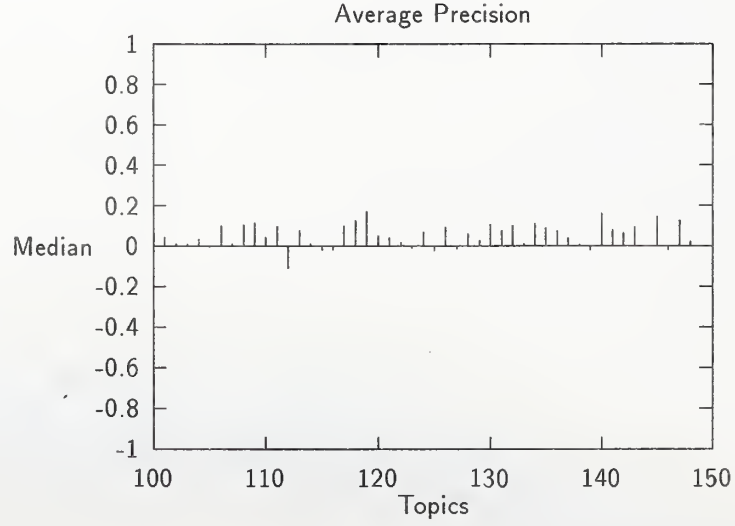
Document Level Averages	
	Precision
At 5 docs	0.6640
At 10 docs	0.6380
At 15 docs	0.6240
At 20 docs	0.5880
At 30 docs	0.5593
At 100 docs	0.4478
At 200 docs	0.3669
At 500 docs	0.2348
At 1000 docs	0.1443

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4013

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00008
0.30	0.00016
0.40	0.00027
0.50	0.00045
0.60	0.00074
0.70	0.00127
0.80	0.00232
0.90	0.00623
1.00	0.42754







Summary Statistics	
Run Number	crnlRR–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	7134

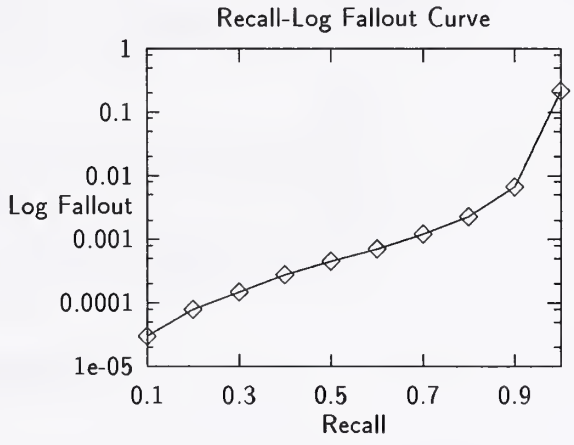
Recall Level Precision Averages	
Recall	Precision
0.00	0.8270
0.10	0.6712
0.20	0.5803
0.30	0.5191
0.40	0.4460
0.50	0.3750
0.60	0.3210
0.70	0.2416
0.80	0.1632
0.90	0.0698
1.00	0.0026

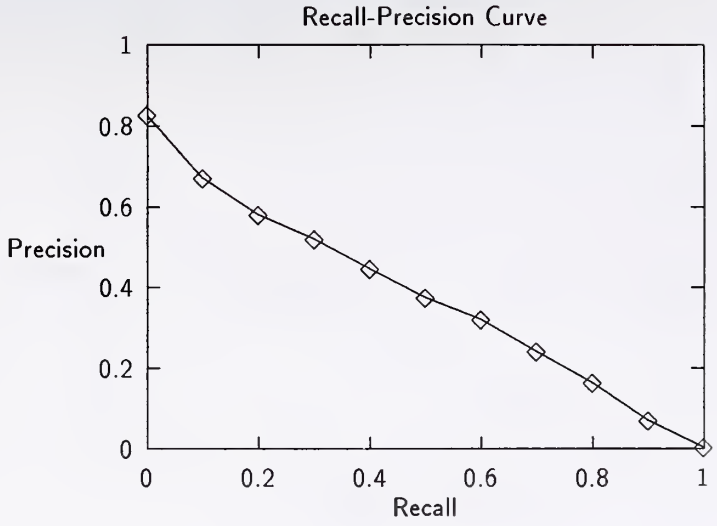
Average precision over all relevant docs	
non-interpolated	0.3699

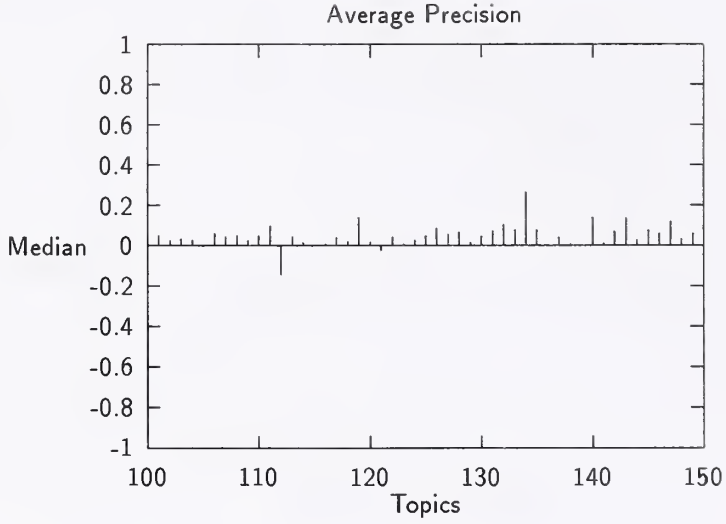
Document Level Averages	
	Precision
At 5 docs	0.6520
At 10 docs	0.6240
At 15 docs	0.6000
At 20 docs	0.5750
At 30 docs	0.5433
At 100 docs	0.4424
At 200 docs	0.3592
At 500 docs	0.2342
At 1000 docs	0.1427

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4064

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00008
0.30	0.00015
0.40	0.00028
0.50	0.00046
0.60	0.00071
0.70	0.00122
0.80	0.00228
0.90	0.00667
1.00	0.21349







Summary Statistics	
Run Number	dortR1–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	7265

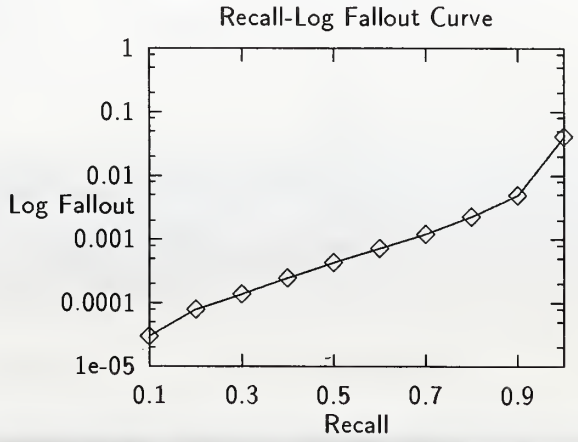
Recall Level Precision Averages	
Recall	Precision
0.00	0.8204
0.10	0.6823
0.20	0.5902
0.30	0.5371
0.40	0.4679
0.50	0.3891
0.60	0.3168
0.70	0.2441
0.80	0.1647
0.90	0.0916
1.00	0.0134

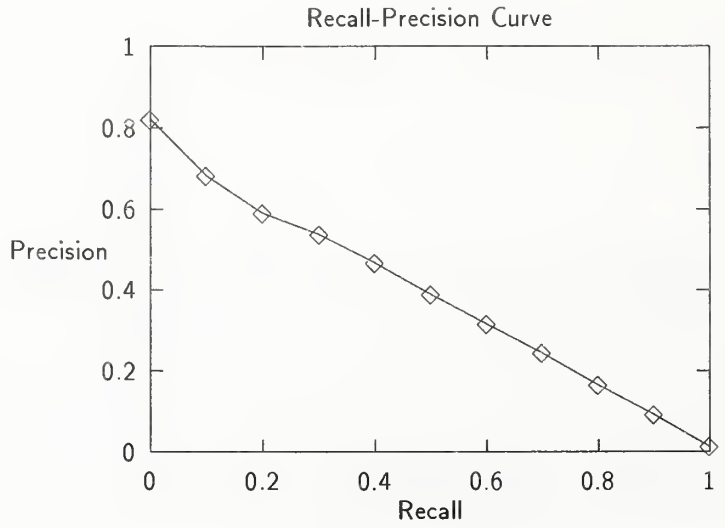
Average precision over all relevant docs	
non-interpolated	0.3825

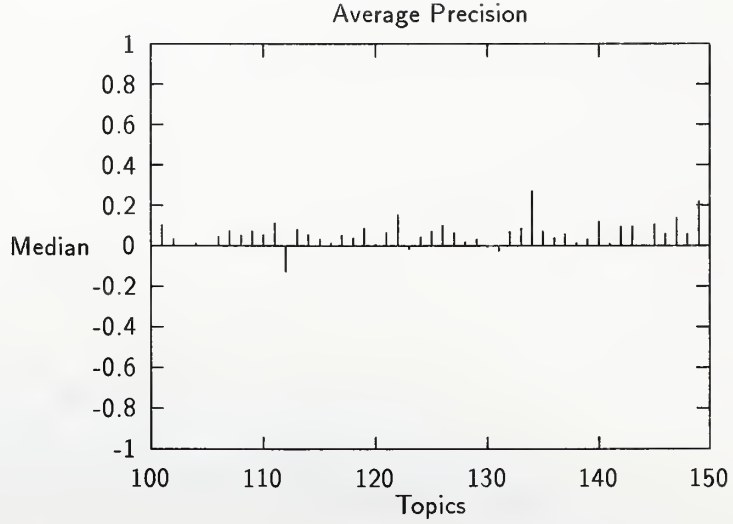
Document Level Averages	
	Precision
At 5 docs	0.6600
At 10 docs	0.6340
At 15 docs	0.6120
At 20 docs	0.5990
At 30 docs	0.5733
At 100 docs	0.4584
At 200 docs	0.3692
At 500 docs	0.2351
At 1000 docs	0.1453

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4129

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00008
0.30	0.00014
0.40	0.00025
0.50	0.00044
0.60	0.00072
0.70	0.00121
0.80	0.00226
0.90	0.00497
1.00	0.04098





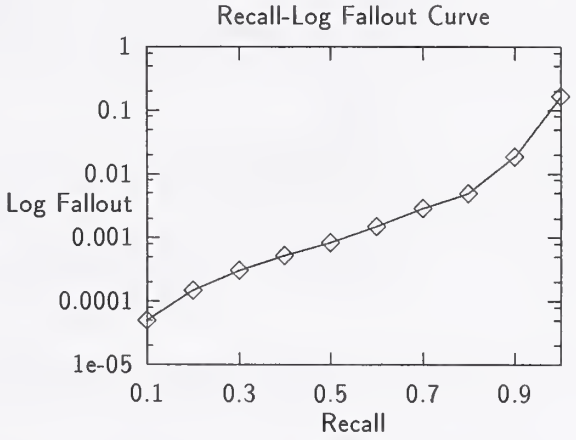


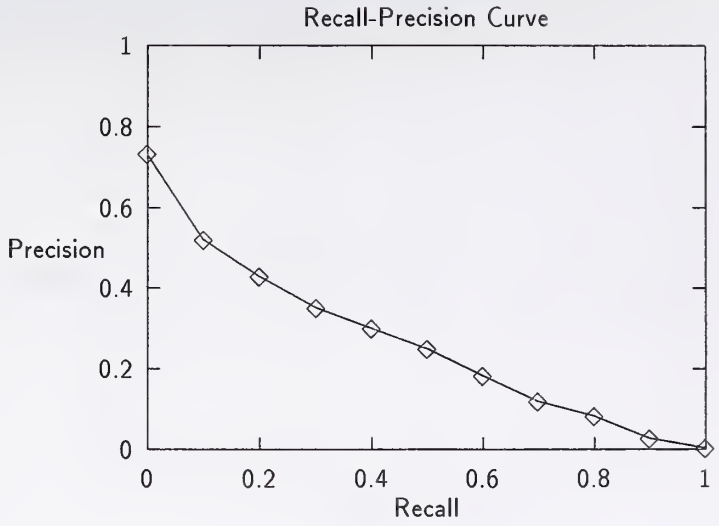
Summary Statistics	
Run Number	erimr1–category A, automatic
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	5441

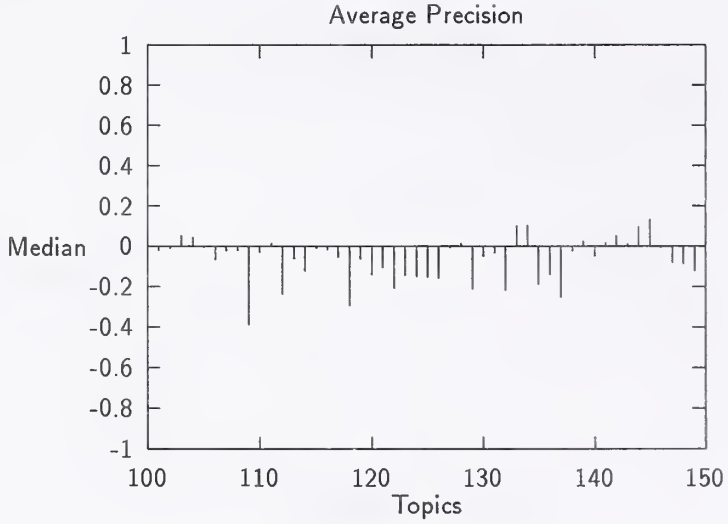
Recall Level Precision Averages	
Recall	Precision
0.00	0.7325
0.10	0.5201
0.20	0.4296
0.30	0.3513
0.40	0.3006
0.50	0.2490
0.60	0.1834
0.70	0.1185
0.80	0.0823
0.90	0.0265
1.00	0.0033
Average precision over all relevant docs	
non-interpolated	0.2528

Document Level Averages	
	Precision
At 5 docs	0.5000
At 10 docs	0.4800
At 15 docs	0.4587
At 20 docs	0.4390
At 30 docs	0.4133
At 100 docs	0.3360
At 200 docs	0.2717
At 500 docs	0.1738
At 1000 docs	0.1088
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3101

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00005
0.20	0.00015
0.30	0.00031
0.40	0.00052
0.50	0.00084
0.60	0.00149
0.70	0.00290
0.80	0.00496
0.90	0.01840
1.00	0.16809







Summary Statistics	
Run Number	ETH003–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	6242

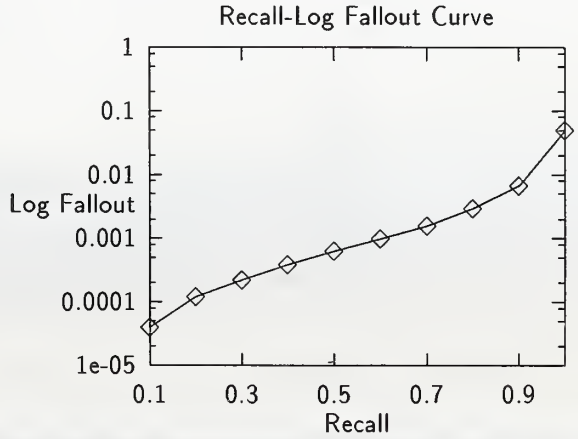
Recall Level Precision Averages	
Recall	Precision
0.00	0.7780
0.10	0.5742
0.20	0.4903
0.30	0.4265
0.40	0.3614
0.50	0.3039
0.60	0.2551
0.70	0.1981
0.80	0.1301
0.90	0.0694
1.00	0.0110

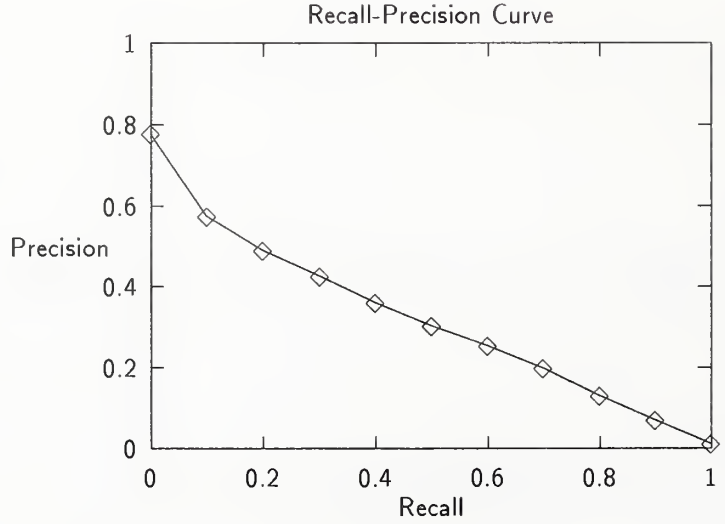
Average precision over all relevant docs	
non-interpolated	0.3092

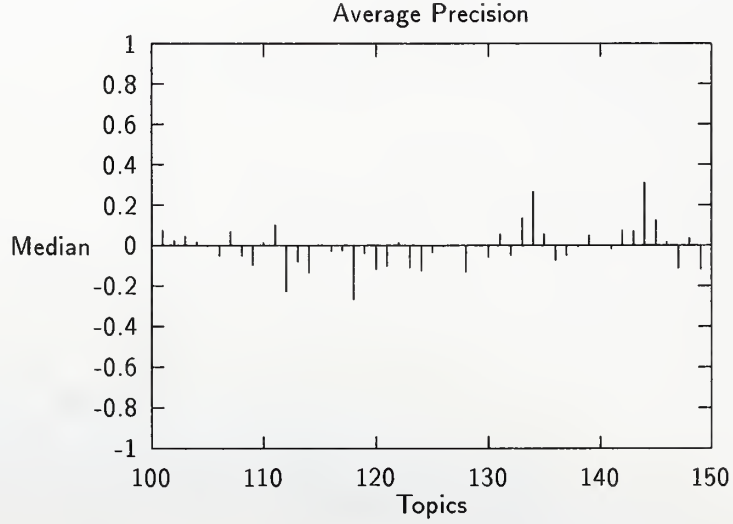
Document Level Averages	
	Precision
At 5 docs	0.5640
At 10 docs	0.5180
At 15 docs	0.4987
At 20 docs	0.4690
At 30 docs	0.4527
At 100 docs	0.3692
At 200 docs	0.3097
At 500 docs	0.2015
At 1000 docs	0.1248

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3440

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00004
0.20	0.00012
0.30	0.00022
0.40	0.00039
0.50	0.00064
0.60	0.00098
0.70	0.00158
0.80	0.00298
0.90	0.00672
1.00	0.05004







Summary Statistics	
Run Number	ETH004–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	6295

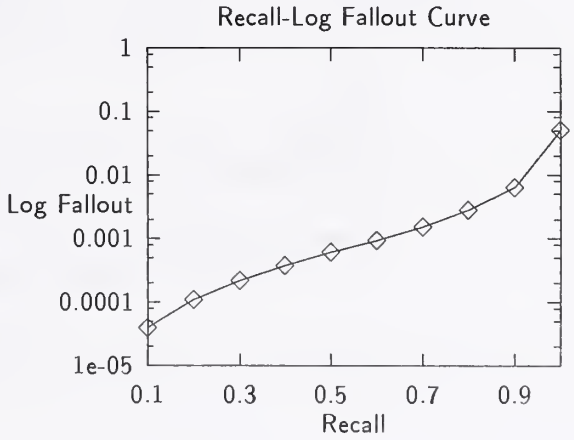
Recall Level Precision Averages	
Recall	Precision
0.00	0.7648
0.10	0.5732
0.20	0.5020
0.30	0.4285
0.40	0.3678
0.50	0.3092
0.60	0.2601
0.70	0.2023
0.80	0.1361
0.90	0.0729
1.00	0.0109

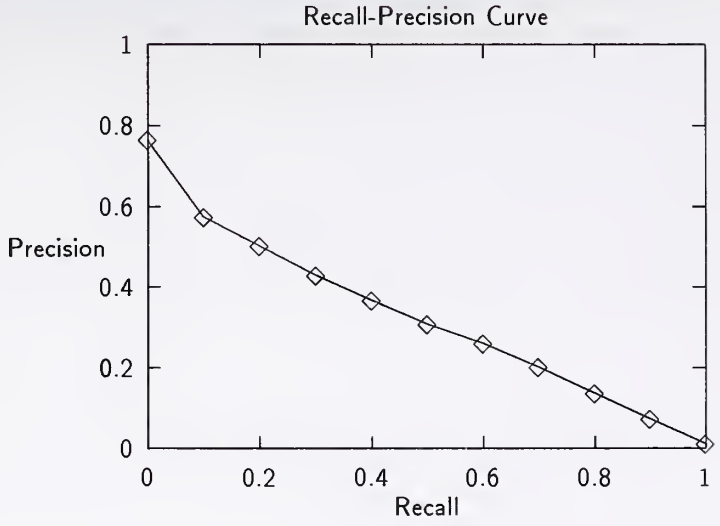
Average precision over all relevant docs	
non-interpolated	0.3154

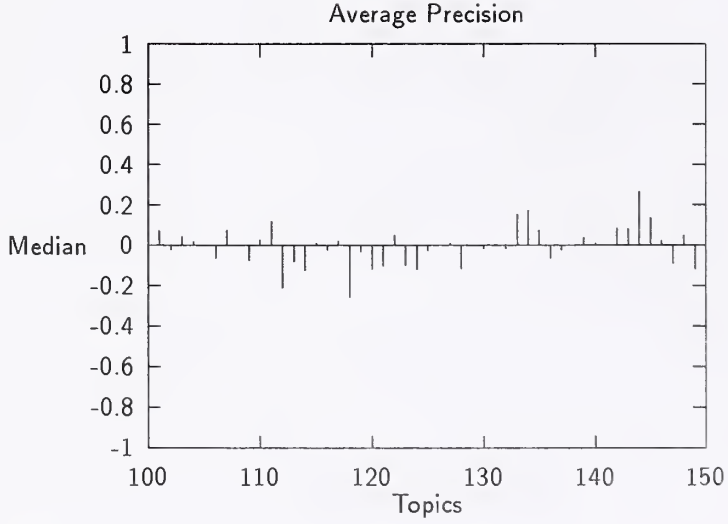
Document Level Averages	
	Precision
At 5 docs	0.6160
At 10 docs	0.5680
At 15 docs	0.5280
At 20 docs	0.5040
At 30 docs	0.4687
At 100 docs	0.3766
At 200 docs	0.3150
At 500 docs	0.2041
At 1000 docs	0.1259

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3468

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00004
0.20	0.00011
0.30	0.00022
0.40	0.00038
0.50	0.00062
0.60	0.00095
0.70	0.00154
0.80	0.00283
0.90	0.00637
1.00	0.05050







Summary Statistics	
Run Number	FDF1–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	6669

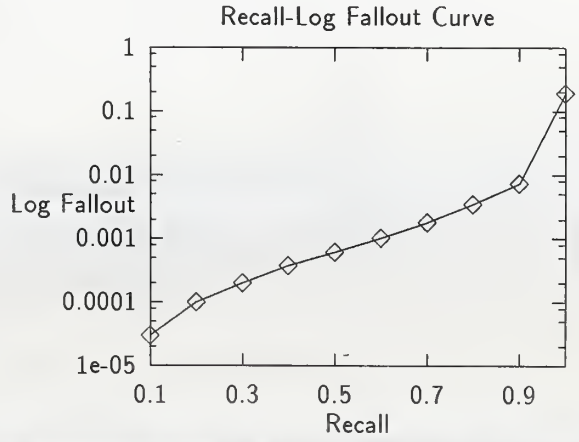
Recall Level Precision Averages	
Recall	Precision
0.00	0.7718
0.10	0.6157
0.20	0.5245
0.30	0.4506
0.40	0.3721
0.50	0.3110
0.60	0.2448
0.70	0.1764
0.80	0.1135
0.90	0.0641
1.00	0.0029

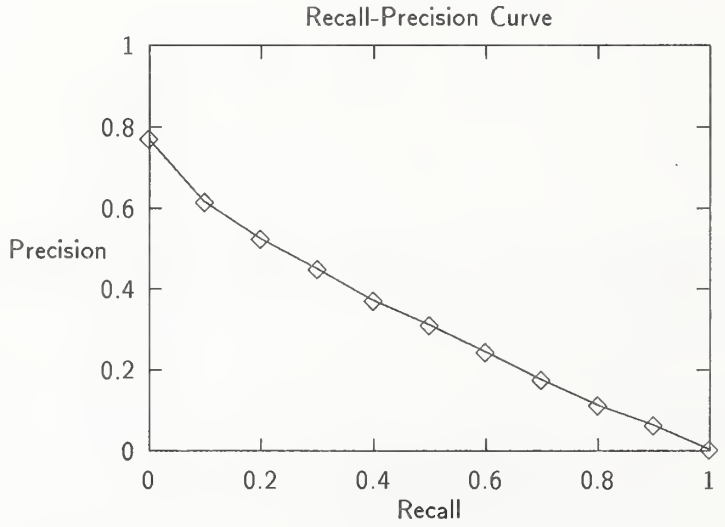
Average precision over all relevant docs	
non-interpolated	0.3156

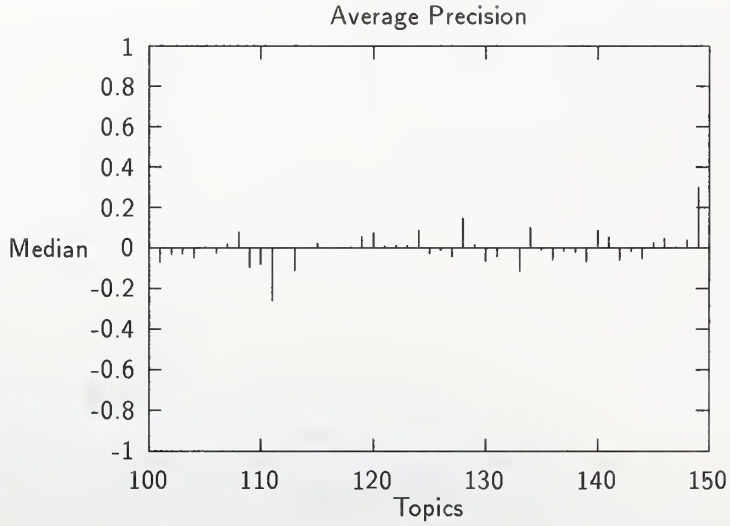
Document Level Averages	
	Precision
At 5 docs	0.5880
At 10 docs	0.6060
At 15 docs	0.5747
At 20 docs	0.5590
At 30 docs	0.5287
At 100 docs	0.4110
At 200 docs	0.3327
At 500 docs	0.2082
At 1000 docs	0.1334

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3540

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00010
0.30	0.00020
0.40	0.00038
0.50	0.00062
0.60	0.00103
0.70	0.00182
0.80	0.00348
0.90	0.00731
1.00	0.19135







Summary Statistics	
Run Number	FDF2–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	6706

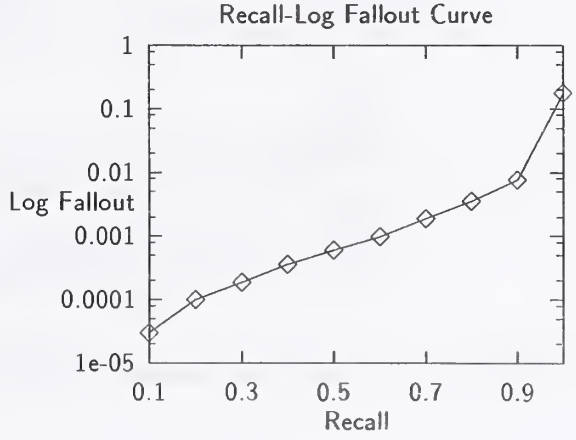
Recall Level Precision Averages	
Recall	Precision
0.00	0.7737
0.10	0.6193
0.20	0.5324
0.30	0.4632
0.40	0.3802
0.50	0.3123
0.60	0.2538
0.70	0.1711
0.80	0.1114
0.90	0.0613
1.00	0.0031

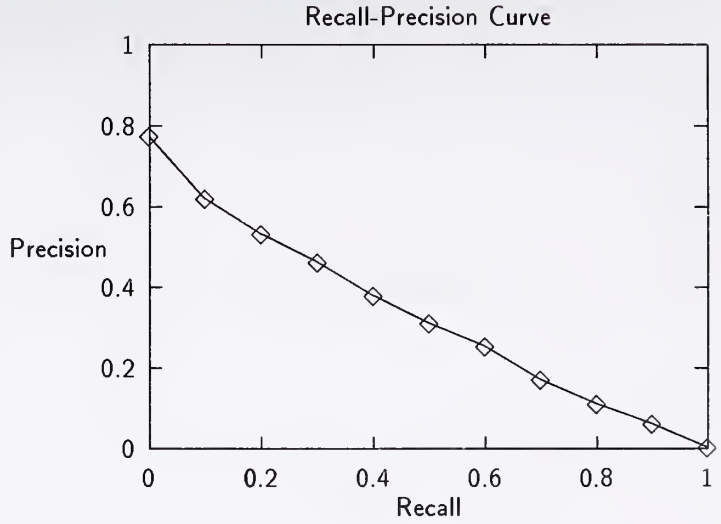
Average precision over all relevant docs	
non-interpolated	0.3189

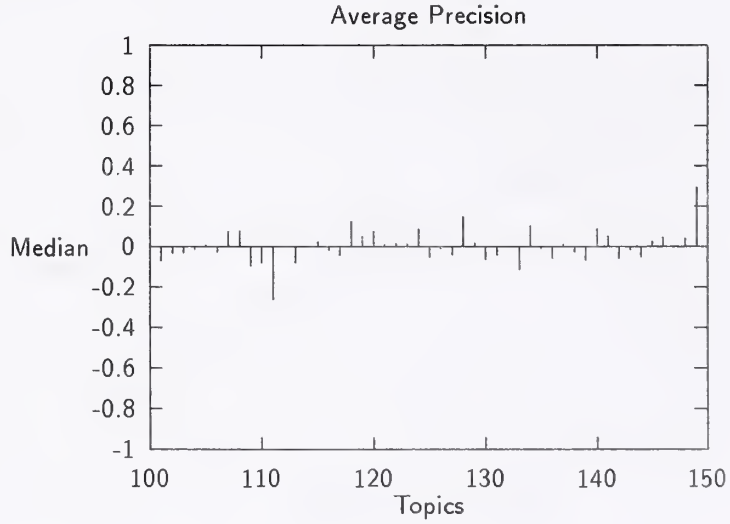
Document Level Averages	
	Precision
At 5 docs	0.5960
At 10 docs	0.6120
At 15 docs	0.5813
At 20 docs	0.5580
At 30 docs	0.5280
At 100 docs	0.4154
At 200 docs	0.3379
At 500 docs	0.2103
At 1000 docs	0.1341

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3556

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00010
0.30	0.00019
0.40	0.00036
0.50	0.00061
0.60	0.00098
0.70	0.00189
0.80	0.00355
0.90	0.00767
1.00	0.17897







Summary Statistics	
Run Number	INQ103–category A, automatic
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	7454

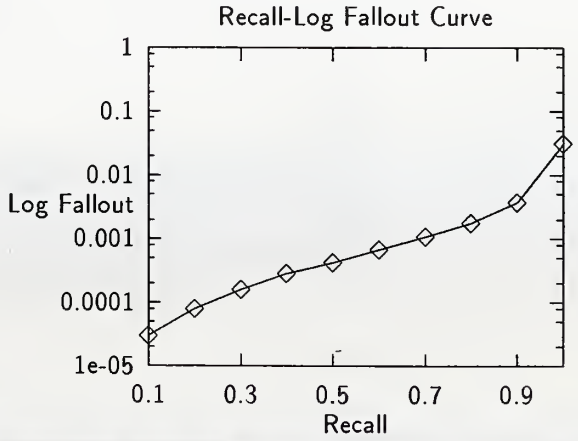
Recall Level Precision Averages	
Recall	Precision
0.00	0.8224
0.10	0.6655
0.20	0.5837
0.30	0.5113
0.40	0.4385
0.50	0.3922
0.60	0.3281
0.70	0.2641
0.80	0.2006
0.90	0.1179
1.00	0.0176

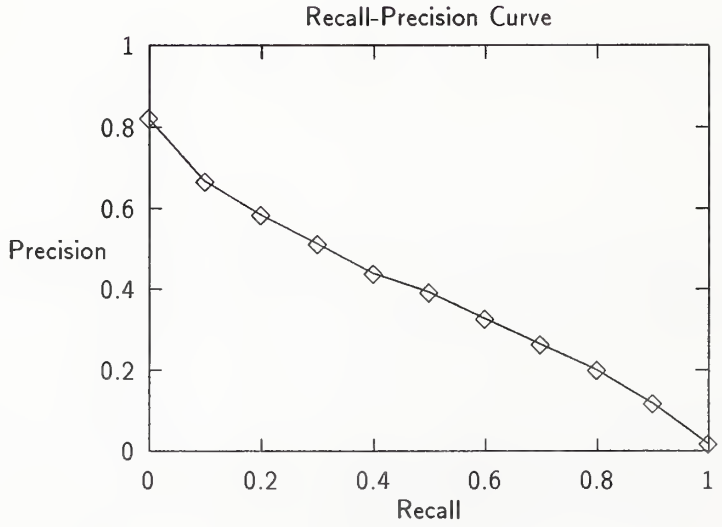
Average precision over all relevant docs	
non-interpolated	0.3838

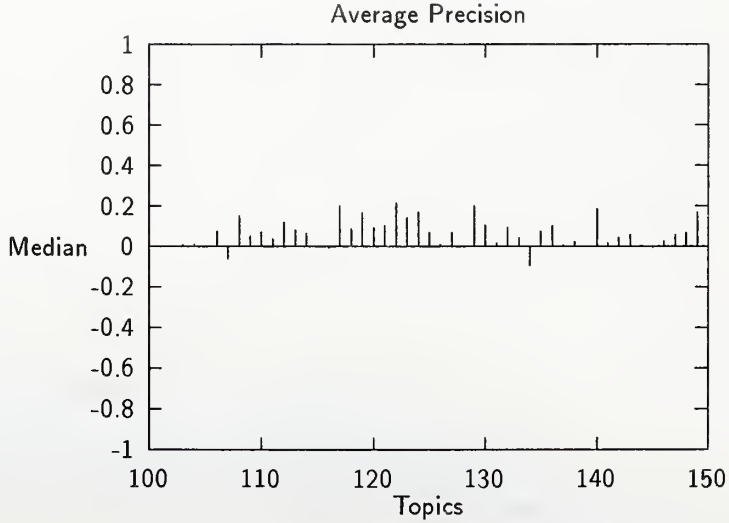
Document Level Averages	
	Precision
At 5 docs	0.6720
At 10 docs	0.6400
At 15 docs	0.6187
At 20 docs	0.6020
At 30 docs	0.5773
At 100 docs	0.4632
At 200 docs	0.3752
At 500 docs	0.2409
At 1000 docs	0.1491

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3971

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00008
0.30	0.00016
0.40	0.00029
0.50	0.00043
0.60	0.00068
0.70	0.00109
0.80	0.00177
0.90	0.00375
1.00	0.03106







Summary Statistics	
Run Number	INQ104–category A, manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	7690

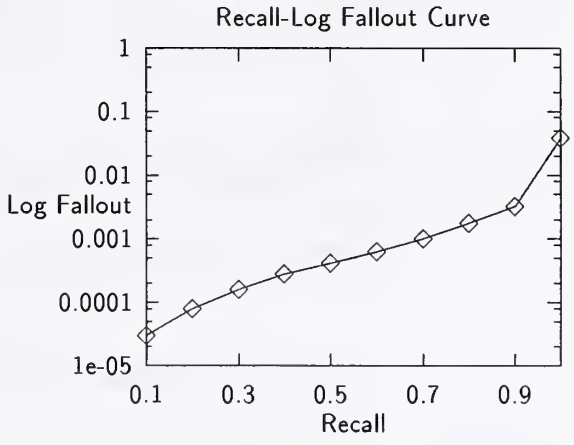
Recall Level Precision Averages	
Recall	Precision
0.00	0.8029
0.10	0.6576
0.20	0.5759
0.30	0.5055
0.40	0.4436
0.50	0.3988
0.60	0.3417
0.70	0.2772
0.80	0.2031
0.90	0.1355
1.00	0.0142

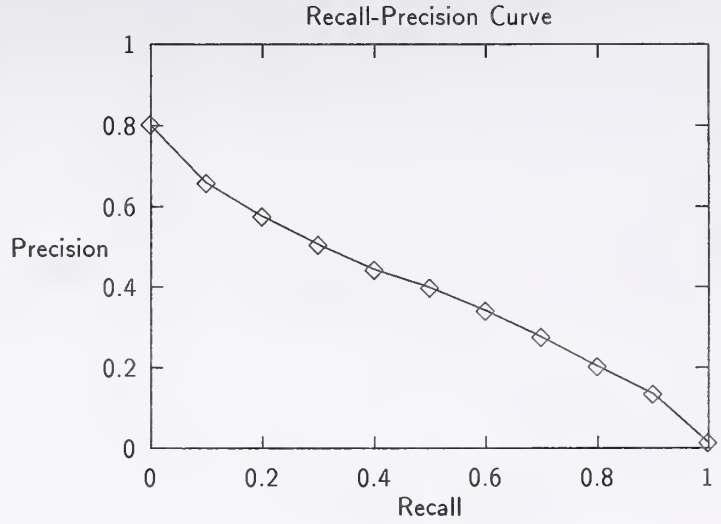
Average precision over all relevant docs	
non-interpolated	0.3880

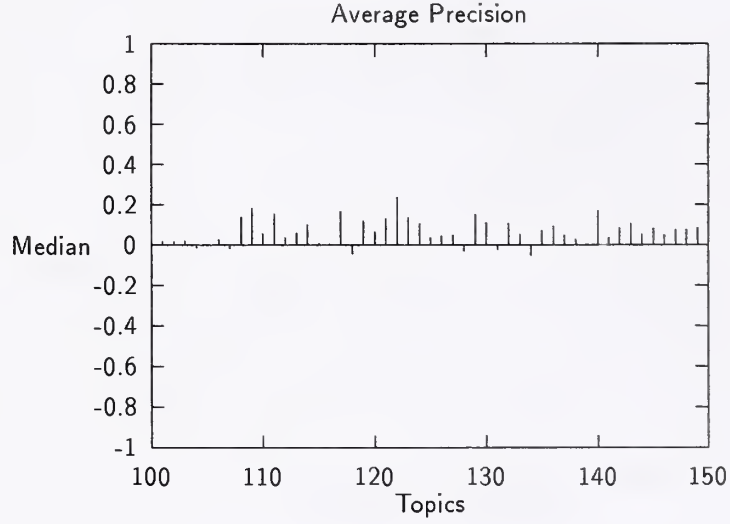
Document Level Averages	
	Precision
At 5 docs	0.6600
At 10 docs	0.6200
At 15 docs	0.5960
At 20 docs	0.5840
At 30 docs	0.5633
At 100 docs	0.4492
At 200 docs	0.3690
At 500 docs	0.2443
At 1000 docs	0.1538

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4048

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00008
0.30	0.00016
0.40	0.00028
0.50	0.00042
0.60	0.00064
0.70	0.00102
0.80	0.00175
0.90	0.00320
1.00	0.03864







Summary Statistics	
Run Number	losPA1–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	6756

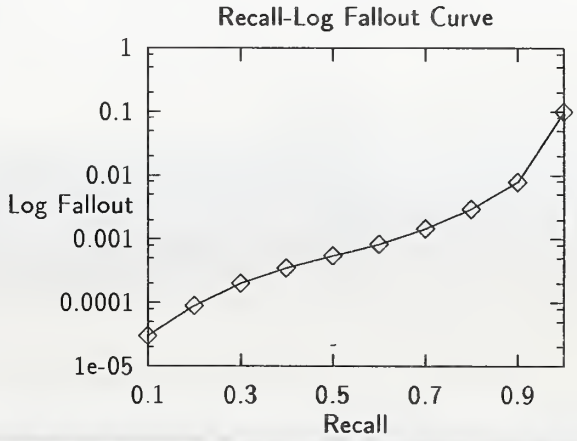
Recall Level Precision Averages	
Recall	Precision
0.00	0.8387
0.10	0.6539
0.20	0.5450
0.30	0.4576
0.40	0.3885
0.50	0.3344
0.60	0.2877
0.70	0.2122
0.80	0.1316
0.90	0.0606
1.00	0.0055

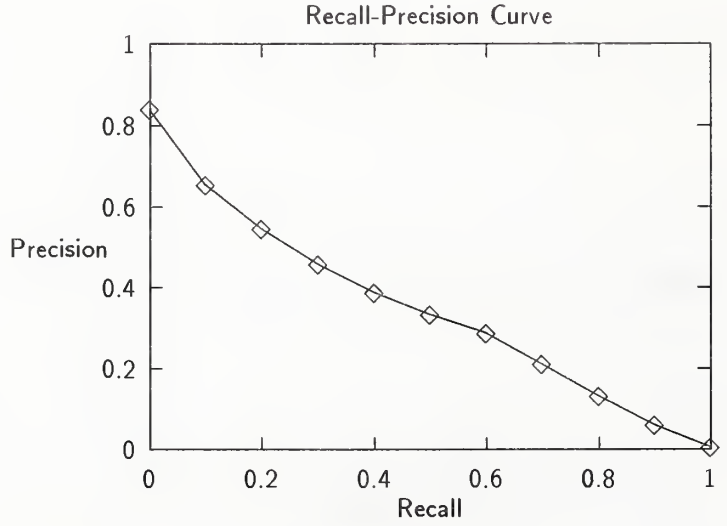
Average precision over all relevant docs	
non-interpolated	0.3373

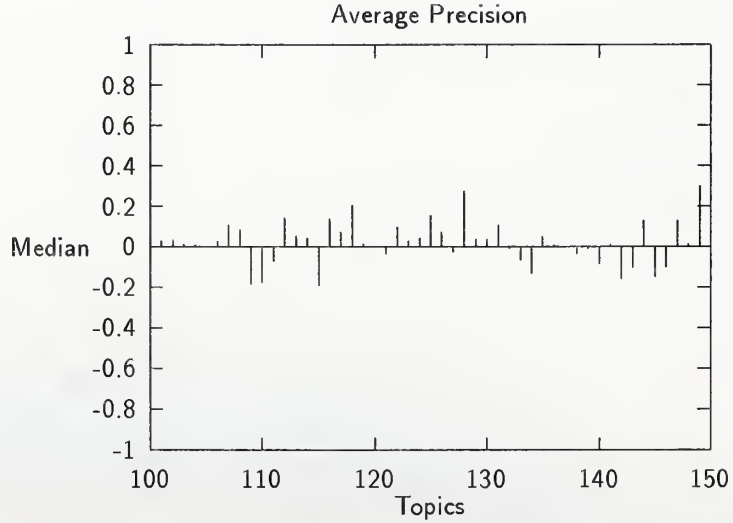
Document Level Averages	
	Precision
At 5 docs	0.5840
At 10 docs	0.5760
At 15 docs	0.5747
At 20 docs	0.5560
At 30 docs	0.5253
At 100 docs	0.4288
At 200 docs	0.3540
At 500 docs	0.2168
At 1000 docs	0.1351

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3725

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00009
0.30	0.00020
0.40	0.00035
0.50	0.00055
0.60	0.00083
0.70	0.00145
0.80	0.00294
0.90	0.00776
1.00	0.10063







Summary Statistics	
Run Number	lsir1–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	6252

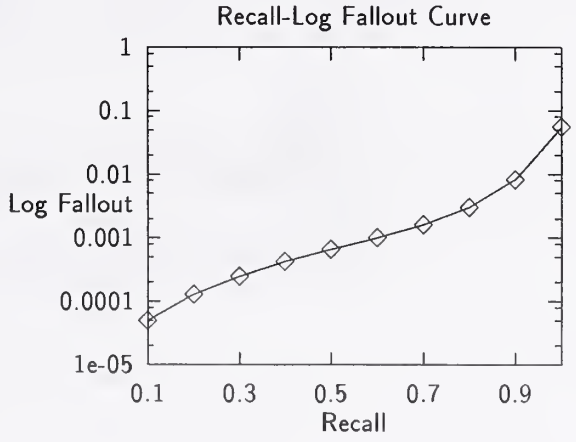
Recall Level Precision Averages	
Recall	Precision
0.00	0.6617
0.10	0.5260
0.20	0.4657
0.30	0.3998
0.40	0.3404
0.50	0.2952
0.60	0.2495
0.70	0.1958
0.80	0.1275
0.90	0.0579
1.00	0.0098

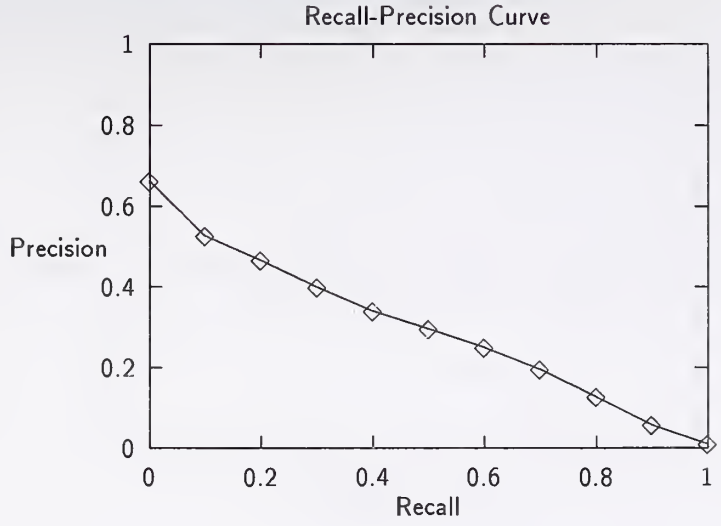
Average precision over all relevant docs	
non-interpolated	0.2880

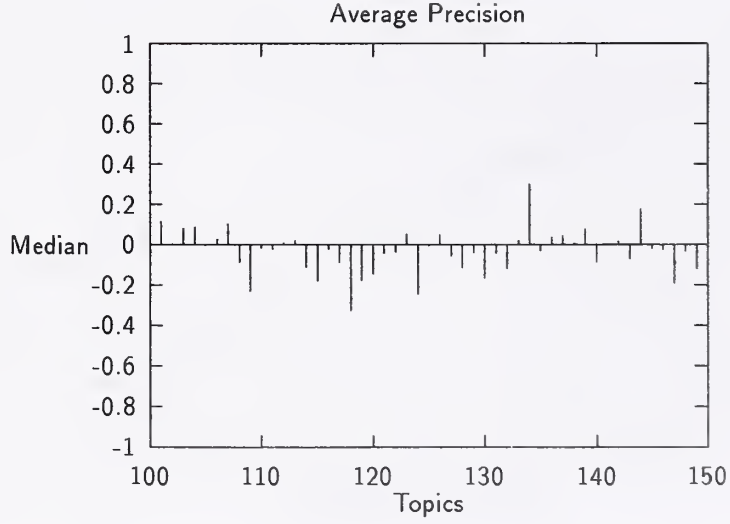
Document Level Averages	
	Precision
At 5 docs	0.4560
At 10 docs	0.4620
At 15 docs	0.4520
At 20 docs	0.4450
At 30 docs	0.4307
At 100 docs	0.3532
At 200 docs	0.2918
At 500 docs	0.1932
At 1000 docs	0.1250

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3386

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00005
0.20	0.00013
0.30	0.00025
0.40	0.00043
0.50	0.00066
0.60	0.00100
0.70	0.00160
0.80	0.00305
0.90	0.00815
1.00	0.05623







Summary Statistics	
Run Number	lsir2–category A, automatic
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	6878

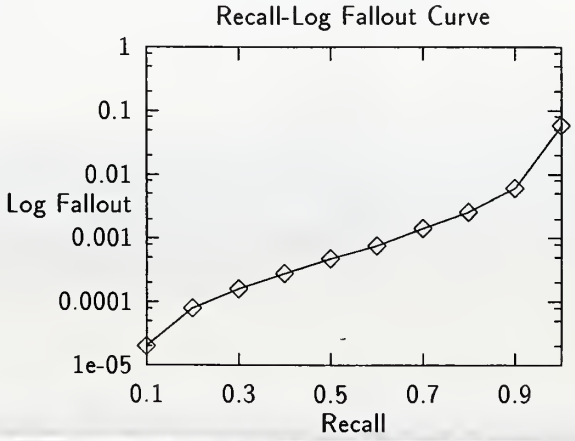
Recall Level Precision Averages	
Recall	Precision
0.00	0.8738
0.10	0.7250
0.20	0.5954
0.30	0.5034
0.40	0.4401
0.50	0.3653
0.60	0.3028
0.70	0.2153
0.80	0.1481
0.90	0.0764
1.00	0.0094

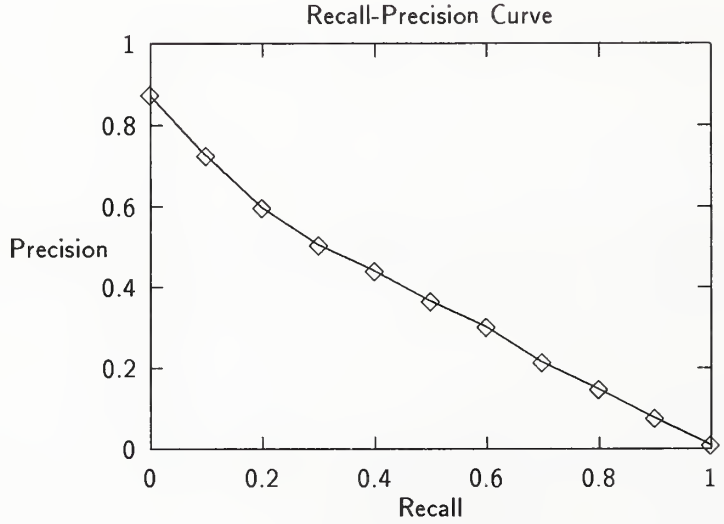
Average precision over all relevant docs	
non-interpolated	0.3737

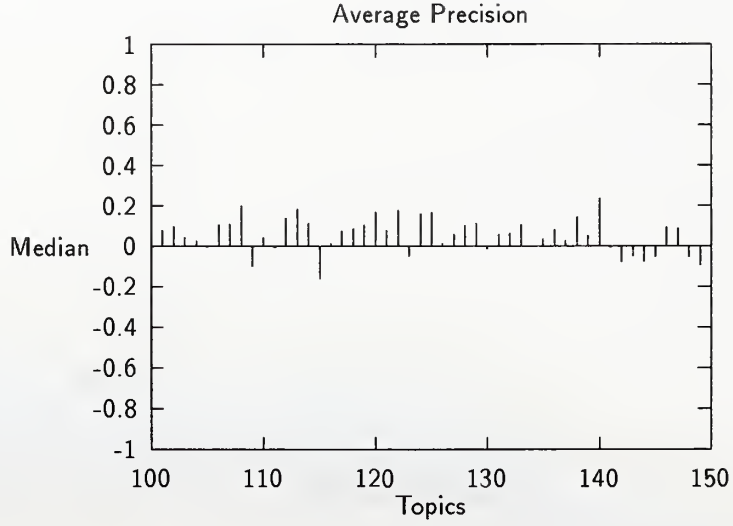
Document Level Averages	
	Precision
At 5 docs	0.7160
At 10 docs	0.6720
At 15 docs	0.6600
At 20 docs	0.6350
At 30 docs	0.5900
At 100 docs	0.4544
At 200 docs	0.3552
At 500 docs	0.2210
At 1000 docs	0.1376

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3950

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00002
0.20	0.00008
0.30	0.00016
0.40	0.00028
0.50	0.00048
0.60	0.00077
0.70	0.00142
0.80	0.00256
0.90	0.00606
1.00	0.05865







Summary Statistics	
Run Number	nyuir1–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	6350

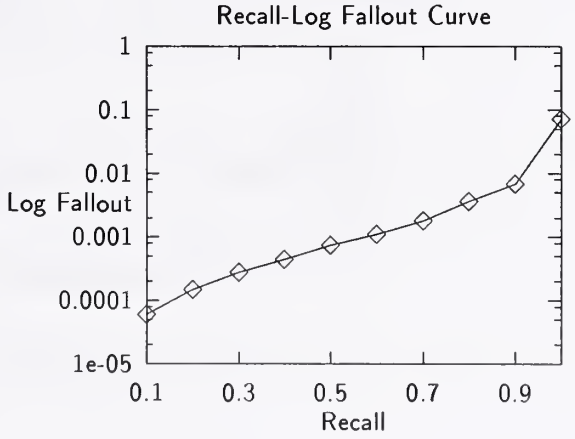
Recall Level Precision Averages	
Recall	Precision
0.00	0.7401
0.10	0.5003
0.20	0.4255
0.30	0.3719
0.40	0.3330
0.50	0.2723
0.60	0.2327
0.70	0.1765
0.80	0.1094
0.90	0.0691
1.00	0.0077

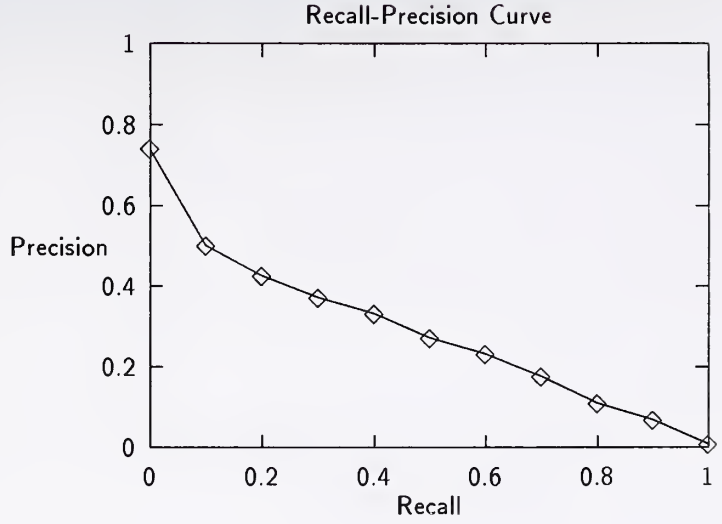
Average precision over all relevant docs	
non-interpolated	0.2743

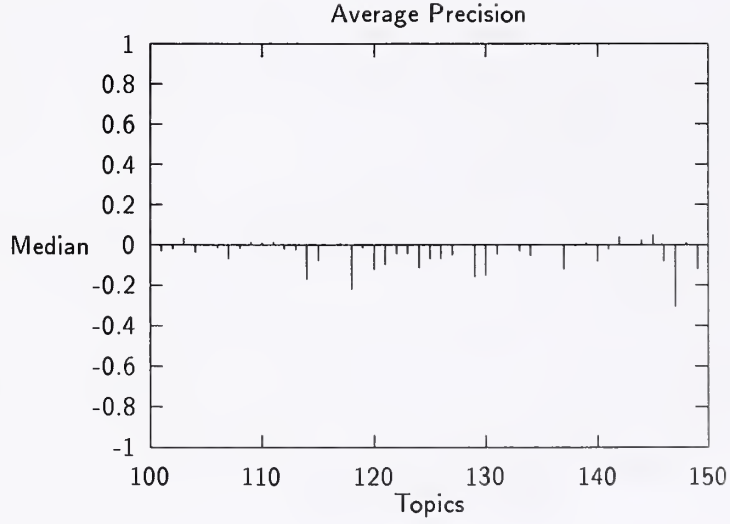
Document Level Averages	
	Precision
At 5 docs	0.5280
At 10 docs	0.4800
At 15 docs	0.4600
At 20 docs	0.4390
At 30 docs	0.4273
At 100 docs	0.3584
At 200 docs	0.3063
At 500 docs	0.2008
At 1000 docs	0.1270

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3135

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00006
0.20	0.00015
0.30	0.00028
0.40	0.00045
0.50	0.00074
0.60	0.00110
0.70	0.00182
0.80	0.00362
0.90	0.00675
1.00	0.07172







Summary Statistics	
Run Number	nyuir2–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	7203

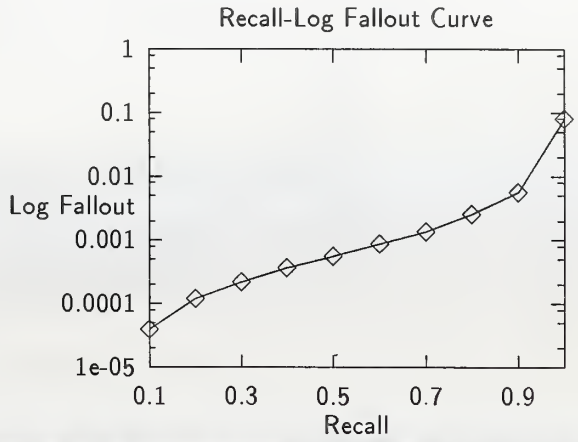
Recall Level Precision Averages	
Recall	Precision
0.00	0.7711
0.10	0.5957
0.20	0.4857
0.30	0.4337
0.40	0.3758
0.50	0.3321
0.60	0.2762
0.70	0.2213
0.80	0.1480
0.90	0.0816
1.00	0.0069

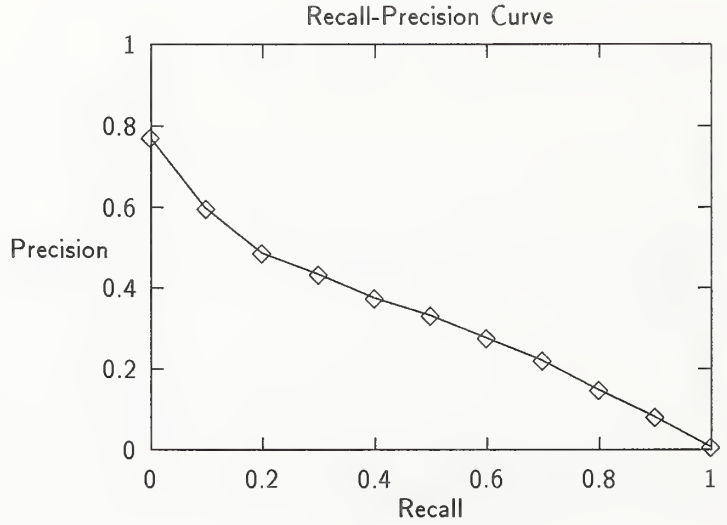
Average precision over all relevant docs	
non-interpolated	0.3244

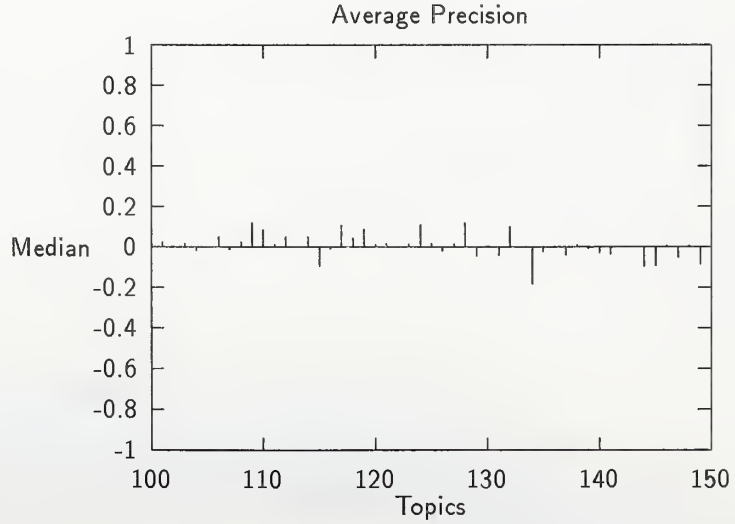
Document Level Averages	
	Precision
At 5 docs	0.6000
At 10 docs	0.5560
At 15 docs	0.5333
At 20 docs	0.5200
At 30 docs	0.4940
At 100 docs	0.4098
At 200 docs	0.3380
At 500 docs	0.2260
At 1000 docs	0.1441

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3510

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00004
0.20	0.00012
0.30	0.00022
0.40	0.00037
0.50	0.00056
0.60	0.00088
0.70	0.00137
0.80	0.00256
0.90	0.00564
1.00	0.08010







Summary Statistics	
Run Number	pircs3–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	7640

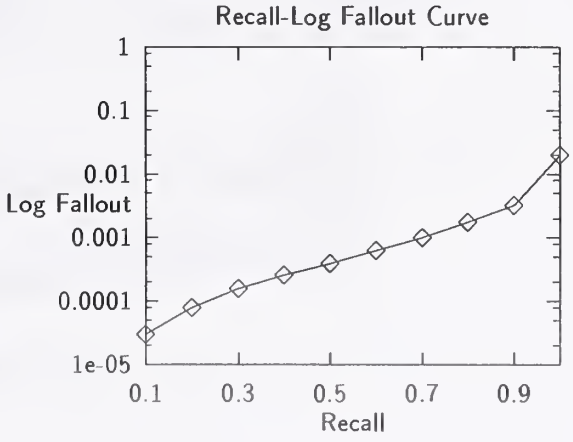
Recall Level Precision Averages	
Recall	Precision
0.00	0.8088
0.10	0.6654
0.20	0.5776
0.30	0.5137
0.40	0.4597
0.50	0.4074
0.60	0.3419
0.70	0.2755
0.80	0.2007
0.90	0.1332
1.00	0.0271

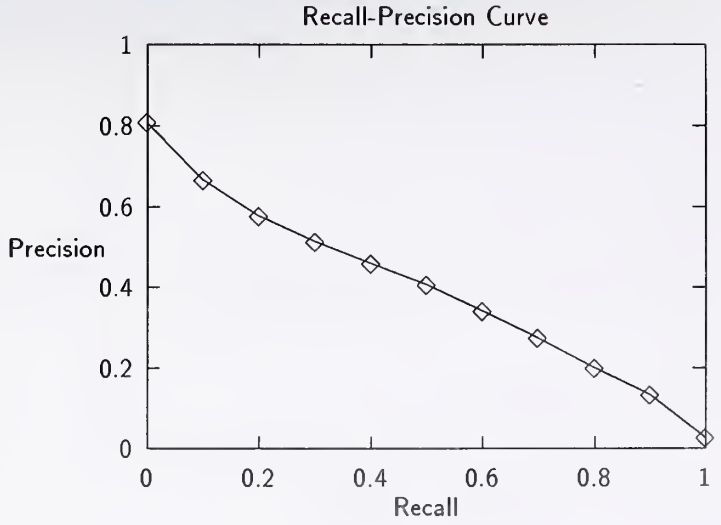
Average precision over all relevant docs	
non-interpolated	0.3887

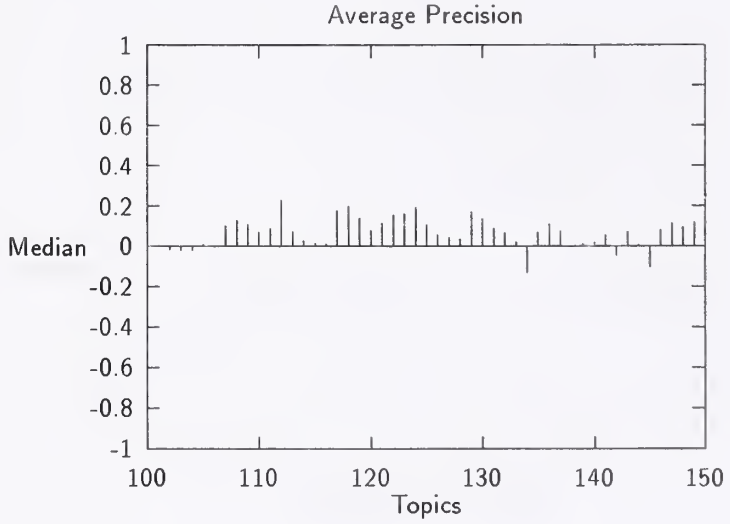
Document Level Averages	
	Precision
At 5 docs	0.6400
At 10 docs	0.6260
At 15 docs	0.6080
At 20 docs	0.5880
At 30 docs	0.5633
At 100 docs	0.4606
At 200 docs	0.3772
At 500 docs	0.2456
At 1000 docs	0.1528

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3991

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00008
0.30	0.00016
0.40	0.00026
0.50	0.00040
0.60	0.00064
0.70	0.00102
0.80	0.00177
0.90	0.00326
1.00	0.01998







Summary Statistics	
Run Number	pircs4–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	7318

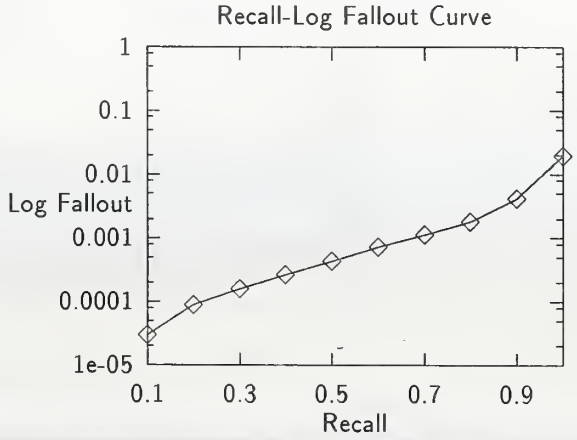
Recall Level Precision Averages	
Recall	Precision
0.00	0.7910
0.10	0.6583
0.20	0.5633
0.30	0.5044
0.40	0.4499
0.50	0.3899
0.60	0.3131
0.70	0.2552
0.80	0.1994
0.90	0.1058
1.00	0.0278

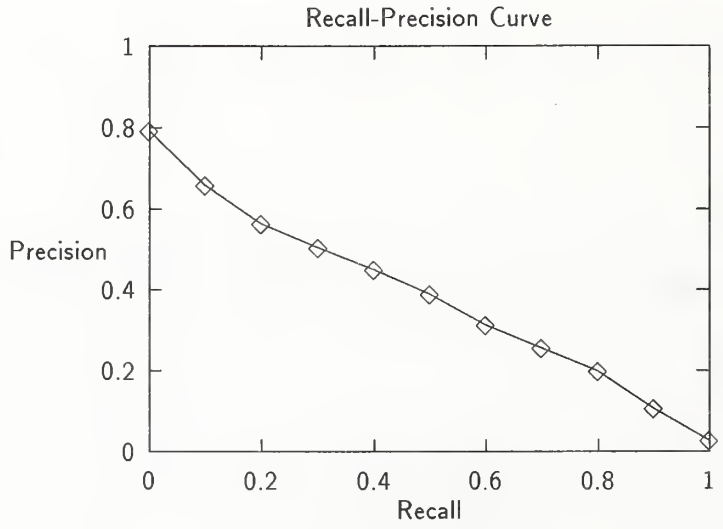
Average precision over all relevant docs	
non-interpolated	0.3749

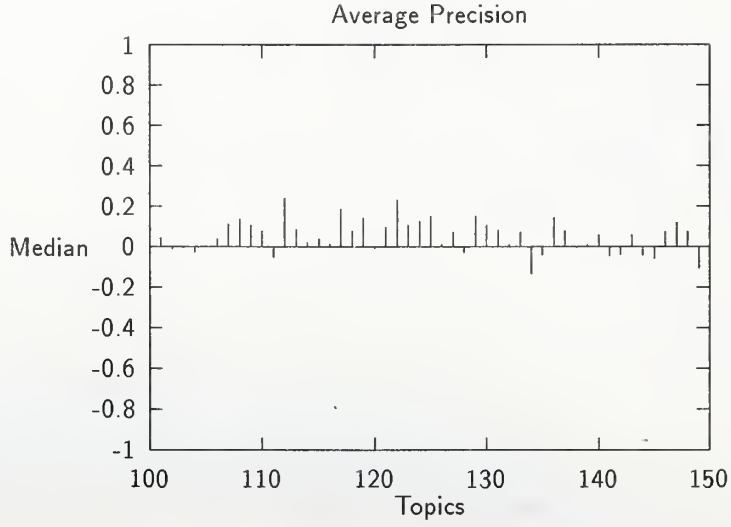
Document Level Averages	
	Precision
At 5 docs	0.6280
At 10 docs	0.6140
At 15 docs	0.5973
At 20 docs	0.5790
At 30 docs	0.5500
At 100 docs	0.4508
At 200 docs	0.3641
At 500 docs	0.2357
At 1000 docs	0.1464

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3899

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00009
0.30	0.00016
0.40	0.00027
0.50	0.00044
0.60	0.00073
0.70	0.00114
0.80	0.00179
0.90	0.00423
1.00	0.01946







Summary Statistics	
Run Number	pixtex–category A, manual
Number of Topics	11

Total number of documents over all topics	
Retrieved:	5791
Relevant:	1686
Rel_ret:	653

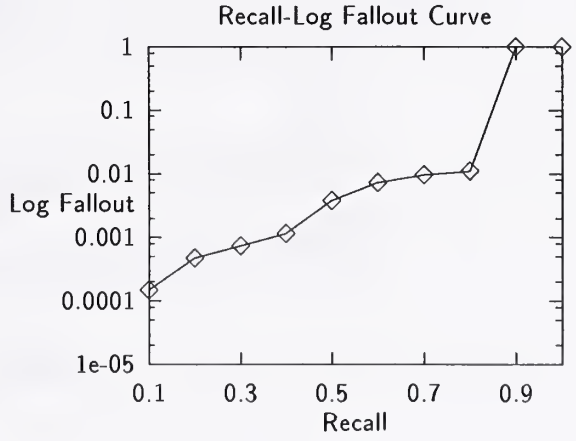
Recall Level Precision Averages	
Recall	Precision
0.00	0.3381
0.10	0.2697
0.20	0.1886
0.30	0.1815
0.40	0.1595
0.50	0.0685
0.60	0.0440
0.70	0.0384
0.80	0.0384
0.90	0.0000
1.00	0.0000

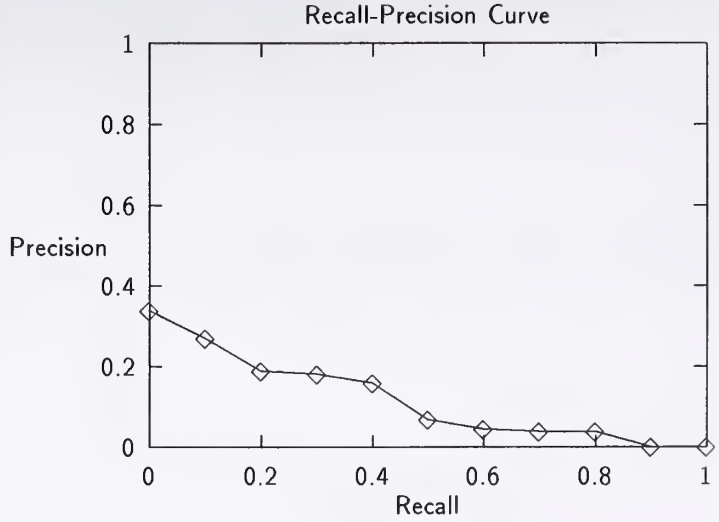
Average precision over all relevant docs	
non-interpolated	0.0947

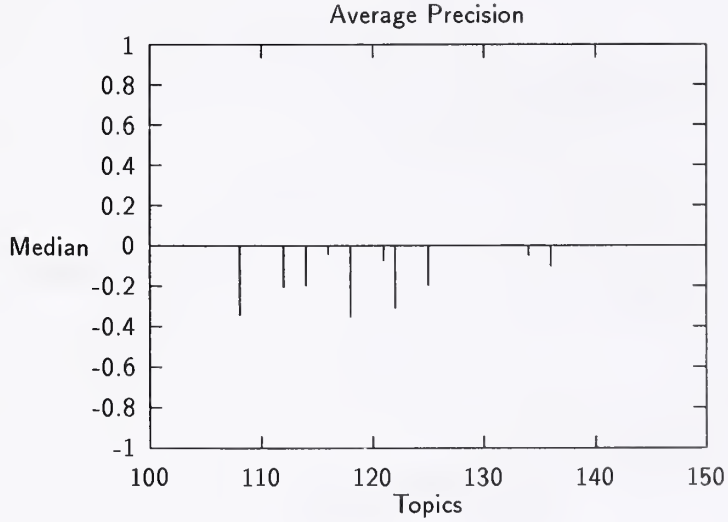
Document Level Averages	
	Precision
At 5 docs	0.0909
At 10 docs	0.1364
At 15 docs	0.1212
At 20 docs	0.1091
At 30 docs	0.1303
At 100 docs	0.1636
At 200 docs	0.1386
At 500 docs	0.0798
At 1000 docs	0.0594

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1510

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00015
0.20	0.00048
0.30	0.00075
0.40	0.00117
0.50	0.00378
0.60	0.00726
0.70	0.00976
0.80	0.01115
0.90	1.00000
1.00	1.00000







Summary Statistics	
Run Number	rutfur1–category A, manual
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	4647

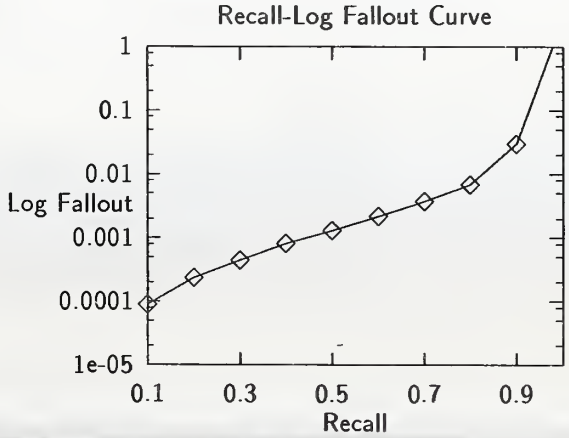
Recall Level Precision Averages	
Recall	Precision
0.00	0.7023
0.10	0.3916
0.20	0.3166
0.30	0.2711
0.40	0.2132
0.50	0.1760
0.60	0.1327
0.70	0.0946
0.80	0.0606
0.90	0.0167
1.00	0.0002

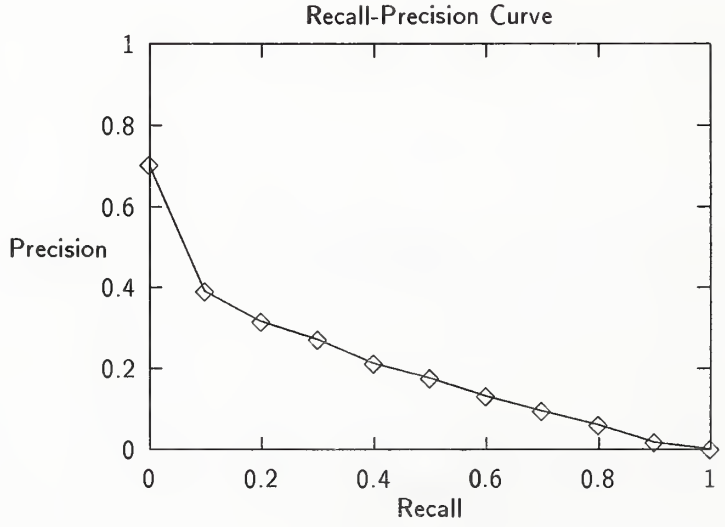
Average precision over all relevant docs	
non-interpolated	0.1854

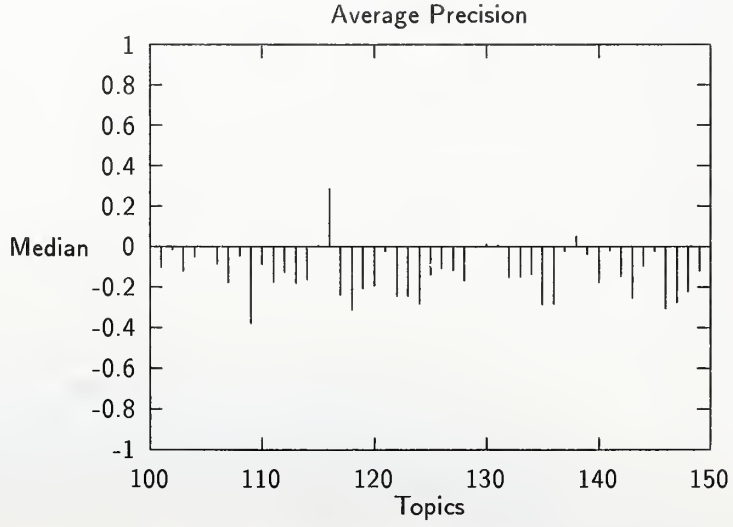
Document Level Averages	
	Precision
At 5 docs	0.4400
At 10 docs	0.3780
At 15 docs	0.3760
At 20 docs	0.3680
At 30 docs	0.3500
At 100 docs	0.2672
At 200 docs	0.2127
At 500 docs	0.1395
At 1000 docs	0.0929

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2413

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00009
0.20	0.00024
0.30	0.00045
0.40	0.00082
0.50	0.00130
0.60	0.00218
0.70	0.00373
0.80	0.00690
0.90	0.02949
1.00	2.78206







Summary Statistics	
Run Number	rutfur2–category A, manual
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	4645

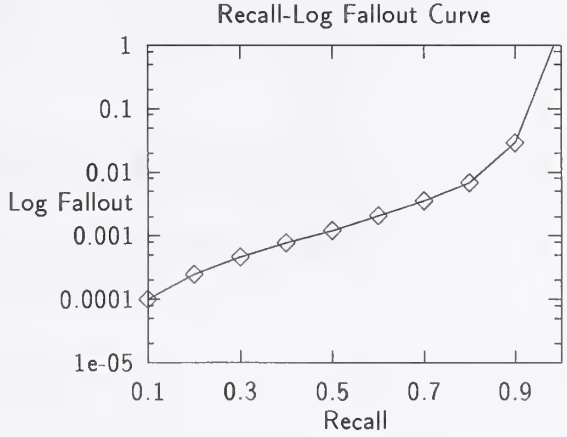
Recall Level Precision Averages	
Recall	Precision
0.00	0.6168
0.10	0.3655
0.20	0.3073
0.30	0.2612
0.40	0.2203
0.50	0.1847
0.60	0.1377
0.70	0.0992
0.80	0.0618
0.90	0.0170
1.00	0.0003

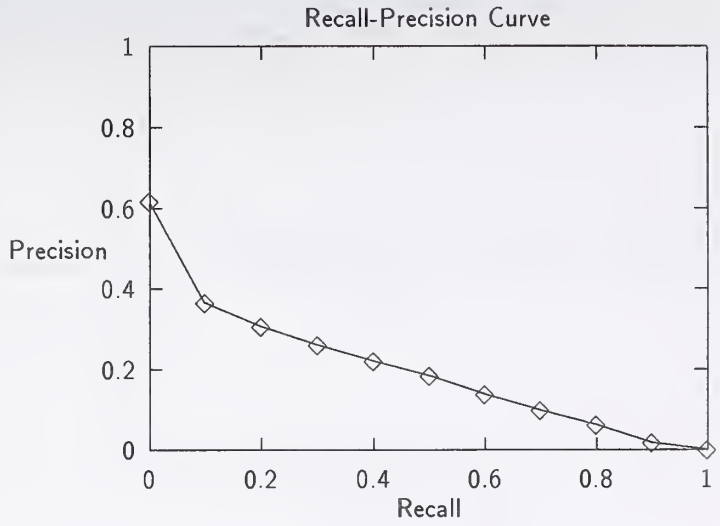
Average precision over all relevant docs	
non-interpolated	0.1817

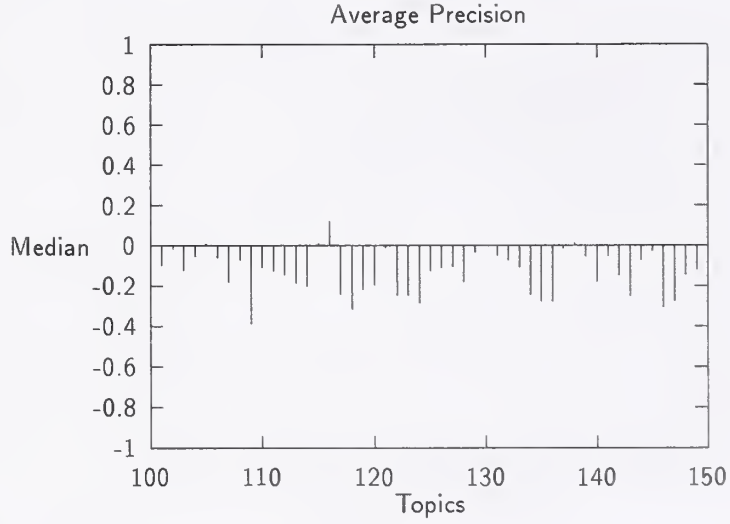
Document Level Averages	
	Precision
At 5 docs	0.4000
At 10 docs	0.3780
At 15 docs	0.3640
At 20 docs	0.3640
At 30 docs	0.3473
At 100 docs	0.2686
At 200 docs	0.2123
At 500 docs	0.1410
At 1000 docs	0.0929

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2371

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00010
0.20	0.00025
0.30	0.00047
0.40	0.00079
0.50	0.00123
0.60	0.00209
0.70	0.00354
0.80	0.00676
0.90	0.02896
1.00	1.85452







Summary Statistics	
Run Number	rutir1–category A, feedback
Number of Topics	50
Total number of documents over all topics	
Retrieved:	36509
Relevant:	9353
Rel_ret:	4817

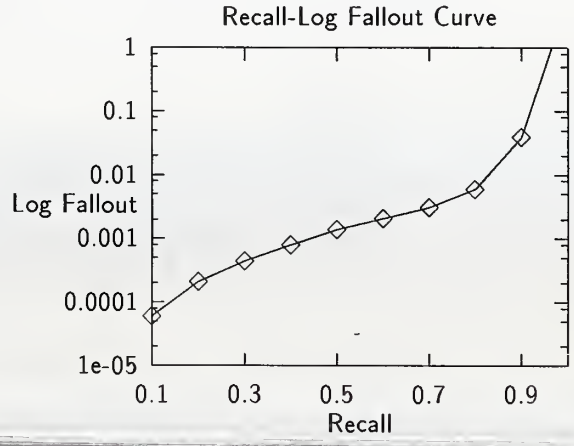
Recall Level Precision Averages	
Recall	Precision
0.00	0.7267
0.10	0.4649
0.20	0.3481
0.30	0.2696
0.40	0.2180
0.50	0.1648
0.60	0.1391
0.70	0.1121
0.80	0.0704
0.90	0.0127
1.00	0.0001

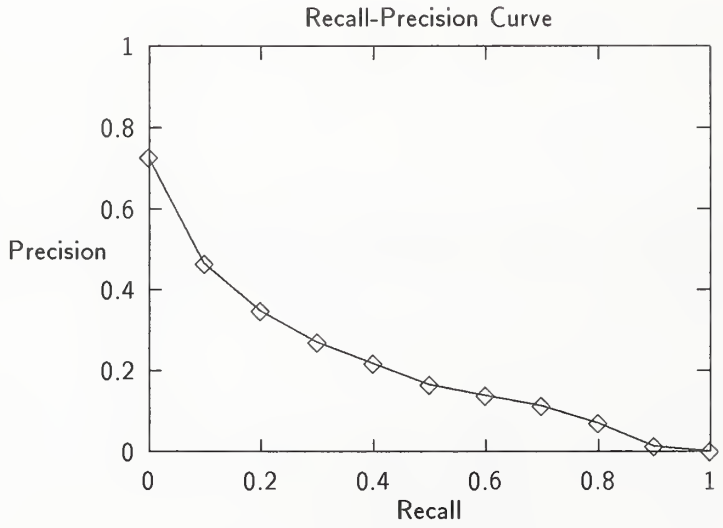
Average precision over all relevant docs	
non-interpolated	0.2045

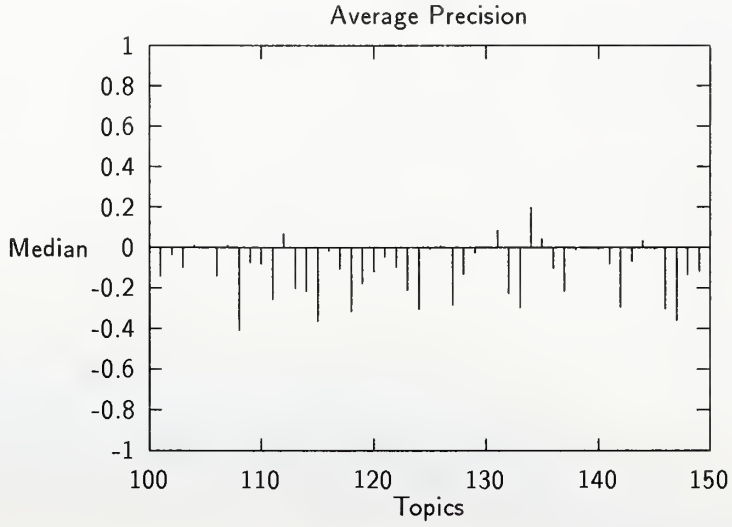
Document Level Averages	
	Precision
At 5 docs	0.4960
At 10 docs	0.4720
At 15 docs	0.4400
At 20 docs	0.4340
At 30 docs	0.3933
At 100 docs	0.2898
At 200 docs	0.2300
At 500 docs	0.1464
At 1000 docs	0.0963

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2549

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00006
0.20	0.00021
0.30	0.00045
0.40	0.00080
0.50	0.00141
0.60	0.00207
0.70	0.00309
0.80	0.00588
0.90	0.03894
1.00	5.56467







Summary Statistics	
Run Number	rutir2–category A, feedback
Number of Topics	50

Total number of documents over all topics	
Retrieved:	43046
Relevant:	9353
Rel_ret:	5800

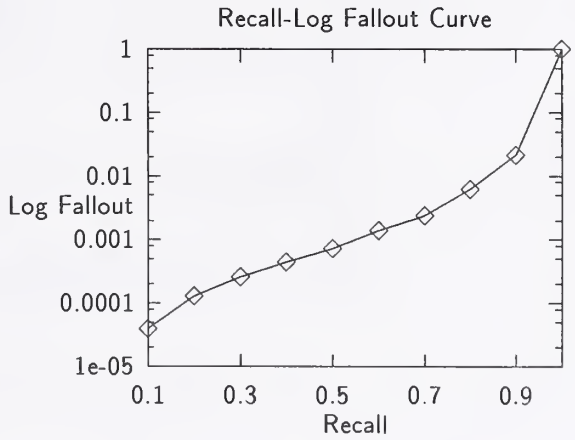
Recall Level Precision Averages	
Recall	Precision
0.00	0.8047
0.10	0.5861
0.20	0.4619
0.30	0.3931
0.40	0.3308
0.50	0.2763
0.60	0.1910
0.70	0.1395
0.80	0.0668
0.90	0.0227
1.00	0.0000

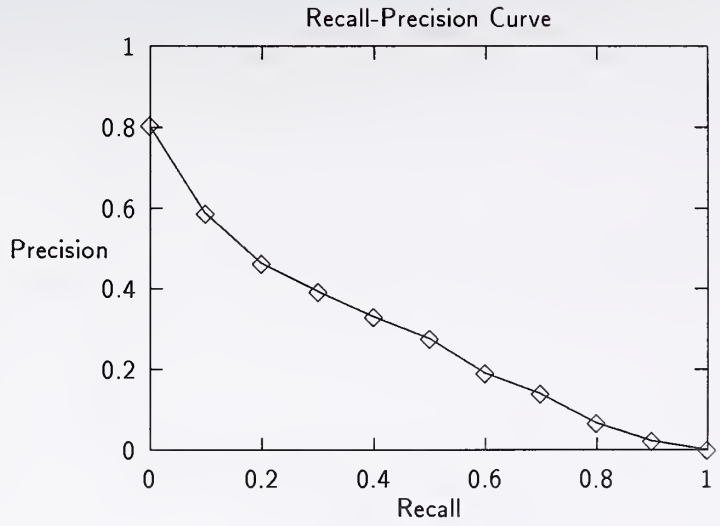
Average precision over all relevant docs	
non-interpolated	0.2755

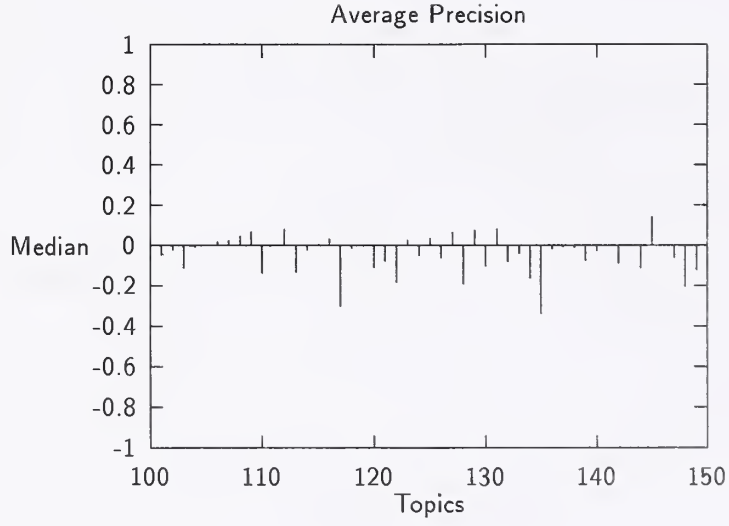
Document Level Averages	
	Precision
At 5 docs	0.5920
At 10 docs	0.5920
At 15 docs	0.5560
At 20 docs	0.5300
At 30 docs	0.4987
At 100 docs	0.3706
At 200 docs	0.3025
At 500 docs	0.1868
At 1000 docs	0.1160

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3210

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00004
0.20	0.00013
0.30	0.00026
0.40	0.00045
0.50	0.00073
0.60	0.00141
0.70	0.00240
0.80	0.00622
0.90	0.02156
1.00	1.00000







Summary Statistics	
Run Number	TOPIC1–category A, manual
Number of Topics	50

Total number of documents over all topics	
Retrieved:	42937
Relevant:	9353
Rel_ret:	5371

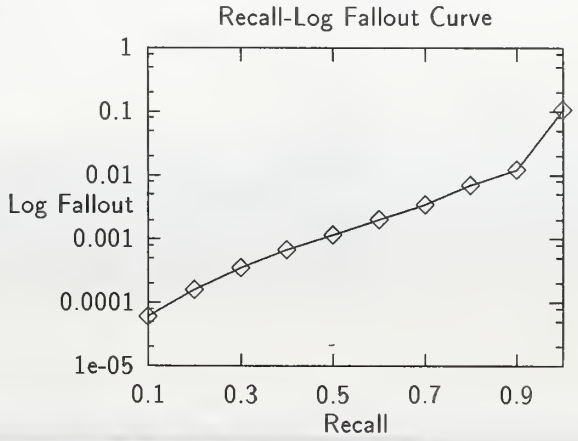
Recall Level Precision Averages	
Recall	Precision
0.00	0.6743
0.10	0.5017
0.20	0.4168
0.30	0.3230
0.40	0.2457
0.50	0.1954
0.60	0.1403
0.70	0.0999
0.80	0.0600
0.90	0.0391
1.00	0.0052

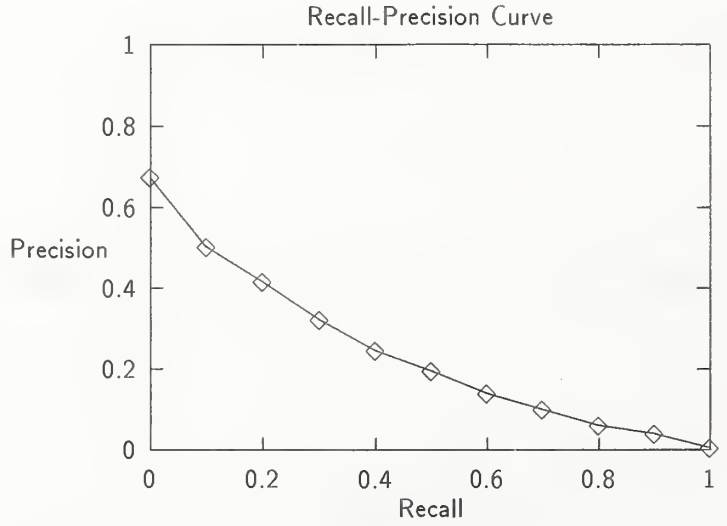
Average precision over all relevant docs	
non-interpolated	0.2243

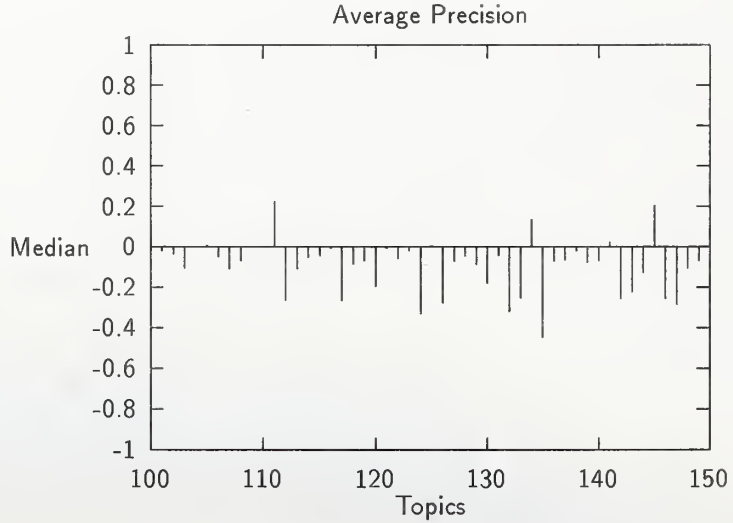
Document Level Averages	
	Precision
At 5 docs	0.5040
At 10 docs	0.4740
At 15 docs	0.4653
At 20 docs	0.4580
At 30 docs	0.4513
At 100 docs	0.3382
At 200 docs	0.2665
At 500 docs	0.1688
At 1000 docs	0.1074

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2786

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00006
0.20	0.00016
0.30	0.00035
0.40	0.00068
0.50	0.00115
0.60	0.00205
0.70	0.00351
0.80	0.00698
0.90	0.01231
1.00	0.10647





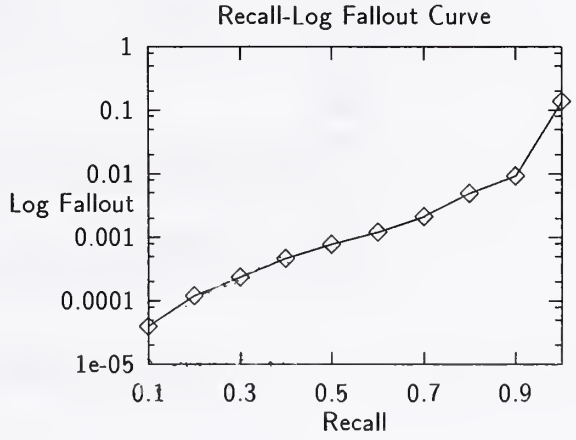


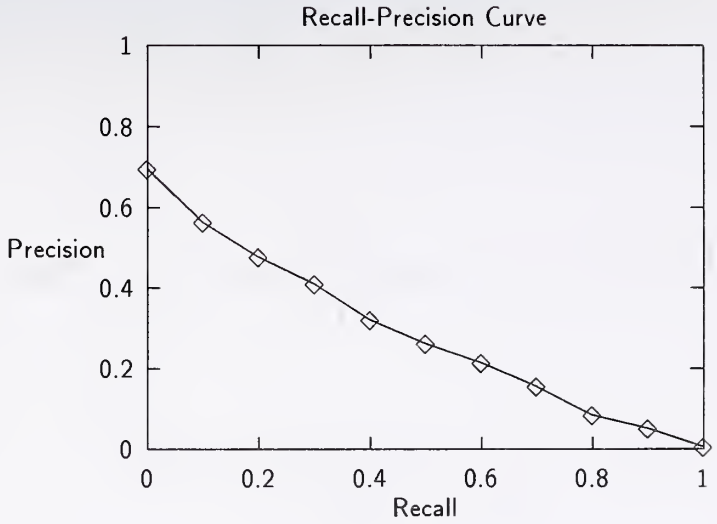
Summary Statistics	
Run Number	TOPIC2–category A, feedback
Number of Topics	50
Total number of documents over all topics	
Retrieved:	43360
Relevant:	9353
Rel_ret:	5952

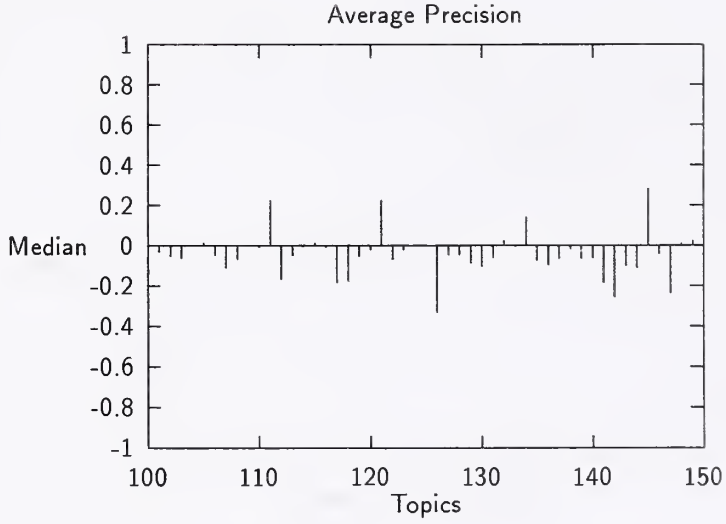
Recall Level Precision Averages	
Recall	Precision
0.00	0.6952
0.10	0.5629
0.20	0.4777
0.30	0.4096
0.40	0.3209
0.50	0.2623
0.60	0.2150
0.70	0.1554
0.80	0.0838
0.90	0.0509
1.00	0.0040
Average precision over all relevant docs	
non-interpolated	0.2774

Document Level Averages	
	Precision
At 5 docs	0.5120
At 10 docs	0.5280
At 15 docs	0.4947
At 20 docs	0.4940
At 30 docs	0.4847
At 100 docs	0.3880
At 200 docs	0.3164
At 500 docs	0.1931
At 1000 docs	0.1190
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3343

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00004
0.20	0.00012
0.30	0.00024
0.40	0.00047
0.50	0.00078
0.60	0.00122
0.70	0.00212
0.80	0.00487
0.90	0.00934
1.00	0.13857







Summary Statistics	
Run Number	UCF101–category A, manual
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	6796

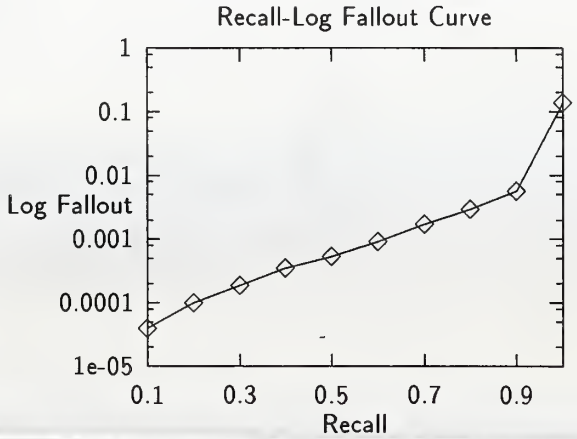
Recall Level Precision Averages	
Recall	Precision
0.00	0.7610
0.10	0.6074
0.20	0.5230
0.30	0.4661
0.40	0.3911
0.50	0.3408
0.60	0.2675
0.70	0.1843
0.80	0.1313
0.90	0.0818
1.00	0.0040

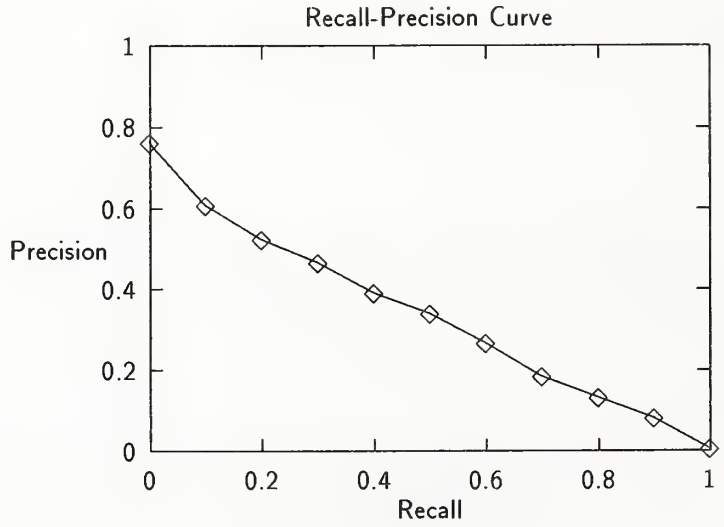
Average precision over all relevant docs	
non-interpolated	0.3278

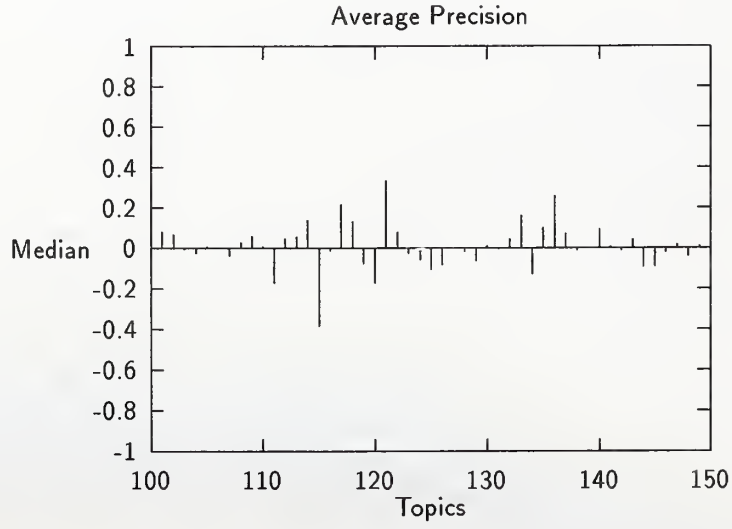
Document Level Averages	
	Precision
At 5 docs	0.5920
At 10 docs	0.5940
At 15 docs	0.5667
At 20 docs	0.5430
At 30 docs	0.5193
At 100 docs	0.4250
At 200 docs	0.3576
At 500 docs	0.2200
At 1000 docs	0.1359

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3707

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00004
0.20	0.00010
0.30	0.00019
0.40	0.00035
0.50	0.00054
0.60	0.00091
0.70	0.00172
0.80	0.00295
0.90	0.00562
1.00	0.13857







Summary Statistics	
Run Number	virtu2–category A, manual
Number of Topics	50

Total number of documents over all topics	
Retrieved:	49785
Relevant:	9353
Rel_ret:	6320

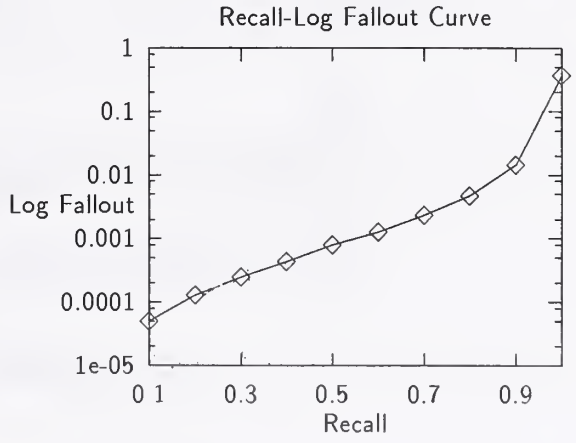
Recall Level Precision Averages	
Recall	Precision
0.00	0.6946
0.10	0.5504
0.20	0.4586
0.30	0.3991
0.40	0.3375
0.50	0.2580
0.60	0.2100
0.70	0.1418
0.80	0.0870
0.90	0.0335
1.00	0.0015

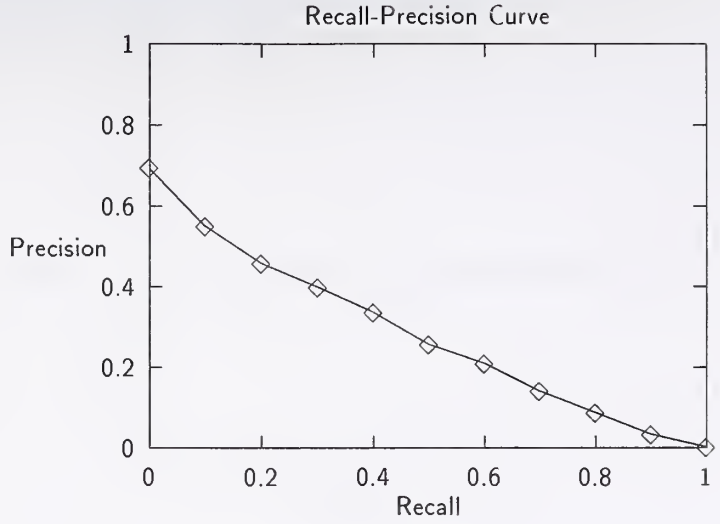
Average precision over all relevant docs	
non-interpolated	0.2717

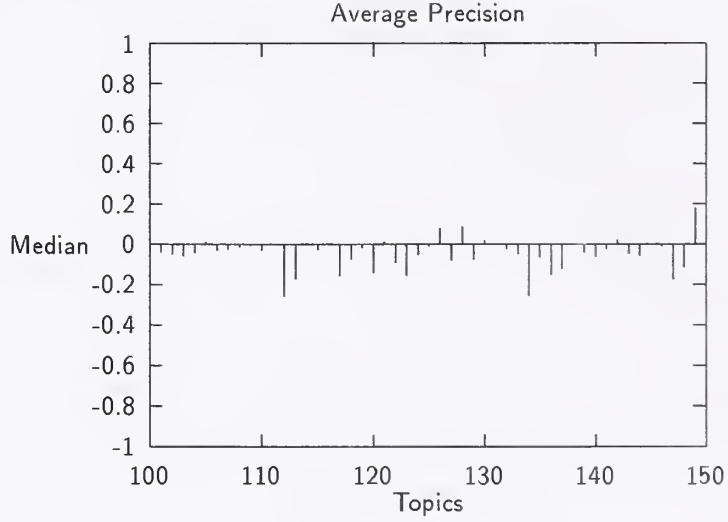
Document Level Averages	
	Precision
At 5 docs	0.5400
At 10 docs	0.5240
At 15 docs	0.4960
At 20 docs	0.4870
At 30 docs	0.4700
At 100 docs	0.3840
At 200 docs	0.3188
At 500 docs	0.2027
At 1000 docs	0.1264

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3204

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00005
0.20	0.00013
0.30	0.00025
0.40	0.00044
0.50	0.00080
0.60	0.00126
0.70	0.00236
0.80	0.00467
0.90	0.01445
1.00	0.37046







Summary Statistics	
Run Number	westp2–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	7186

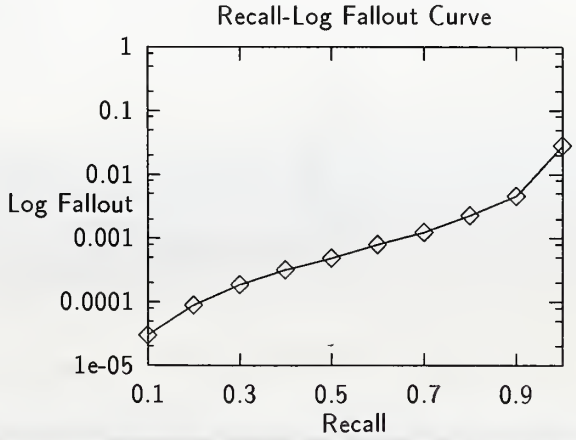
Recall Level Precision Averages	
Recall	Precision
0.00	0.7869
0.10	0.6309
0.20	0.5401
0.30	0.4721
0.40	0.4096
0.50	0.3638
0.60	0.2945
0.70	0.2377
0.80	0.1646
0.90	0.0985
1.00	0.0195

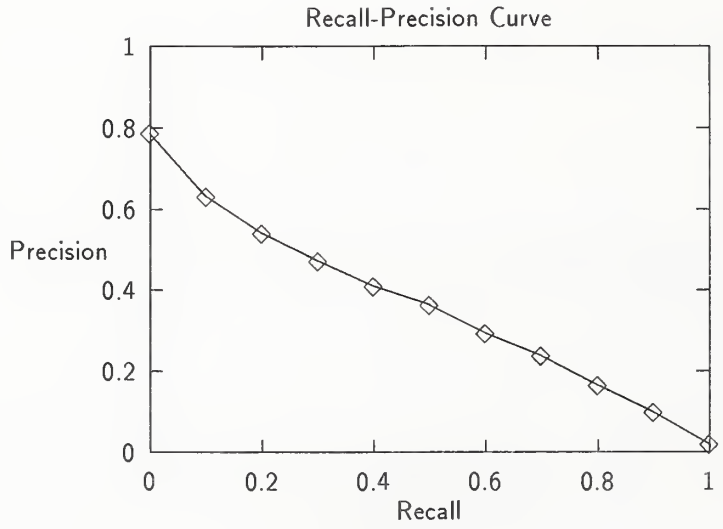
Average precision over all relevant docs	
non-interpolated	0.3535

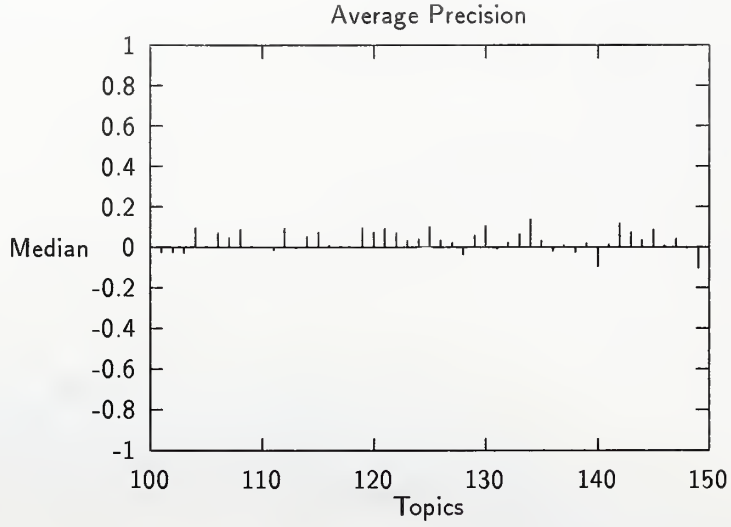
Document Level Averages	
	Precision
At 5 docs	0.6440
At 10 docs	0.6060
At 15 docs	0.5707
At 20 docs	0.5570
At 30 docs	0.5213
At 100 docs	0.4246
At 200 docs	0.3480
At 500 docs	0.2265
At 1000 docs	0.1437

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3827

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00009
0.30	0.00019
0.40	0.00032
0.50	0.00049
0.60	0.00080
0.70	0.00125
0.80	0.00226
0.90	0.00458
1.00	0.02798







Summary Statistics	
Run Number	xerox1–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	7165

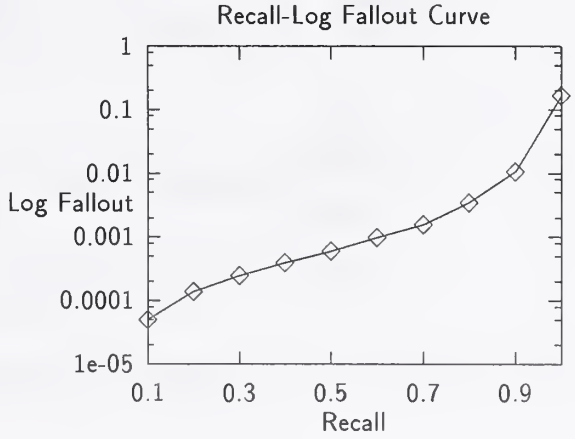
Recall Level Precision Averages	
Recall	Precision
0.00	0.7008
0.10	0.5119
0.20	0.4494
0.30	0.4034
0.40	0.3576
0.50	0.3182
0.60	0.2524
0.70	0.1980
0.80	0.1131
0.90	0.0446
1.00	0.0033

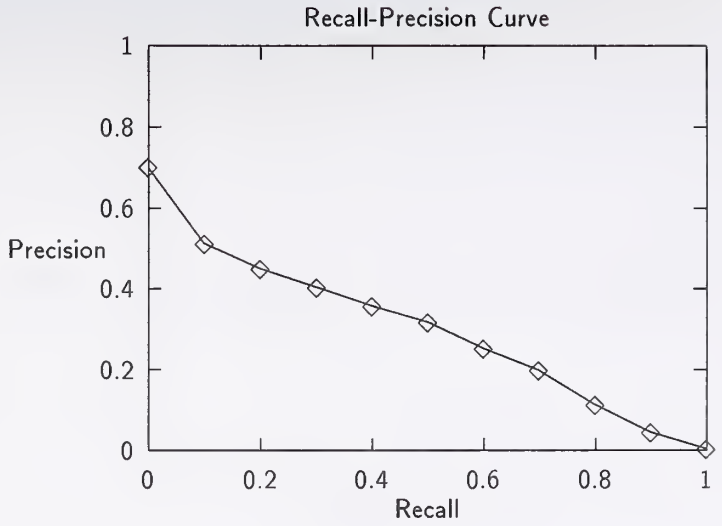
Average precision over all relevant docs	
non-interpolated	0.2867

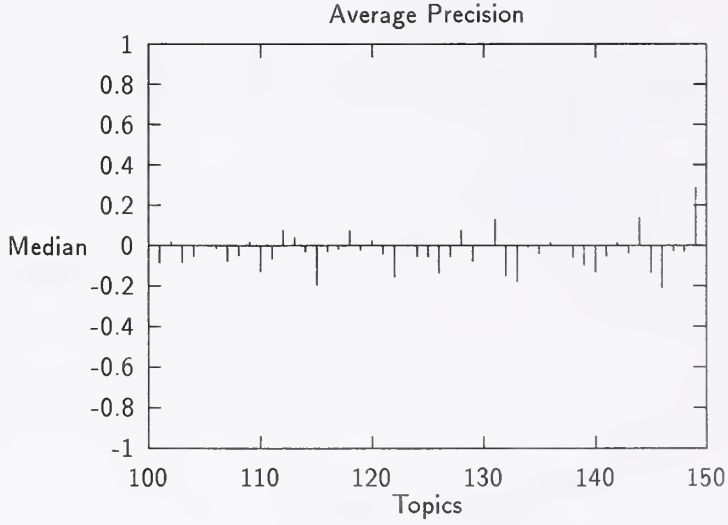
Document Level Averages	
	Precision
At 5 docs	0.5160
At 10 docs	0.4700
At 15 docs	0.4627
At 20 docs	0.4360
At 30 docs	0.4300
At 100 docs	0.3640
At 200 docs	0.3155
At 500 docs	0.2199
At 1000 docs	0.1433

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3421

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00005
0.20	0.00014
0.30	0.00025
0.40	0.00040
0.50	0.00060
0.60	0.00099
0.70	0.00158
0.80	0.00349
0.90	0.01073
1.00	0.16809







Summary Statistics	
Run Number	xerox2–category A, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	9353
Rel_ret:	7242

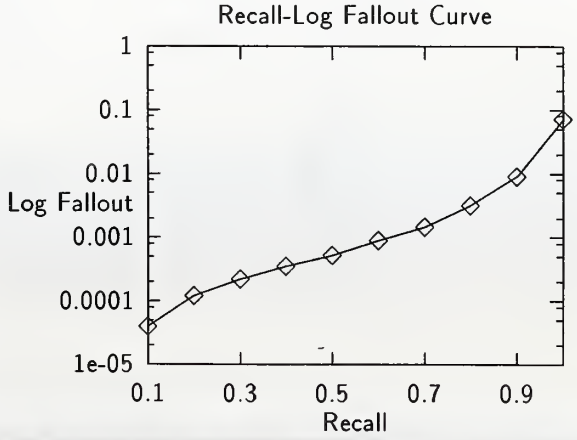
Recall Level Precision Averages	
Recall	Precision
0.00	0.7445
0.10	0.5559
0.20	0.4889
0.30	0.4317
0.40	0.3875
0.50	0.3456
0.60	0.2717
0.70	0.2088
0.80	0.1234
0.90	0.0522
1.00	0.0077

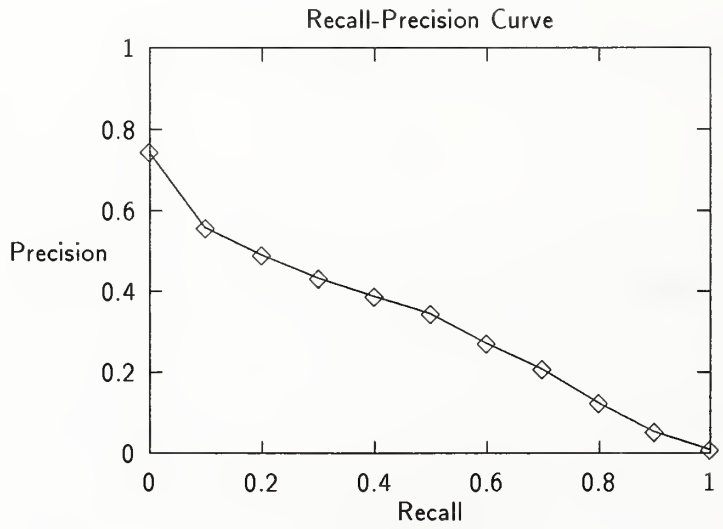
Average precision over all relevant docs	
non-interpolated	0.3111

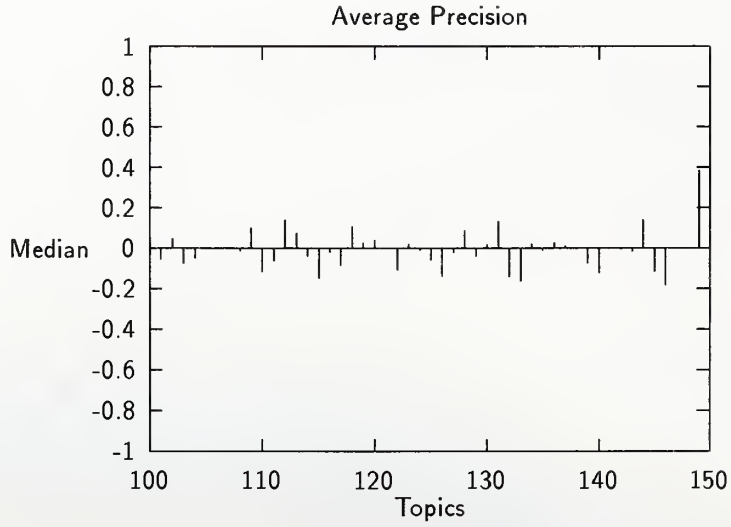
Document Level Averages	
	Precision
At 5 docs	0.5400
At 10 docs	0.5040
At 15 docs	0.4960
At 20 docs	0.4620
At 30 docs	0.4573
At 100 docs	0.3892
At 200 docs	0.3395
At 500 docs	0.2263
At 1000 docs	0.1448

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3660

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00004
0.20	0.00012
0.30	0.00022
0.40	0.00035
0.50	0.00053
0.60	0.00090
0.70	0.00148
0.80	0.00316
0.90	0.00909
1.00	0.07172







Summary Statistics	
Run Number	expst1–category B, automatic
Number of Topics	50

Total number of documents over all topics	
Retrieved:	50000
Relevant:	2559
Rel_ret:	1168

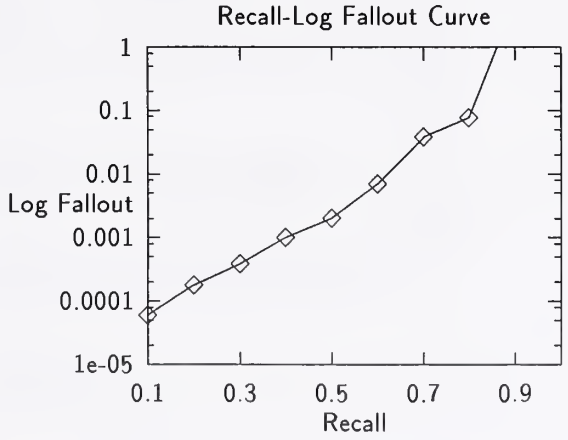
Recall Level Precision Averages	
Recall	Precision
0.00	0.6602
0.10	0.5046
0.20	0.3919
0.30	0.3042
0.40	0.1855
0.50	0.1232
0.60	0.0461
0.70	0.0102
0.80	0.0058
0.90	0.0001
1.00	0.0001

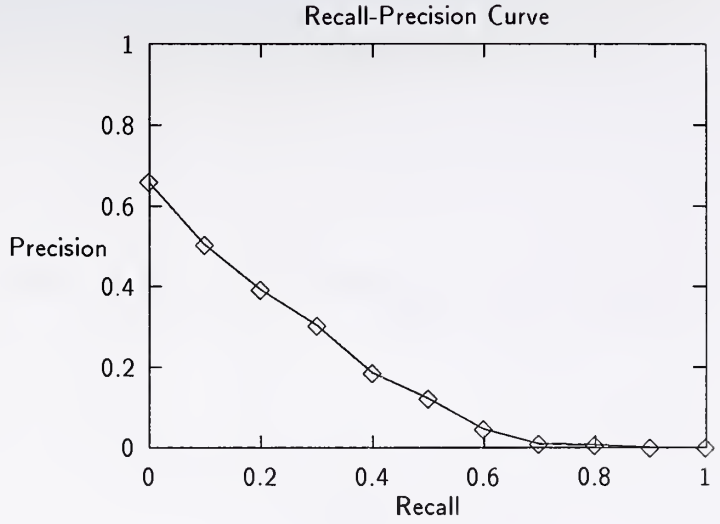
Average precision over all relevant docs	
non-interpolated	0.1838

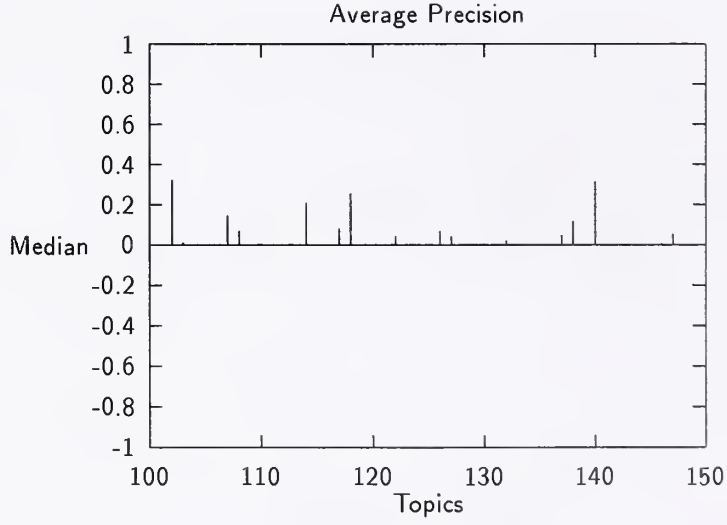
Document Level Averages	
	Precision
At 5 docs	0.4640
At 10 docs	0.3960
At 15 docs	0.3493
At 20 docs	0.3220
At 30 docs	0.2847
At 100 docs	0.1534
At 200 docs	0.0983
At 500 docs	0.0444
At 1000 docs	0.0234

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2355

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00006
0.20	0.00018
0.30	0.00039
0.40	0.00100
0.50	0.00202
0.60	0.00704
0.70	0.03854
0.80	0.07780
0.90	5.10581
1.00	5.67312







Summary Statistics	
Run Number	stpat1–category B, feedback, frozen evaluation
Number of Topics	50

Total number of documents over all topics	
Retrieved:	49298
Relevant:	2559
Rel_ret:	2254

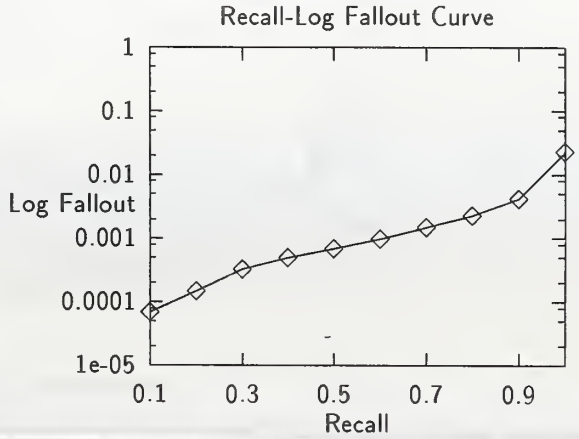
Recall Level Precision Averages	
Recall	Precision
0.00	0.6500
0.10	0.4648
0.20	0.4262
0.30	0.3387
0.40	0.3140
0.50	0.2906
0.60	0.2577
0.70	0.2089
0.80	0.1673
0.90	0.1077
1.00	0.0240

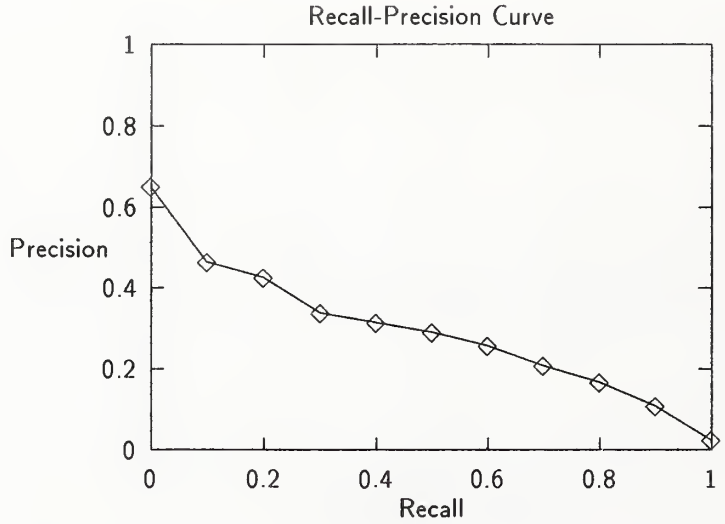
Average precision over all relevant docs	
non-interpolated	0.2737

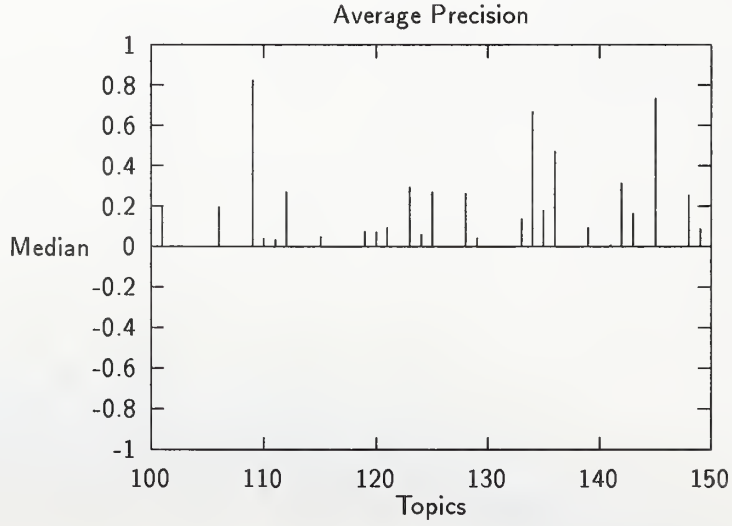
Document Level Averages	
	Precision
At 5 docs	0.3720
At 10 docs	0.3440
At 15 docs	0.3227
At 20 docs	0.3060
At 30 docs	0.2940
At 100 docs	0.2198
At 200 docs	0.1516
At 500 docs	0.0789
At 1000 docs	0.0451

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3006

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00007
0.20	0.00015
0.30	0.00033
0.40	0.00050
0.50	0.00069
0.60	0.00098
0.70	0.00150
0.80	0.00226
0.90	0.00423
1.00	0.02307







Summary Statistics	
Run Number	UCFSJM–category B, manual
Number of Topics	50

Total number of documents over all topics	
Retrieved:	47205
Relevant:	2559
Rel_ret:	2234

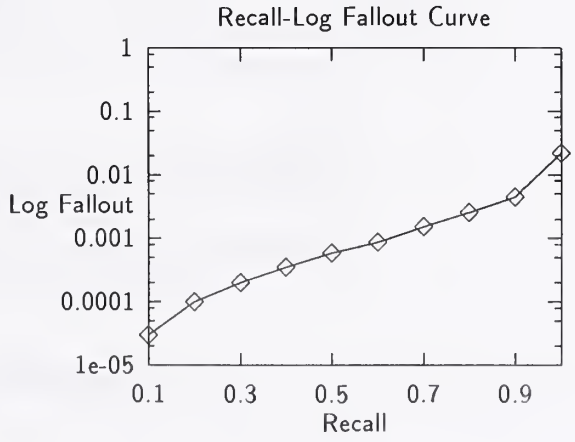
Recall Level Precision Averages	
Recall	Precision
0.00	0.7818
0.10	0.6319
0.20	0.5273
0.30	0.4551
0.40	0.3935
0.50	0.3257
0.60	0.2821
0.70	0.2043
0.80	0.1489
0.90	0.1018
1.00	0.0252

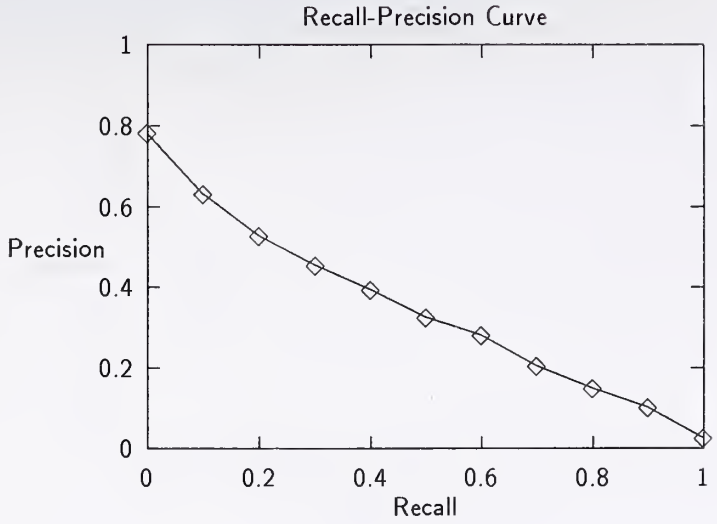
Average precision over all relevant docs	
non-interpolated	0.3326

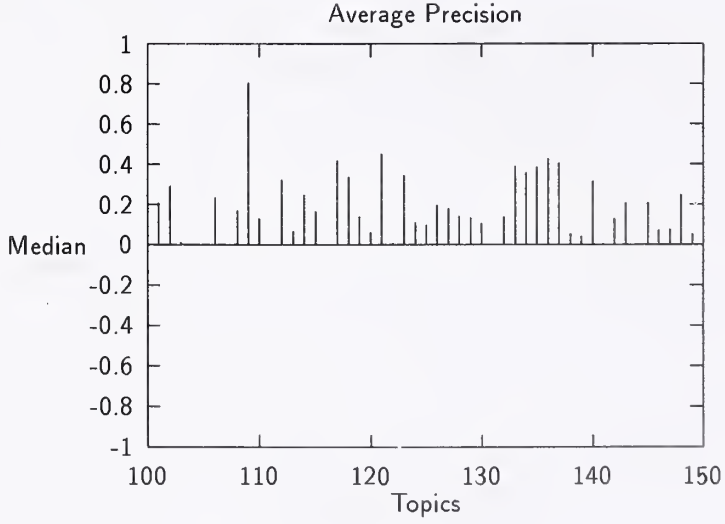
Document Level Averages	
	Precision
At 5 docs	0.5240
At 10 docs	0.4520
At 15 docs	0.4147
At 20 docs	0.3940
At 30 docs	0.3673
At 100 docs	0.2328
At 200 docs	0.1593
At 500 docs	0.0823
At 1000 docs	0.0447

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3605

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00003
0.20	0.00010
0.30	0.00020
0.40	0.00035
0.50	0.00059
0.60	0.00087
0.70	0.00155
0.80	0.00259
0.90	0.00451
1.00	0.02195







Summary Statistics	
Run Number	TeknosN05–AP, manual
Number of Topics	3

Total number of documents over all topics	
Retrieved:	829
Relevant:	535
Rel_ret:	50

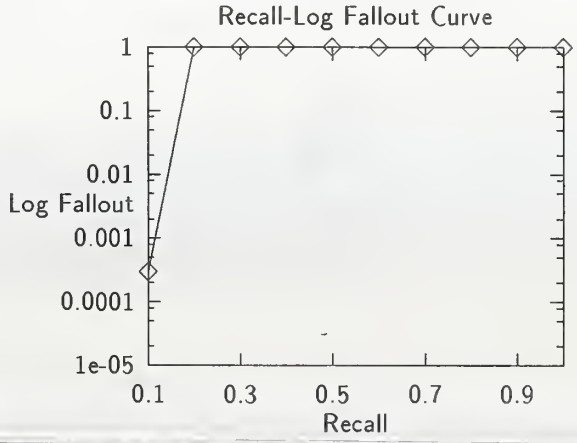
Recall Level Precision Averages	
Recall	Precision
0.00	0.6712
0.10	0.2986
0.20	0.0000
0.30	0.0000
0.40	0.0000
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000

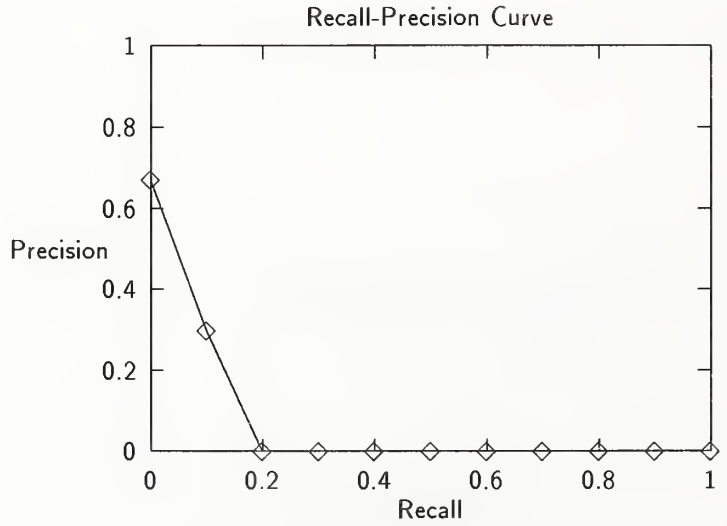
Average precision over all relevant docs	
non-interpolated	0.0489

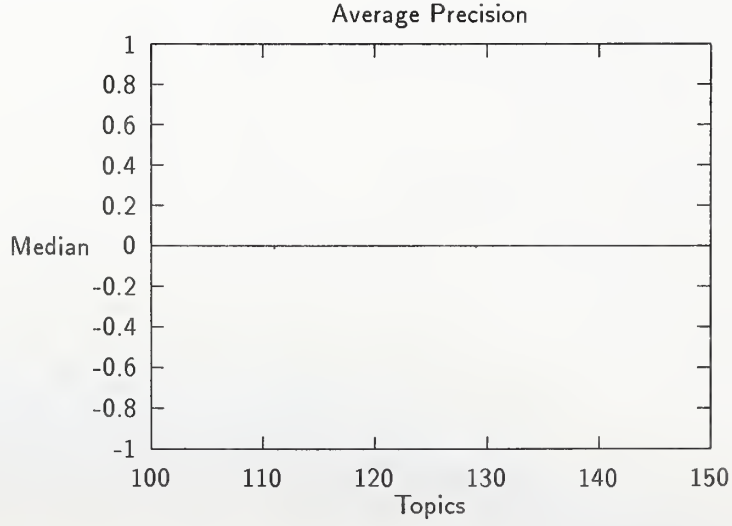
Document Level Averages	
	Precision
At 5 docs	0.3333
At 10 docs	0.3333
At 15 docs	0.3111
At 20 docs	0.2833
At 30 docs	0.2444
At 100 docs	0.1400
At 200 docs	0.0700
At 500 docs	0.0307
At 1000 docs	0.0167

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0711

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00030
0.20	1.00000
0.30	1.00000
0.40	1.00000
0.50	1.00000
0.60	1.00000
0.70	1.00000
0.80	1.00000
0.90	1.00000
1.00	1.00000







Summary Statistics	
Run Number	TeknosN10–AP, manual
Number of Topics	3

Total number of documents over all topics	
Retrieved:	1132
Relevant:	535
Rel_ret:	69

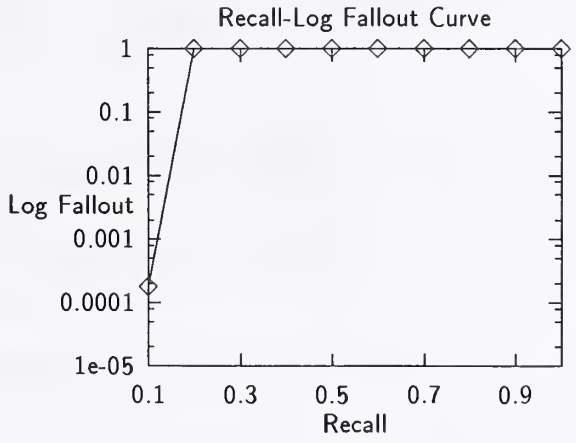
Recall Level Precision Averages	
Recall	Precision
0.00	0.6771
0.10	0.4116
0.20	0.0000
0.30	0.0000
0.40	0.0000
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000

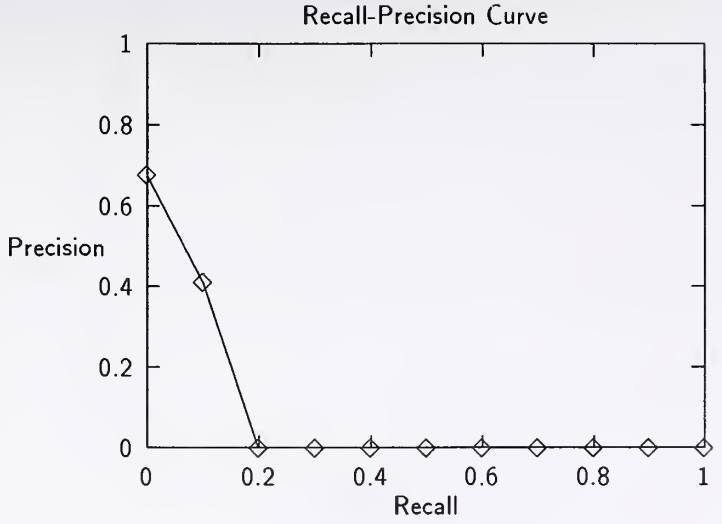
Average precision over all relevant docs	
non-interpolated	0.0613

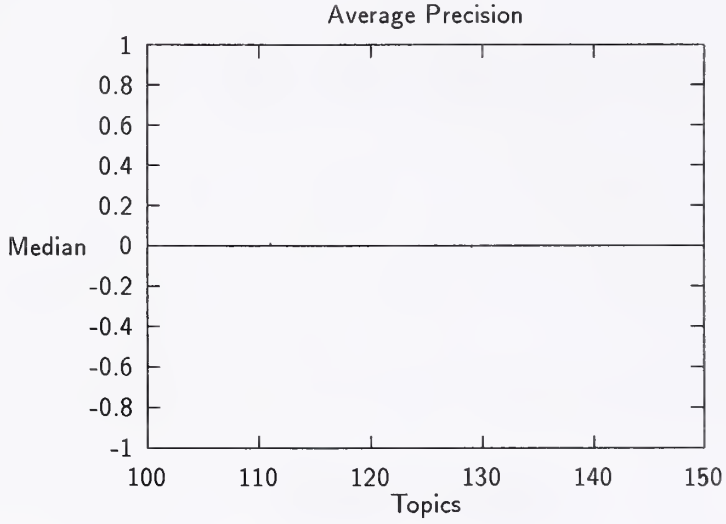
Document Level Averages	
	Precision
At 5 docs	0.3333
At 10 docs	0.3333
At 15 docs	0.3556
At 20 docs	0.3333
At 30 docs	0.2667
At 100 docs	0.1700
At 200 docs	0.0967
At 500 docs	0.0413
At 1000 docs	0.0230

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0962

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00018
0.20	1.00000
0.30	1.00000
0.40	1.00000
0.50	1.00000
0.60	1.00000
0.70	1.00000
0.80	1.00000
0.90	1.00000
1.00	1.00000







Summary Statistics	
Run Number	TeknosN15–AP, manual
Number of Topics	3

Total number of documents over all topics	
Retrieved:	1293
Relevant:	535
Rel_ret:	78

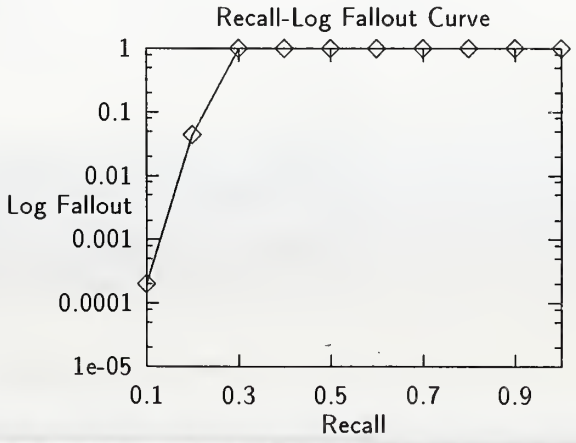
Recall Level Precision Averages	
Recall	Precision
0.00	0.7500
0.10	0.3874
0.20	0.0059
0.30	0.0000
0.40	0.0000
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000

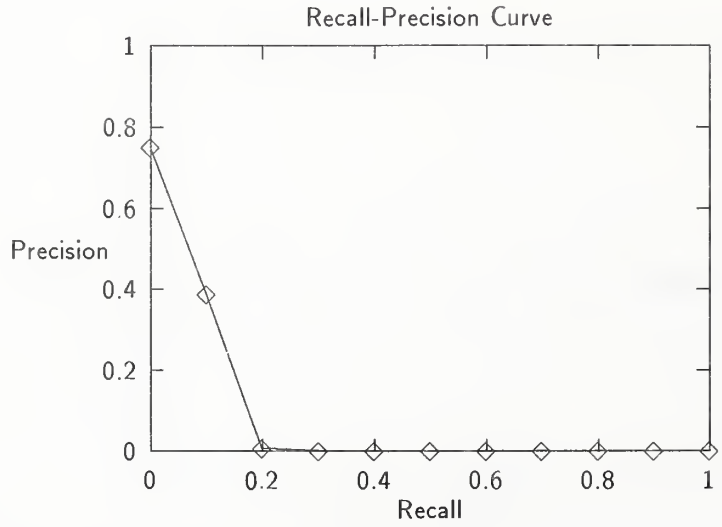
Average precision over all relevant docs	
non-interpolated	0.0602

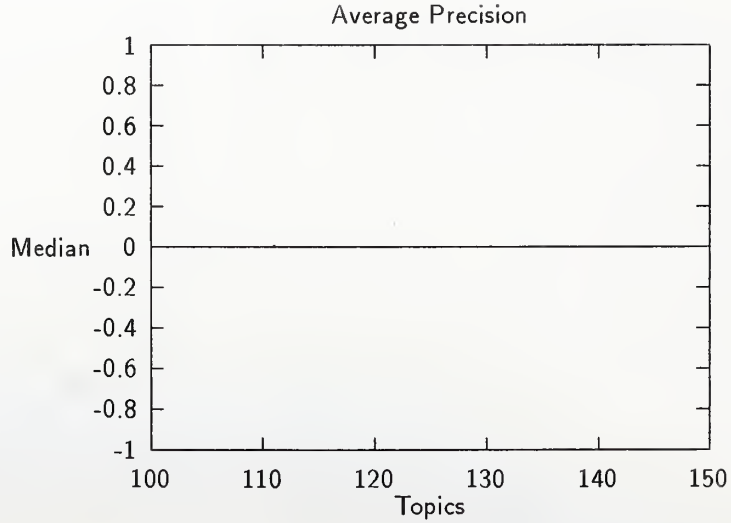
Document Level Averages	
	Precision
At 5 docs	0.4000
At 10 docs	0.3333
At 15 docs	0.3556
At 20 docs	0.3333
At 30 docs	0.2889
At 100 docs	0.1567
At 200 docs	0.1050
At 500 docs	0.0453
At 1000 docs	0.0250

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1076

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00020
0.20	0.04361
0.30	1.00000
0.40	1.00000
0.50	1.00000
0.60	1.00000
0.70	1.00000
0.80	1.00000
0.90	1.00000
1.00	1.00000







Summary Statistics	
Run Number	TeknosN20–AP, manual
Number of Topics	3

Total number of documents over all topics	
Retrieved:	1387
Relevant:	535
Rel_ret:	81

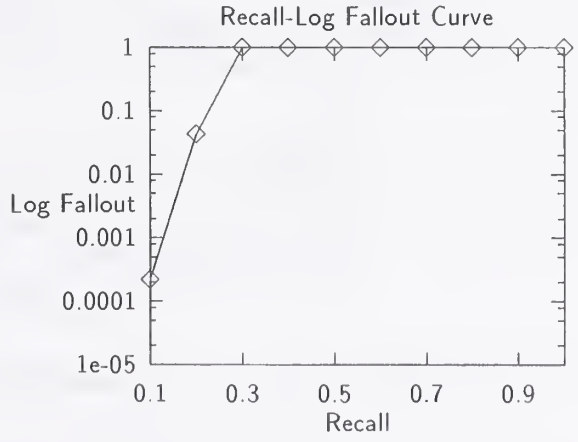
Recall Level Precision Averages	
Recall	Precision
0.00	0.7333
0.10	0.3661
0.20	0.0060
0.30	0.0000
0.40	0.0000
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000

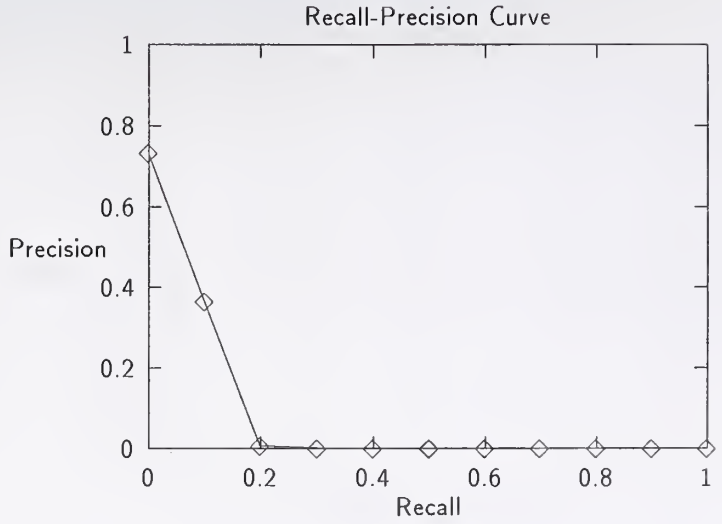
Average precision over all relevant docs	
non-interpolated	0.0578

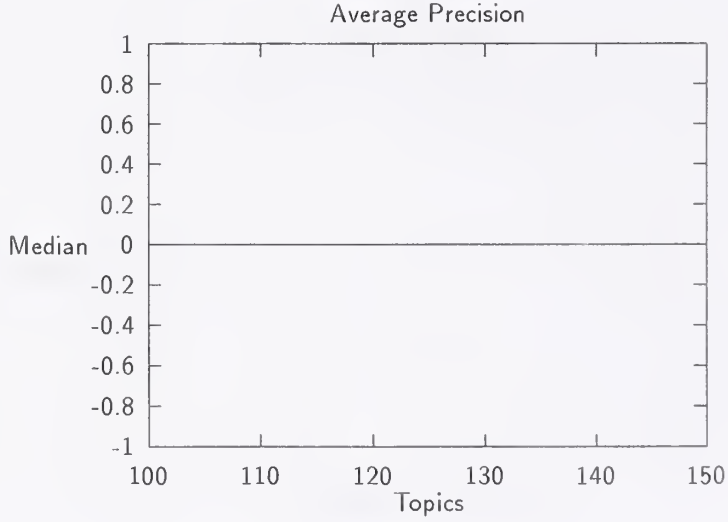
Document Level Averages	
	Precision
At 5 docs	0.4000
At 10 docs	0.3333
At 15 docs	0.3111
At 20 docs	0.3167
At 30 docs	0.2778
At 100 docs	0.1400
At 200 docs	0.1100
At 500 docs	0.0467
At 1000 docs	0.0257

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1138

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00022
0.20	0.04288
0.30	1.00000
0.40	1.00000
0.50	1.00000
0.60	1.00000
0.70	1.00000
0.80	1.00000
0.90	1.00000
1.00	1.00000







Summary Statistics	
Run Number	TeknosN30–AP, manual
Number of Topics	3
Total number of documents over all topics	
Retrieved:	1544
Relevant:	535
Rel_ret:	88

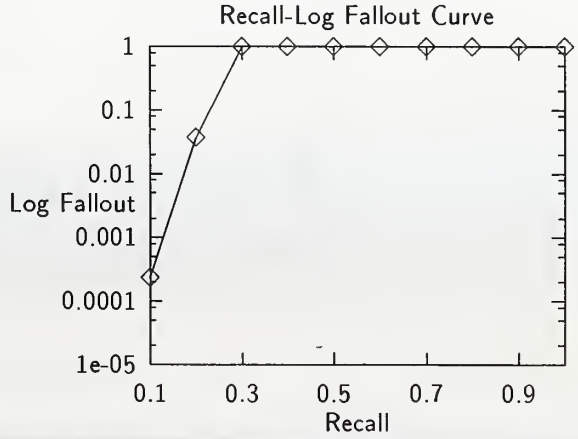
Recall Level Precision Averages	
Recall	Precision
0.00	0.6847
0.10	0.3515
0.20	0.0068
0.30	0.0000
0.40	0.0000
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000

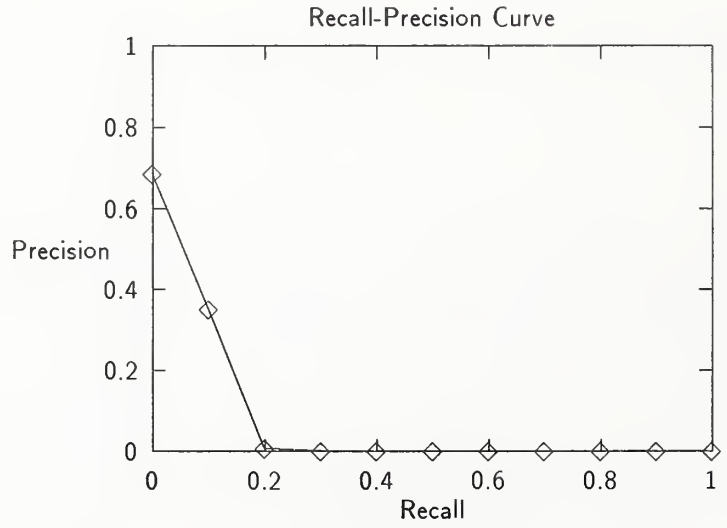
Average precision over all relevant docs	
non-interpolated	0.0551

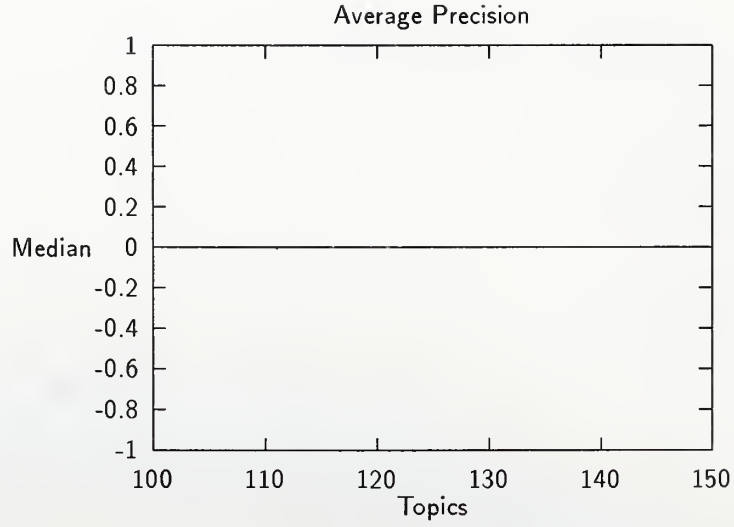
Document Level Averages	
	Precision
At 5 docs	0.3333
At 10 docs	0.3000
At 15 docs	0.3333
At 20 docs	0.3333
At 30 docs	0.2889
At 100 docs	0.1267
At 200 docs	0.1133
At 500 docs	0.0493
At 1000 docs	0.0277

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1146

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00024
0.20	0.03780
0.30	1.00000
0.40	1.00000
0.50	1.00000
0.60	1.00000
0.70	1.00000
0.80	1.00000
0.90	1.00000
1.00	1.00000







Summary Statistics	
Run Number	TeknosRel–AP, manual
Number of Topics	3
Total number of documents over all topics	
Retrieved:	97
Relevant:	535
Rel_ret:	60

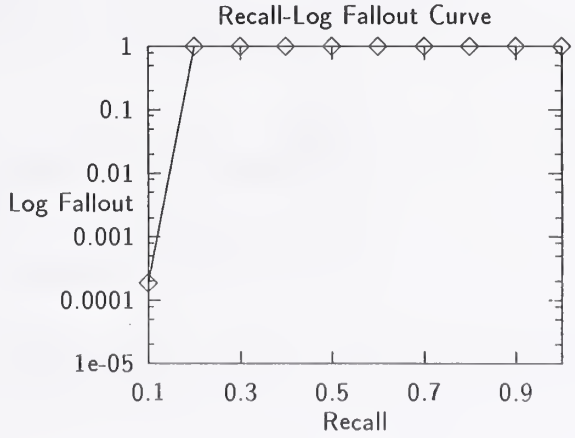
Recall Level Precision Averages	
Recall	Precision
0.00	0.8889
0.10	0.4111
0.20	0.0000
0.30	0.0000
0.40	0.0000
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000

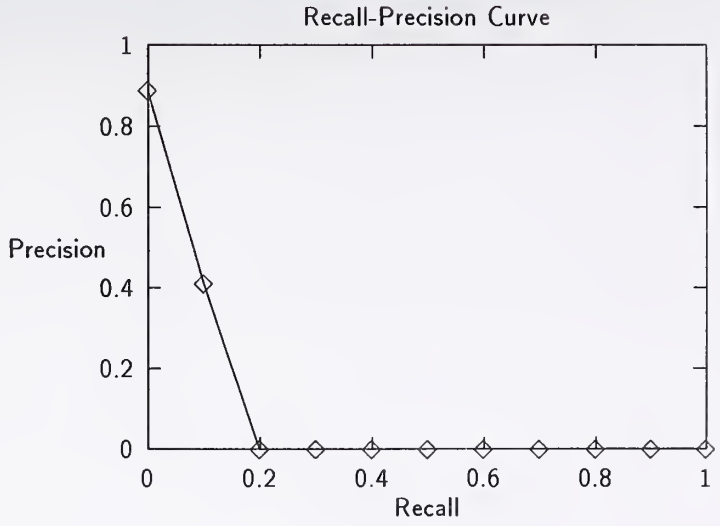
Average precision over all relevant docs	
non-interpolated	0.0857

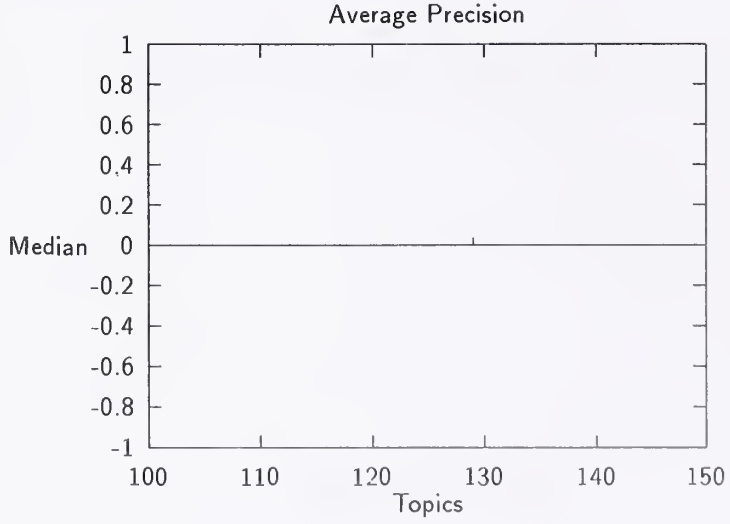
Document Level Averages	
	Precision
At 5 docs	0.6667
At 10 docs	0.6667
At 15 docs	0.6667
At 20 docs	0.6667
At 30 docs	0.5667
At 100 docs	0.2000
At 200 docs	0.1000
At 500 docs	0.0400
At 1000 docs	0.0200

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1200

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00019
0.20	1.00000
0.30	1.00000
0.40	1.00000
0.50	1.00000
0.60	1.00000
0.70	1.00000
0.80	1.00000
0.90	1.00000
1.00	1.00000







Summary Statistics	
Run Number	TeknosW05–AP, manual
Number of Topics	3

Total number of documents over all topics	
Retrieved:	378
Relevant:	535
Rel_ret:	31

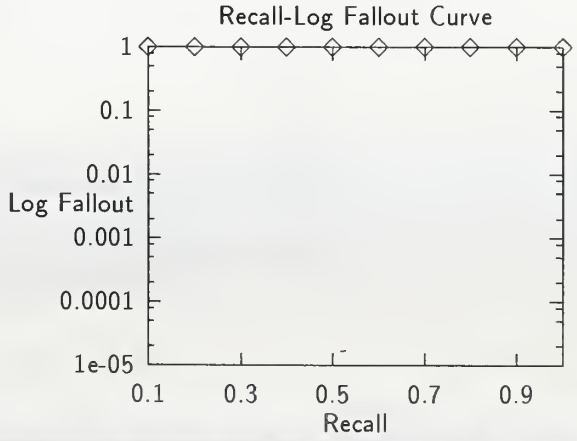
Recall Level Precision Averages	
Recall	Precision
0.00	0.4491
0.10	0.0000
0.20	0.0000
0.30	0.0000
0.40	0.0000
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000

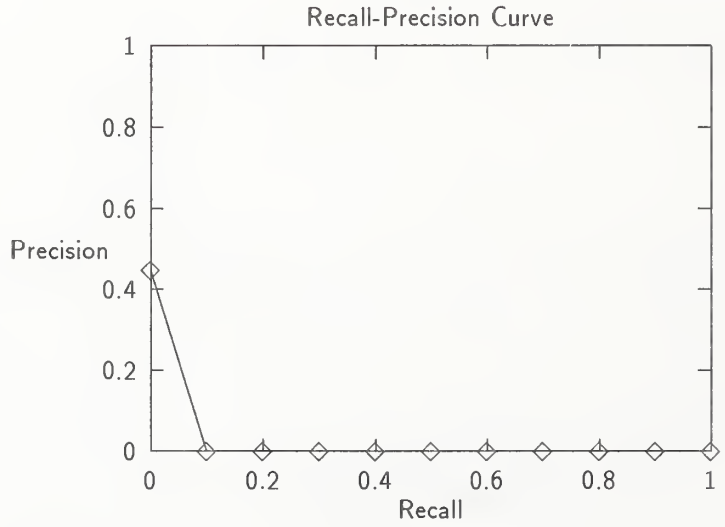
Average precision over all relevant docs	
non-interpolated	0.0341

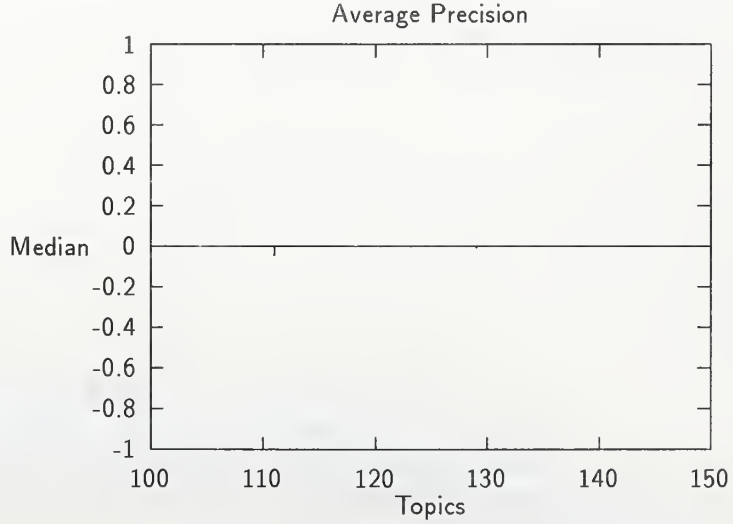
Document Level Averages	
	Precision
At 5 docs	0.3333
At 10 docs	0.3000
At 15 docs	0.2889
At 20 docs	0.2833
At 30 docs	0.2222
At 100 docs	0.0867
At 200 docs	0.0467
At 500 docs	0.0207
At 1000 docs	0.0103

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0488

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	1.00000
0.20	1.00000
0.30	1.00000
0.40	1.00000
0.50	1.00000
0.60	1.00000
0.70	1.00000
0.80	1.00000
0.90	1.00000
1.00	1.00000







Summary Statistics	
Run Number	TeknosW10–AP, manual
Number of Topics	3
Total number of documents over all topics	
Retrieved:	637
Relevant:	535
Rel_ret:	45

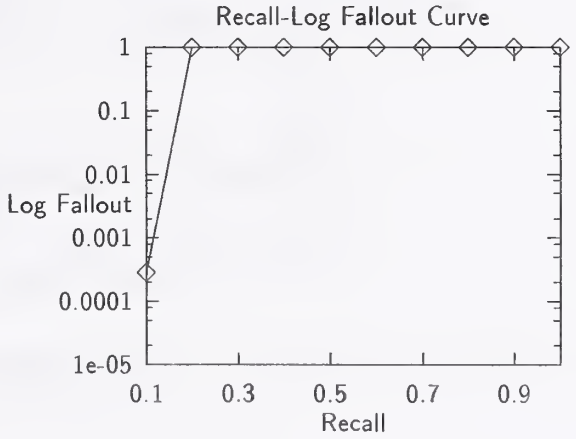
Recall Level Precision Averages	
Recall	Precision
0.00	0.4425
0.10	0.3073
0.20	0.0000
0.30	0.0000
0.40	0.0000
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000

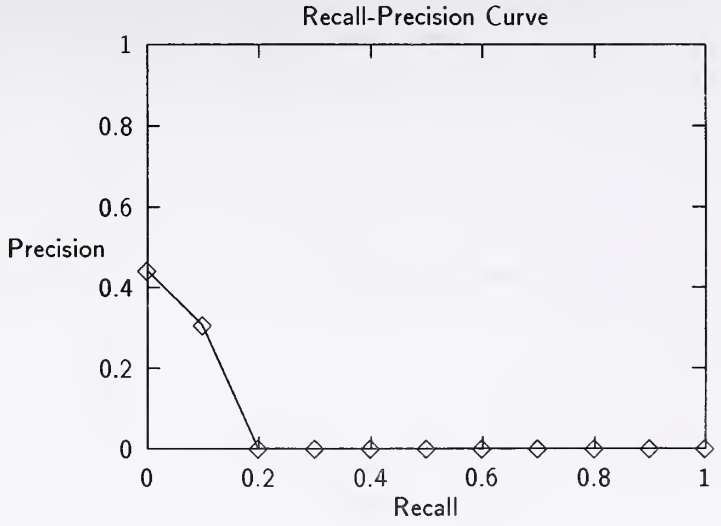
Average precision over all relevant docs	
non-interpolated	0.0487

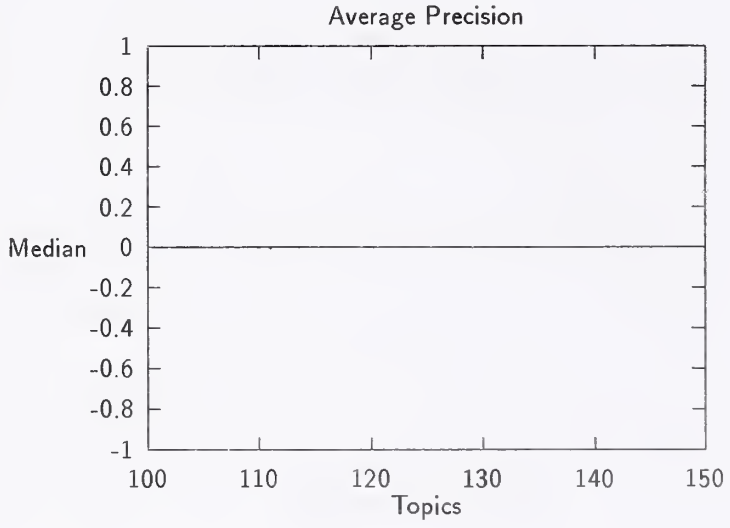
Document Level Averages	
	Precision
At 5 docs	0.4000
At 10 docs	0.3333
At 15 docs	0.3111
At 20 docs	0.3167
At 30 docs	0.2444
At 100 docs	0.1300
At 200 docs	0.0683
At 500 docs	0.0300
At 1000 docs	0.0150

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0725

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00029
0.20	1.00000
0.30	1.00000
0.40	1.00000
0.50	1.00000
0.60	1.00000
0.70	1.00000
0.80	1.00000
0.90	1.00000
1.00	1.00000







Summary Statistics	
Run Number	TeknosW15–AP, manual
Number of Topics	3

Total number of documents over all topics	
Retrieved:	794
Relevant:	535
Rel_ret:	57

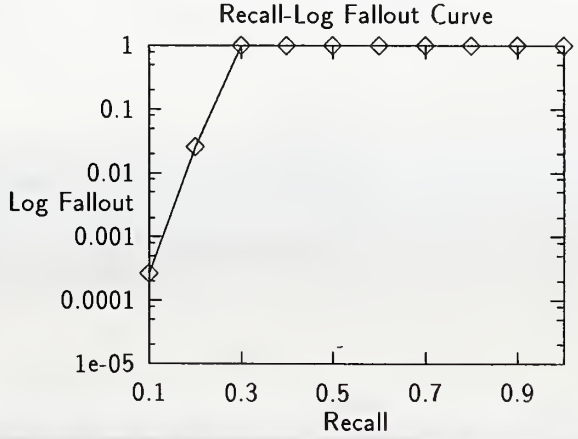
Recall Level Precision Averages	
Recall	Precision
0.00	0.4833
0.10	0.3257
0.20	0.0100
0.30	0.0000
0.40	0.0000
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000

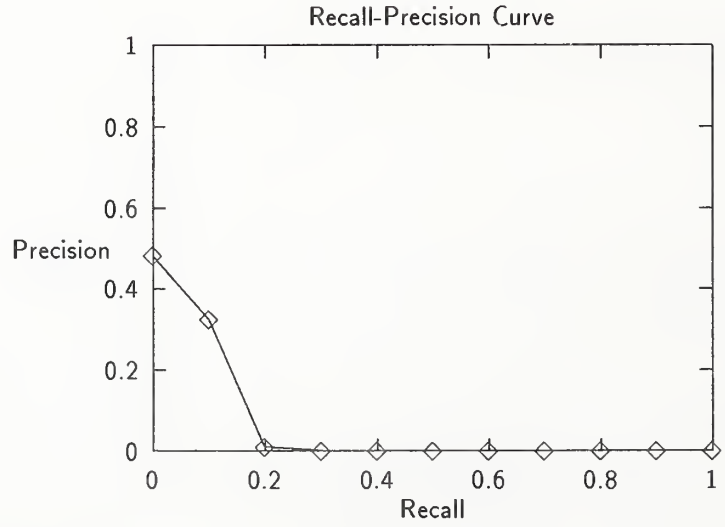
Average precision over all relevant docs	
non-interpolated	0.0544

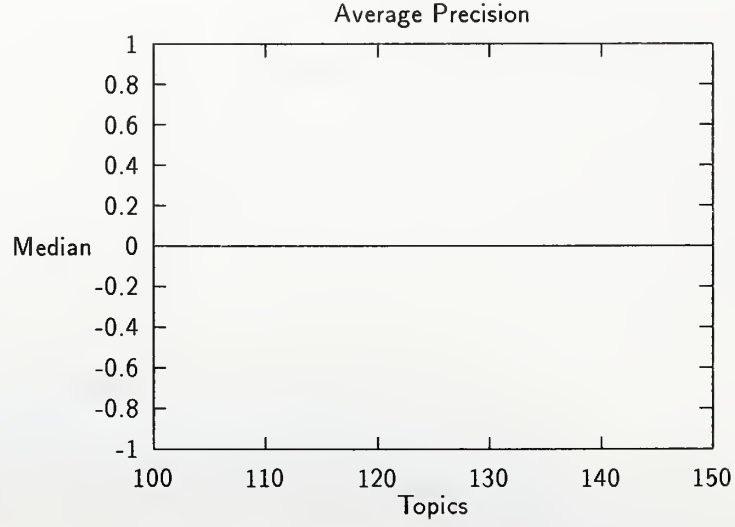
Document Level Averages	
	Precision
At 5 docs	0.4000
At 10 docs	0.4000
At 15 docs	0.3556
At 20 docs	0.3167
At 30 docs	0.2556
At 100 docs	0.1433
At 200 docs	0.0783
At 500 docs	0.0360
At 1000 docs	0.0190

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0835

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00027
0.20	0.02562
0.30	1.00000
0.40	1.00000
0.50	1.00000
0.60	1.00000
0.70	1.00000
0.80	1.00000
0.90	1.00000
1.00	1.00000







Summary Statistics	
Run Number	TeknosW20–AP, manual
Number of Topics	3
Total number of documents over all topics	
Retrieved:	922
Relevant:	535
Rel_ret:	65

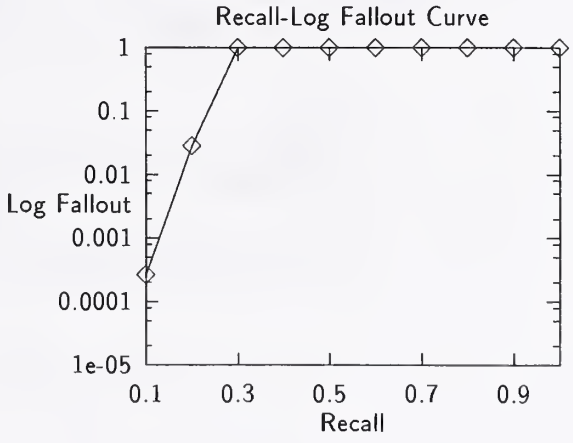
Recall Level Precision Averages	
Recall	Precision
0.00	0.7500
0.10	0.3236
0.20	0.0090
0.30	0.0000
0.40	0.0000
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000

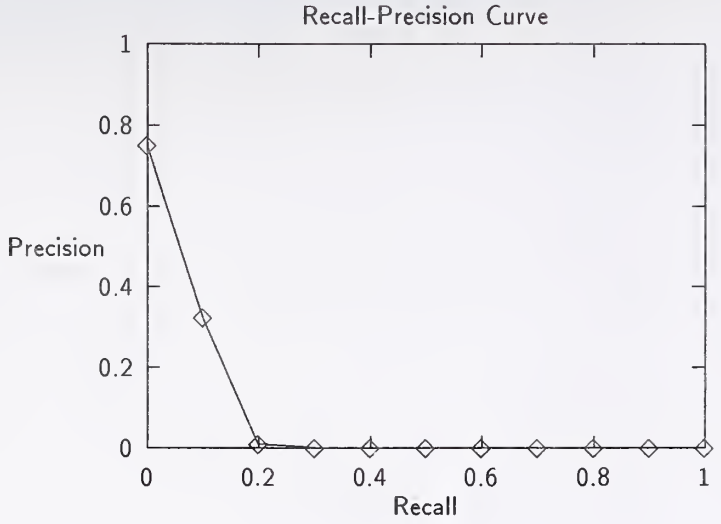
Average precision over all relevant docs	
non-interpolated	0.0586

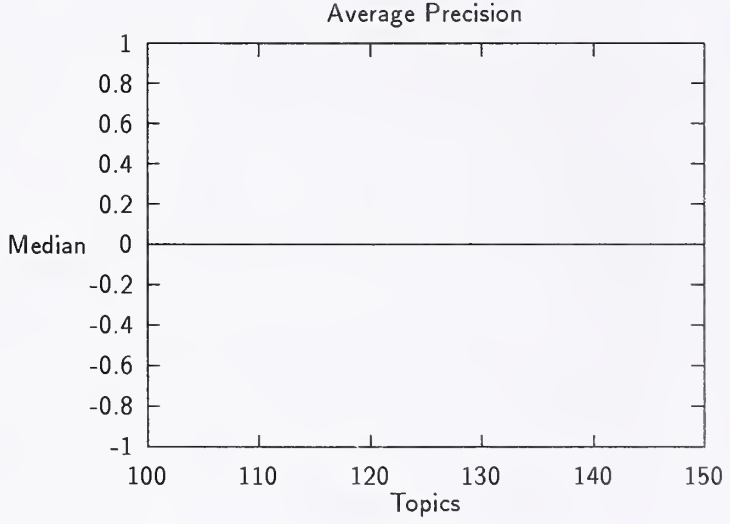
Document Level Averages	
	Precision
At 5 docs	0.4000
At 10 docs	0.4000
At 15 docs	0.3778
At 20 docs	0.3500
At 30 docs	0.2556
At 100 docs	0.1567
At 200 docs	0.0917
At 500 docs	0.0400
At 1000 docs	0.0217

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0964

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00027
0.20	0.02850
0.30	1.00000
0.40	1.00000
0.50	1.00000
0.60	1.00000
0.70	1.00000
0.80	1.00000
0.90	1.00000
1.00	1.00000







Summary Statistics	
Run Number	TeknosW30–AP, manual
Number of Topics	3
Total number of documents over all topics	
Retrieved:	1091
Relevant:	535
Rel_ret:	72

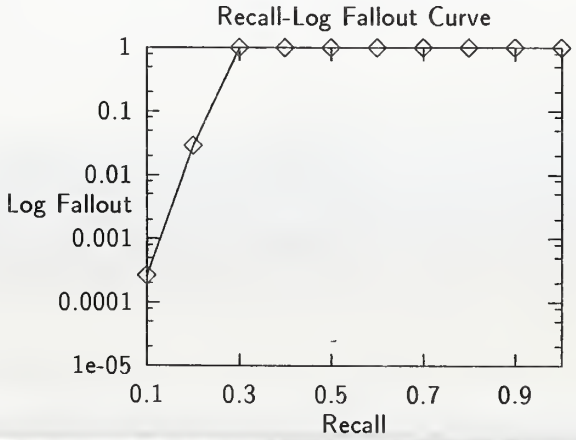
Recall Level Precision Averages	
Recall	Precision
0.00	0.7111
0.10	0.3257
0.20	0.0088
0.30	0.0000
0.40	0.0000
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000

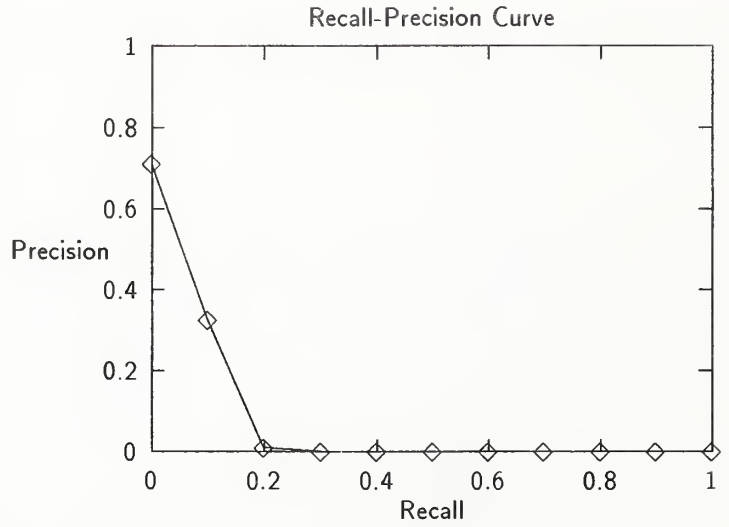
Average precision over all relevant docs	
non-interpolated	0.0593

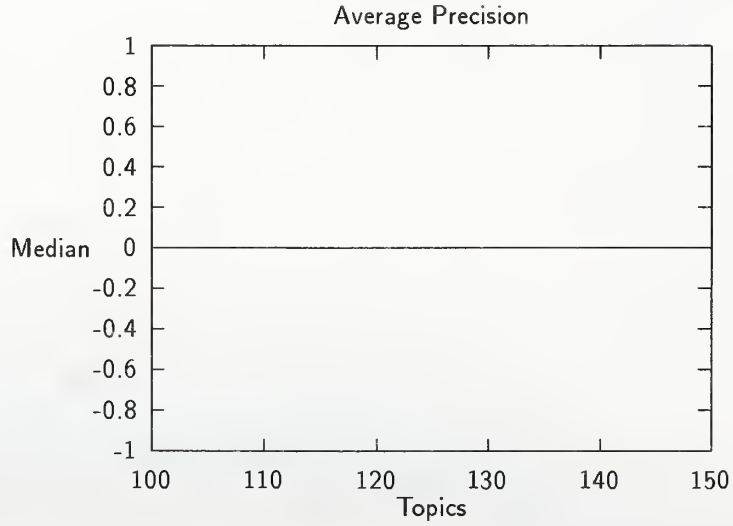
Document Level Averages	
	Precision
At 5 docs	0.4667
At 10 docs	0.3667
At 15 docs	0.4222
At 20 docs	0.3667
At 30 docs	0.2667
At 100 docs	0.1633
At 200 docs	0.0983
At 500 docs	0.0447
At 1000 docs	0.0240

R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0972

Recall Fallout Averages	
Recall	Fallout
0.00	0.00000
0.10	0.00027
0.20	0.02915
0.30	1.00000
0.40	1.00000
0.50	1.00000
0.60	1.00000
0.70	1.00000
0.80	1.00000
0.90	1.00000
1.00	1.00000







APPENDIX B

This appendix contains the system description forms filled out by each participating group. These forms are meant to supplement the system papers and contain a standarded and formatted description of system features and timing aspects.

System Summary and Timing
Organization Name: Xerox PARC
List of Run ID's: Xerox1, Xerox2, Xerox3, Xerox4

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list: 951
- Stemming Algorithm: Two-level Finite-state lexicon
  - Morphological Analysis: yes
- Term Weighting: applied at run-time, typically tf.idf
- Phrase Discovery? : yes
  - Kind of Phrase: word pairs
  - Method Used (statistical, syntactic, other): statistical
- Tokenizer? : Rule-based Finite-state
  - Patterns which are tokenized: dates, numbers, abbreviations, words, model numbers
- Other Techniques for building Data Structures: automatically constructed word thesaurus from analysis of lexical cooccurrence events; uses SVD matrix decomposition to reduce dimensionality

Statistics on Data Structures built from TREC Text

- Inverted index
  - Run ID : Xerox1, Xerox2, Xerox3, Xerox4
  - Total Storage (in MB): 1200
  - Total Computer Time to Build (in hours): 120
  - Automatic Process? (If not, number of manual hours): yes
  - Use of Term Positions? : yes
  - Only Single Terms Used? : no

- Special Routing Structures
  - Run ID : Xerox1, Xerox2
  - Type of Structure: Neural net classifier
  - Total Storage (in MB): less than 1
  - Total Computer Time to Build (in hours): 100
  - Automatic Process? (If not, number of manual hours): yes
  - Brief Description of Method: Used standard back-propagation neural net training method

- Other Data Structures built from TREC text
  - Run ID : xerox1, xerox2, xerox3, xerox4
  - Type of Structure: Document profiles
  - Total Storage (in MB): 700
  - Total Computer Time to Build (in hours): 120
  - Automatic Process? (If not, number of manual hours): yes
  - Brief Description of Method: These are parsed, tokenized, but uninverted document representation that speed subsequent computations

- Other Data Structures built from TREC text
  - Run ID : xerox1, xerox2, xerox3, xerox4
  - Type of Structure: Word thesaurus vectors
  - Total Storage (in MB): 80
  - Total Computer Time to Build (in hours): 24
  - Automatic Process? (If not, number of manual hours): yes
  - Brief Description of Method: One vector per unique word that captures the cooccurrence patterns for that word. Used to define similarity

between words

- Other Data Structures built from TREC text
  - Run ID : xerox1, xerox2, xerox3, xerox4
  - Type of Structure: Document segments: TextTiles
  - Total Storage (in MB): 20
  - Total Computer Time to Build (in hours): 24
  - Automatic Process? (If not, number of manual hours): yes
  - Brief Description of Method: Segment long texts into topic coherent multi-paragraph sections

Query construction

Automatically Built Queries (Ad-Hoc)

- Topic Fields Used: yes
- Average Computer Time to Build Query (in cpu seconds): less than 1
- Method used in Query Construction
  - Term Weighting (weights based on terms in topics)? : yes
  - Phrase Extraction from Topics? : yes
  - Tokenizer? : rule-based finite-state
    - Patterns which are Tokenized: dates, numbers, abbreviations, words, model numbers
  - Expansion of Queries using Previously-Constructed Data Structure? :
    - Structure Used: word thesaurus vectors

Automatically Built Queries (Routing)

- Topic Fields Used: yes
- Average Computer Time to Build Query (in cpu seconds): 120
- Method used in Query Construction
  - Terms Selected From
    - Topics: yes
    - All Training Documents: yes
  - Term Weighting with Weights Based on terms in
    - Topics: yes
    - All Training Documents: yes
  - Phrase Extraction from
    - Topics: yes
    - All Training Documents: yes
  - Tokenizer
    - Patterns which are tokenized (dates, phone numbers, common patterns, etc): dates, numbers, abbreviations, words, model number
    - from Topics: dates, numbers, abbreviations, words, model number
    - from All Training Documents: dates, numbers, abbreviations, words, model number
    - from Documents with Relevance Judgments: dates, numbers, abbreviations, words, model number
  - Expansion of Queries using Previously-Constructed Data Structure:
    - Structure Used: word thesaurus vectors
  - Other: query terms were partitioned into topic-cherent groups called query factors. These factors were scored separately over documents and document segments, or TextTiles

Searching

Search Times

- Run ID : xerox1, xerox2
- Computer Time to Search (Average per Query, in CPU seconds): 150
- Component Times : build neural net input representation, score against trained neural net

- Run ID : xerox3, xerox4
- Computer Time to Search (Average per Query, in CPU seconds): 60
- Component Times : score and filter document used standard tf-idf method, apply probabilistic scoring to reorder documents

Machine Searching Methods

- Vector Space Model? : yes, for initial filtering
- Probabilistic Model? : yes, to reorder results
- Fuzzy Logic? : yes
- Neural Networks? : yes, for routing only

Factors in Ranking

- Term Frequency? : yes
- Inverse Document Frequency? : yes
- Other Term Weights? : TextTile scores, from document segments and Factor scores from query factors
- Semantic Closeness? : yes, based on word thesaurus vectors
- Proximity of Terms? : yes, though phrases
- Document Length? : yes
- Percentage of Query Terms which match? : yes
- Other: We tested various combinations of whole query vs query factors vs whole documents vs document TextTiles to generate scores. For example, one predictor scored documents according to the min score across factor of their maximum score across a document's TextTiles

Machine Information

- Machine Type for TREC Experiment: Sun Sparc Station 10
- Was the Machine Dedicated or Shared: Dedicated
- Amount of Hard Disk Storage (in MB): 7000
- Amount of RAM (in MB): 100
- Clock Rate of CPU (in MHz): 50

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System: variable. Some parts are the result of considerable engineering, other parts were custom for TREC 3
- Given appropriate resources
  - Could your system run faster? : yes
  - By how much (estimate)? : 100

System Summary and Timing
Organization Name: University of Massachusetts
List of Run ID's: INQ101, INQ102, INQ103, INQ104

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list: 419
- Controlled Vocabulary? : 4 (assigned automatically)
- Stemming Algorithm: Porter
- Term Weighting: tf.idf
- Phrase Discovery? : Yes, using PhraseFinder.
  - Kind of Phrase: Noun phrases of 3 words or less.
  - Method Used (statistical, syntactic, other): statistical
- Other Techniques for building Data Structures: PhraseFinder.

Statistics on Data Structures built from TREC Text

- Inverted index
  - Run ID : INQ101, INQ102
  - Total Storage (in MB): 1,285
  - Total Computer Time to Build (in hours): 26.2
  - Automatic Process? (If not, number of manual hours): Yes.
  - Use of Term Positions? : Yes.
  - Only Single Terms Used? : Yes.
- Inverted index
  - Run ID : INQ103, INQ104
  - Total Storage (in MB): 467
  - Total Computer Time to Build (in hours): 11.6
  - Automatic Process? (If not, number of manual hours): Yes.
  - Use of Term Positions? : Yes.
  - Only Single Terms Used? : Yes.
- Other Data Structures built from TREC text
  - Run ID : INQ101, INQ102
  - Type of Structure: PhraseFinder database
  - Total Storage (in MB): 530
  - Total Computer Time to Build (in hours): About 4 CPU days on a DEC 5000 workstation.
  - Automatic Process? (If not, number of manual hours): Yes.
  - Brief Description of Method: Index noun phrases by words that co-occur in a small window of text. Use normal retrieval algorithm to retrieve phrases. See RIAO-4 paper for details.

Data Built from Sources Other than the Input Text

- Externally-built Auxiliary File
  - Type of File (Treebank, WordNet, etc.): Gazetteer.
  - Total Storage (in MB): less than 1.
  - Number of Concepts Represented: US cities, foreign countries
  - Type of Representation: lists.

Query construction

Automatically Built Queries (Ad-Hoc)

- Topic Fields Used: Title, Description, Narrative.

- Average Computer Time to Build Query (in cpu seconds):
  Without query expansion, about 7 seconds (probably 1 second if optimized).
- Method used in Query Construction
  - Phrase Extraction from Topics? : Yes, noun phrases.
  - Syntactic Parsing of Topics? : Only part-of-speech tagging to find noun phrases.
  - Proper Noun Identification Algorithm? : Only part-of-speech tagging to find noun phrases.
  - Tokenizer? : Yes.
    - Patterns which are Tokenized: #city,#usa. Stop phrases are recognized and removed.
  - Expansion of Queries using Previously-Constructed Data Structure? : Yes, PhraseFinder added phrases to each query. (See paper.)
    - Structure Used: PhraseFinder database.
  - Automatic Addition of Boolean Connectors or Proximity Operators? : Yes. Noun phrases were broken into two word subphrases and placed in a #phrase operator.

Automatically Built Queries (Routing)

- Topic Fields Used: No.
- Method used in Query Construction
  - Terms Selected From
    - Only Documents with Relevance Judgments: Yes.
  - Term Weighting with Weights Based on terms in
    - All Training Documents: Yes.
  - Automatic Addition of Boolean connectors or Proximity Operators using information from
    - Documents with Relevance Judgments: Yes.

Manually Constructed Queries (Ad-Hoc)

- Topic Fields Used: Title, Description, Narrative.
- Average Time to Build Query (in Minutes): 5 minutes per query.
- Type of Query Builder
  - Computer System Expert: Yes.
- Method used in Query Construction
  - Term Weighting? : Yes.
  - Proximity Operators? : Yes.

Manually Constructed Queries (Routing)

- Topic Fields Used: An automatic query (INQ103) was combined with a manual query from TREC-2. No additional effort was required for TREC-3.
- Average Time to Build Query (in Minutes): 0.

Searching

Search Times

- Run ID : INQ101
- Computer Time to Search (Average per Query, in CPU seconds): 239

- Run ID : INQ102
- Computer Time to Search (Average per Query, in CPU seconds): 244

Machine Searching Methods

- Probabilistic Model? : Yes.

Factors in Ranking

- Term Frequency? : Yes.
- Inverse Document Frequency? : Yes.
- Other Term Weights? : Yes, query term weights, based on frequency of word in query.
- Proximity of Terms? : Yes, both in proximity operators and passage retrieval.
- Document Length? : Yes.

Machine Information

- Machine Type for TREC Experiment: DEC Alpha 3000-500.
- Was the Machine Dedicated or Shared: Shared.
- Amount of Hard Disk Storage (in MB): 40,716
- Amount of RAM (in MB): 128
- Clock Rate of CPU (in MHz): 175

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System: About 15 person years of ongoing research and development.
- Given appropriate resources
  - Could your system run faster? : Yes.
  - By how much (estimate)? : At least a factor of 2.

Significant Areas of System

System Summary and Timing
Organization Name: New York University
List of Run ID's: nyuir1, nyuir2

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list: 365
- Controlled Vocabulary? : no
- Stemming Algorithm: yes, lexicon
  - Morphological Analysis: partial
- Term Weighting: yes
- Phrase Discovery? : yes
  - Kind of Phrase: syntactic
  - Method Used (statistical, syntactic, other): syntactic with statistical disambiguation
- Syntactic Parsing? : yes
- Proper Noun Identification Algorithm? : yes
- Tokenizer? yes :
  - Patterns which are tokenized: names, fixed phrases
- Other Techniques for building Data Structures: domain map

Statistics on Data Structures built from TREC Text

- Inverted index
  - Run ID : nyuir
  - Total Storage (in MB): 804
  - Total Computer Time to Build (in hours): 20
  - Automatic Process? (If not, number of manual hours): yes
  - Use of Term Positions? : no
  - Only Single Terms Used? : no
- Clusters
  - Run ID : not used in TREC-3

Query construction

Automatically Built Queries (Ad-Hoc)

- Topic Fields Used: all
- Average Computer Time to Build Query (in cpu seconds): 2
- Method used in Query Construction
  - Term Weighting (weights based on terms in topics)? : yes
  - Phrase Extraction from Topics? : yes
  - Syntactic Parsing of Topics? : yes
  - Word Sense Disambiguation? : no
  - Proper Noun Identification Algorithm? : yes
  - Tokenizer? : yes
    - Patterns which are Tokenized: names, fixed phrases
  - Heuristic Associations to Add Terms? : no
  - Expansion of Queries using Previously-Constructed Data Structure? : yes, but not used in TREC-3
  - Automatic Addition of Boolean Connectors or Proximity Operators? :no

Automatically Built Queries (Routing)

- Topic Fields Used: desc, con

- Average Computer Time to Build Query (in cpu seconds): 2
- Method used in Query Construction
  - Terms Selected From
    - Topics: yes
    - Only Documents with Relevance Judgments: yes
  - Term Weighting with Weights Based on terms in
    - Topics: no
    - All Training Documents: yes
  - Phrase Extraction from
    - Topics: yes
    - All Training Documents: yes
  - Syntactic Parsing
    - Topics: yes
    - All Training Documents: yes
  - Proper Noun Identification Algorithm from
    - Topics: yes
    - All Training Documents: yes

Searching

Search Times

- Run ID : nyuir
- Computer Time to Search (Average per Query, in CPU seconds): 60

Machine Searching Methods

- Vector Space Model? : yes

Factors in Ranking

- Term Frequency? : yes
- Inverse Document Frequency? : yes
- Syntactic Clues? : phrases and names weighted differently
- Document Length? : yes

Machine Information

- Machine Type for TREC Experiment: Sun SparcStation 10
- Was the Machine Dedicated or Shared: dedicated
- Amount of Hard Disk Storage (in MB): 500
- Amount of RAM (in MB): 128

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System: substantial, new version installed
- Given appropriate resources
  - Could your system run faster? : yes
  - By how much (estimate)? : 8-10 times
- Features the System is Missing that would be beneficial: automatic feedback

System Summary and Timing
Organization Name: Dublin City University
List of Run ID's: DCUNL1 DCUNL2

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list: all words in the STOPWORD grammatical category of RUCL
- Stemming Algorithm:
  - Morphological Analysis: RUCL parser developed as part of SIMPR project
- Term Weighting: tf*IDF and a combination of lexical labels
- Phrase Discovery? :
  - Kind of Phrase: yes
  - Method Used (statistical, syntactic, other): syntactic
- Syntactic Parsing? : yes
- Word Sense Disambiguation? : no
- Proper Noun Identification Algorithm? : no
- Other Techniques for building Data Structures: lexical parsing of text using RUCL parser

Statistics on Data Structures built from TREC Text

- Inverted index
  - Run ID : DCUNL1 DCUNL2
  - Total Storage (in MB): 157
  - Total Computer Time to Build (in hours): 50
  - Automatic Process? (If not, number of manual hours): yes
  - Use of Term Positions? : no
  - Only Single Terms Used? : no
- Clusters
- N-grams, Suffix arrays, Signature Files
- Knowledge Bases
- Special Routing Structures
- Other Data Structures built from TREC text
  - Run ID : DCUNL1 DCUNL2
  - Type of Structure: Tree Structured Analytics (tree representations of syntactic structure of clauses)
  - Total Storage (in MB): 672
  - Total Computer Time to Build (in hours): 2200
  - Automatic Process? (If not, number of manual hours): yes
  - Brief Description of Method: lexical parsing of text using RUCL parser. This parsed text is then converted into sets of TSAs

Query construction

Automatically Built Queries (Ad-Hoc)

- Topic Fields Used: narrative and description
- Method used in Query Construction
  - Term Weighting (weights based on terms in topics)? : tf*IDF and lexical label weighting
  - Phrase Extraction from Topics? : yes
  - Syntactic Parsing of Topics? : yes

- Other: construction of TSAs from queries to replicate the syntactic structure used in query text

Searching

Search Times
- Run ID : DCUNL1 DCUNL2
- Computer Time to Search (Average per Query, in CPU seconds): 1800
- Component Times :
  parsing of query : 240
  presearch using tf*IDF index : 300
  (threshold of 17% of total # of docs.)
  TSA matching : 1260

Machine Searching Methods

- Other: our approach is a combination of VSM and tree structure matching where the tree structures are derived from syntax.

Factors in Ranking

- Factors in Ranking
  - Term Frequency? : yes
  - Inverse Document Frequency? : yes
  - Other Term Weights? : yes
  - Syntactic Clues? : yes
  - Proximity of Terms? : yes, by finding the common ancestor in the TSAs to see if two matched terms are in a similar syntactic construct
  - Document Length? : yes, normalise by the number of clauses in the document

Machine Information

- Machine Type for TREC Experiment: Sun Server 690MP
- Was the Machine Dedicated or Shared: Very Shared
- Amount of Hard Disk Storage (in MB): 14,000
- Amount of RAM (in MB): 128
- Clock Rate of CPU (in MHz): don't know

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System: Excluding the development of the RUCL parser, all software was developed in-house and the software engineering took 4 months with about 15 months of background research and experimentation
- Given appropriate resources
  - Could your system run faster? : yes ... much
  - By how much (estimate)? : many orders of magnitude ... the system used is a demonstrator prototype with no effort put into optimising the execution speeds at all.
- Features the System is Missing that would be beneficial: word-word conceptual similarity scoring is where our next efforts will be put

Significant Areas of System

- Brief Description of features in your system which you feel impact the system and are not answered by above questions:

  Our system is implemented as a two-phase operation. A pre-search based on tf*IDF weighting of word baseforms gives us the top 1000 documents and these 1000 documents are then re-ranked using our TSA-based scoring mechanism. This implementation strategy was adopted for efficiency reasons.

System Summary and Timing
Organization Name: Cornell University
List of Run ID's: Crnl1RR Crnl1QR Crnl1EA Crnl1LA Crnl1ES Crnl1VS

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list:
  571 English, 342 Spanish
- Stemming Algorithm: English - Modified Lovins, Spanish - ad-hoc
  - Morphological Analysis: Nothing Additional
- Term Weighting: Cornell's "lnc" and "ltc" weights
- Phrase Discovery? : Yes
  - Kind of Phrase: Adjacent words
  - Method Used (statistical, syntactic, other):
    Any pair of adjacent non-stopwords that
    occurred 25 times in D1
- Proper Noun Identification Algorithm? : Exists, not used
- Tokenizer? :
  - Patterns which are tokenized: nothing special

Statistics on Data Structures built from TREC Text

- Inverted index
  - Run ID : Crnl1EA, Crnl1LA (D12,ad-hoc)
  - Total Storage (in MB): 863
  - Total Computer Time to Build (in hours): 3.6
  - Automatic Process? (If not, number of manual hours): yes
  - Use of Term Positions? : no
  - Only Single Terms Used? : no

- Inverted index
  - Run ID : Crnl1QR, Crnl1RR (D3,routing)
  - Total Storage (in MB): 395
  - Total Computer Time to Build (in hours): 1.8
  - Automatic Process? (If not, number of manual hours): yes
  - Use of Term Positions? : no
  - Only Single Terms Used? : no

- Inverted index
  - Run ID : Crnl1VS, Crnl1ES (Spanish)
  - Total Storage (in MB): 84
  - Total Computer Time to Build (in hours): 0.38
  - Automatic Process? (If not, number of manual hours): yes
  - Use of Term Positions? : no
  - Only Single Terms Used? : yes

- Special Routing Structures
  - Run ID : Crnl1QR, Crnl1ER
  - Type of Structure: Occurrence statistics for the most
    frequently occurring (in learning set rel docs) 1000
    terms for each routing query.
  - Total Storage (in MB): 1.5
  - Total Computer Time to Build (in hours): 1.0
  - Automatic Process? (If not, number of manual hours): yes

- Special Routing Structures

- Run ID : CrnlER
- Type of Structure: For each query, history of how well 7 different feedback runs worked on it (learning on D1, testing on D2)
- Total Storage (in MB): 0.1
- Total Computer Time to Build (in hours): 1.1
- Automatic Process? (If not, number of manual hours): yes

Query construction

Automatically Built Queries (Ad-Hoc)

- Topic Fields Used: title, nationality, narrative, factors, descriptions, concepts
- Average Computer Time to Build Query (in cpu seconds): .03
- Method used in Query Construction
  - Term Weighting (weights based on terms in topics)? : yes
  - Phrase Extraction from Topics? : yes
  - Tokenizer? :
    - Patterns which are Tokenized: recognized and discarded constructs of general form "relevant documents {consist|contain|include|...}"

Automatically Built Queries (Routing)

- Topic Fields Used: title, nationality, narrative, factors, descriptions, concepts
- Average Computer Time to Build Query (in cpu seconds): .04
- Method used in Query Construction
  - Terms Selected From
    - Topics: yes
    - Only Documents with Relevance Judgments: yes
  - Term Weighting with Weights Based on terms in
    - Topics: yes
    - All Training Documents: yes
  - Phrase Extraction from
    - Topics: yes
    - Only documents with Relevance Judgments: yes
  - Tokenizer
    - from Topics: recognized and discarded constructs of general form "relevant documents {consist|contain|include|...}"

Searching

Search Times

- Run ID : CrnlEA
- Computer Time to Search (Average per Query, in CPU seconds): 71
- Component Times : Initial search (getting top 30 docs) 2.5 seconds. Expansion of query by 510 terms in top 30 docs, 1 second. Running expanded query, 36 seconds Keeping track of top 1000 docs, 31 seconds (the bottom hundred docs are constantly churning)

- Run ID : CrnlLA
- Computer Time to Search (Average per Query, in CPU seconds): 189
- Component Times : Initial global Search getting top 1750 docs, 8.5 seconds. Reindexing each top doc, splitting into 200

word overlapping windows,comparing query with each window, adding global sim with best local sim, 181 seconds.

- Run ID : Crnl RR
- Computer Time to Search (Average per Query, in CPU seconds): 43 seconds
- Component Times : Calculation of similarities (average of 394 terms per query), 9 seconds.Keeping track of top 1000 docs, 34 seconds.

- Run ID : Crnl QR
- Computer Time to Search (Average per Query, in CPU seconds): 36
- Component Times : Calculation of similarities (average of 372 terms per query), 11 seconds. Keeping track of top 1000 docs, 25 seconds.Note. Each query was a separate invocation of the SMART.

- Run ID : Crnl VS
- Computer Time to Search (Average per Query, in CPU seconds): 3.2 seconds
- Component Times : Calculations of similarities .08 seconds Keeping track of top 1000 docs, 3.12 seconds

- Run ID : Crnl ES
- Computer Time to Search (Average per Query, in CPU seconds): 27.1
- Component Times : Calculations of similarities (average of 525.9 terms per query), 1.3 seconds Keeping track of top 1000 docs, 25.8 seconds (the bottom hundred docs are constantly churning!)

Machine Searching Methods

- Vector Space Model? : yes
- Probabilistic Model? : yes (some queries in Crnl QR)

Factors in Ranking

- Term Frequency? : yes
- Inverse Document Frequency? : yes
- Other Term Weights? : weights in learning docs for routing. weights in top retrieved docs for Crnl EA
- Proximity of Terms? : to form adjacent phrase terms
- Information Theoretic Weights? : yes in Crnl QR
- Document Length? : yes? (cosine normalization)

Machine Information

- Machine Type for TREC Experiment: Sparc 20/51
- Was the Machine Dedicated or Shared: Dedicated
- Amount of Hard Disk Storage (in MB): 18 Gbytes
- Amount of RAM (in MB): 160 Mbytes
- Clock Rate of CPU (in MHz): 50?

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System: 7 Person years?
- Given appropriate resources
  - Could your system run faster? : yes
  - By how much (estimate)? : Indexing, not much. Retrieval, all runs can be sped up by a factor of 20 using retrieval optimization techniques.

System Summary and Timing
Organization Name: CITRI
List of Run ID's: citri1, citri2

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list: 0
- Stemming Algorithm: Lovins
- Term Weighting: XX

Statistics on Data Structures built from TREC Text

- Inverted index
  - citri1
  - Total Storage (in MB): 132
  - Total Computer Time to Build (in hours): 4
  - Automatic Process (Number of Manual Hours): 0
  - Use of Term Positions: no
  - Only Single Terms Used: yes
- Inverted index
  - citri2
  - Total Storage (in MB): 27
  - Total Computer Time to Build (in hours): 4
  - Automatic Process (Number of Manual Hours): 0
  - Use of Term Positions: no
  - Only Single Terms Used: yes
- Other Data Structures built from TREC text
  - Type of Structure: supplementary inverted file for citri2
  - Total Storage (in MB): 121
  - Total Computer Time to Build (in hours): 4
  - Automatic Process (Number of Manual Hours): 0
  - Brief Description of Method: main index on blocks of 100 records, supplementary index for detail withing blocks
- Other Data Structures built from TREC text
  - Type of Structure: compressed form of trec text
  - Total Storage (in MB): 670
  - Total Computer Time to Build (in hours): 4
  - Automatic Process (Number of Manual Hours): 0
  - Brief Description of Method: word-based model, semi-static Huffman coding

Query construction

Automatically Built Queries (Ad-Hoc)

- Topic Fields Used: all
- Average Computer Time to Build Query (in cpu seconds): negligible
- Method used in Query Construction
  - Term Weighting (weights based on terms in topics): XX

Searching

Search Times

- Run ID: citri1
- Total Computer Time to Search (in CPU seconds): 22

- Retrieval Time (in CPU seconds): 22
- Ranking Time (in CPU seconds): 0

- Run ID: citri2
- Total Computer Time to Search (in CPU seconds): 10
- Retrieval Time (in CPU seconds): 10
- Ranking Time (in CPU seconds): 0

Machine Searching Methods

- Vector Space Model: yes

Factors in Ranking

- Term Frequency: yes
- Inverse Document Frequency: yes
- Document Length: yes

Machine Information

- Machine Type for TREC Experiment: Sparc 10 model 512
- Was the Machine Dedicated or Shared: shared
- Amount of Hard Disk Storage (in MB): 10 Gb
- Amount of RAM (in MB): 160 Mb
- Clock Rate of CPU (in MHz): 55

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System: moderate
- Given appropriate resources
  - Could your system run faster: yes
  - By how much (estimate): factor of 5, using optimisations described last year
- Features the System is Missing that would be beneficial: provision for interactive use, relevance feedback

Significant Areas of System

- Brief Description of features in your system which you feel impact the system and are not answered by above questions:

  This year's runs are a simulation of a distributed system, so times are artificially high and some quantities (such as amount of index processed) can't be measured because unnecessary work is being done

System Summary and Timing
Organization Name: Siemens Corporate Research, Inc.
List of Run ID's: siems1 siems2

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list: 571
- Stemming Algorithm: standard SMART
- Term Weighting: yes (cosine-normalized tf)
- Phrase Discovery? : 011

Statistics on Data Structures built from TREC Text

- Inverted index
  - Run ID : siems1, siems2
  - Total Storage (in MB): 201
  - Total Computer Time to Build (in hours): 1.8
  - Automatic Process? (If not, number of manual hours): yes
  - Use of Term Positions? : no
  - Only Single Terms Used? : yes

- Inverted index
  - Run ID : siems1, siems2
  - Total Storage (in MB): 95
  - Total Computer Time to Build (in hours): .75
  - Automatic Process? (If not, number of manual hours): yes
  - Use of Term Positions? : no
  - Only Single Terms Used? : yes

- Inverted index
  - Run ID : siems1, siems2
  - Total Storage (in MB): 72
  - Total Computer Time to Build (in hours): 1.5
  - Automatic Process? (If not, number of manual hours): yes
  - Use of Term Positions? : no
  - Only Single Terms Used? : yes

- Inverted index
  - Run ID : siems1, siems2
  - Total Storage (in MB): 194
  - Total Computer Time to Build (in hours): 1.4
  - Automatic Process? (If not, number of manual hours): yes
  - Use of Term Positions? : no
  - Only Single Terms Used? : yes

- Inverted index
  - Run ID : siems1, siems2
  - Total Storage (in MB): 130
  - Total Computer Time to Build (in hours): 2.2
  - Automatic Process? (If not, number of manual hours): yes
  - Use of Term Positions? : no
  - Only Single Terms Used? : yes

- Knowledge Bases
  - Automatic Process? (If not, number of manual hours): yes

Data Built from Sources Other than the Input Text

- Internally-built Auxiliary File
  - Domain (independent or specific): collection specific
  - Type of File (thesaurus, knowledge base, lexicon, etc.):
    siems1: ranks of relevant docs in training queries
    siems2: training query clusters
  - Total Storage (in MB):
    siems1: ~ .1 per collection
    siems2: ~ .025 per collection
  - Total Computer Time to Build (in hours):
    ~.5 per collection including doing retrieval

Query construction

Automatically Built Queries (Ad-Hoc)

- Topic Fields Used: title, nat, narr, fac, desc, con
- Average Computer Time to Build Query (in cpu seconds):
  .03 per collection
- Method used in Query Construction
  - Term Weighting (weights based on terms in topics)? : yes,
    (cosine-normalized tf*idf, within each collection)

Searching

Search Times

- Run ID : siems1
- Computer Time to Search (Average per Query, in CPU seconds):
  42 (assuming parallel execution of searches)
- Component Times :
  index query in fusion's query collection -- .03 (once)
  index query in 5 subcollections -- .03 (per collection)
  retrieve 1000 docs on each subcollection -- 11 (slowest single collection)
  fuse --30 (once)

- Run ID : siems2
- Computer Time to Search (Average per Query, in CPU seconds):
  12 (assuming parallel execution of searches)
- Component Times :
  index query in 5 subcollections -- .03 (per collection)
  retrieve 1000 docs on each subcollection -- 11 (slowest single)
  fuse -- .5

Machine Searching Methods

- Machine Searching Methods
  - Vector Space Model? : yes
  - Cluster Searching? : query clusters in siems2

Factors in Ranking

- Factors in Ranking
  - Term Frequency? : yes
  - Inverse Document Frequency? : yes
  - Document Length? : yes

- Other: in siems1, expected usefulness of subcollection

Machine Information

- Machine Type for TREC Experiment: SPARC-10/41
- Was the Machine Dedicated or Shared: mostly dedicated
- Amount of Hard Disk Storage (in MB): ~ 13,000
- Amount of RAM (in MB): 128
- Clock Rate of CPU (in MHz): 40 MHz

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System:
  individual subcollection retrieval done by SMART, which is a well-tuned research prototype;
  fusion code experimental with almost no tuning

- Given appropriate resources
  - Could your system run faster? : yes
  - By how much (estimate)? : with some effort, fusion time for method in siems1 could be halved

System Summary and Timing
Organization Name: Virginia Tech
List of Run ID's: VTc5s2, VTc2s2

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list: 418
- Stemming Algorithm: Plural Removal, as provided by SMART.
- Term Weighting: wgt = 0.5+0.5 * tf/max_tf(doc). SMART "ann" weighting
- Tokenizer? : As provided by SMART.
  - Patterns which are tokenized: dates, times
- Other Techniques for building Data Structures: Source text was preparsed from SGML to SMART format.

Statistics on Data Structures built from TREC Text

- Document Vector Files
  - Run ID : VTc5s2, VTc2s2
  - Total Storage (in MB): 1100
  - Total Computer Time to Build (in hours): 75
  - Automatic Process? (If not, number of manual hours): Yes
  - Use of Term Positions? : No
  - Only Single Terms Used? : Yes

Query construction

Manually Constructed Queries (Ad-Hoc)

- Topic Fields Used: title, description, narrative
- Average Time to Build Query (in Minutes): 10
- Type of Query Builder
  - Computer System Expert: Yes
- Method used in Query Construction
  - Term Weighting? : Yes
  - Boolean Connectors (AND, OR, NOT)? : Yes; Pnorm queries only.
  - Addition of Terms not Included in Topic? : Yes (Pnorm and one set of vector queries).
    - Source of Terms: General knowledge of computer system expert, limited use to compensate for obvious omissions in text of topic descriptions.
  - Other: Three sets of queries were constructed: one pnorm boolean query set and two different length vector query sets.

Searching

Search Times

- Search Times
  - Run ID : VTc5s2, VTc2s2
  - Computer Time to Search (Average per Query, in CPU seconds): 2750
  - Component Times : Average 5 minutes per query for each of the nine collections, 50 seconds for combination of collection results.

Machine Searching Methods

- Machine Searching Methods
  - Vector Space Model? : Yes

- Boolean Matching? : Yes; P-norm soft boolean evaluation.
- Other: Combination of results from both pnorm and vector queries.

Factors in Ranking

- Factors in Ranking
  - Term Frequency? : Yes
  - Other Term Weights? : Yes; maximum term weight in document.
  - Document Length? : Yes; Indirectly via use of maximum term frequency.

Machine Information

- Machine Type for TREC Experiment: DEC station 5000/125
- Was the Machine Dedicated or Shared: Shared
- Amount of Hard Disk Storage (in MB): 1000
- Amount of RAM (in MB): 40
- Clock Rate of CPU (in MHz): 25

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System: None for TREC3
- Given appropriate resources
  - Could your system run faster? : Yes
  - By how much (estimate)? : 2-4 times faster

Significant Areas of System

- Brief Description of features in your system which you feel impact the system and are not answered by above questions: Basic system was 1985 version of SMART, with many enhancements (ie, pnorm query processing) added from previous projects before TREC.

Software development during TREC1 and TREC2 consisted of creation of external routines for merging/combining the results from individual SMART retrieval runs, and adding support for multiple query and index processing during a single retrieval run. Collections were indexed and searched separately; values above for index creation and query search times are combined totals for building or searching all collections.

System Summary and Timing
Organization Name: City University
List of Run ID's: citya1 citya2 cityr1 cityr2 cityi1

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list:
  247 (citya1 citya2)
- Length (in words) of the stopword list:
  247 + 226 semi-stopwords (cityr1 cityr2 cityi1)
- Stemming Algorithm: Modified Porter with spelling normalization

Statistics on Data Structures built from TREC Text

- Inverted index
  - Run ID : citya1 citya2
  - Total Storage (in MB): 1200
  - Total Computer Time to Build (in hours): about 25 hours
  - Automatic Process? (If not, number of manual hours): Yes
  - Use of Term Positions? : Yes
  - Only Single Terms Used? : Some prespecified phrases are mapped to tokens
- Inverted index
  - Run ID : cityr1 cityr2 cityi1
  - Total Storage (in MB): 530
  - Total Computer Time to Build (in hours): about 14 hours
  - Automatic Process? (If not, number of manual hours): Yes
  - Use of Term Positions? : Yes
  - Only Single Terms Used? : Some prespecified phrases are mapped to tokens

Data Built from Sources Other than the Input Text

- Internally-built Auxiliary File
  - Domain (independent or specific): Somewhat angled towards American data
  - Type of File (thesaurus, knowledge base, lexicon, etc.): Contains synonym classes, go phrases, stop- and semi-stopwords, prefixes
  - Total Storage (in MB): << 1
  - Number of Concepts Represented: About 1500
  - Type of Representation: Lookup table
  - Total Manual Time to Build (in hours): Not recorded,done haphazardly. Much of it is common to many databases
  - Total Manual Time to Modify for TREC (if already built): Not known
  - Use of Manual Labor
    - Other: Yes

Query construction

Automatically Built Queries (Ad-Hoc)

- Run Id: citya1
- Topic Fields Used: Title, narrative, description
- Average Computer Time to Build Query (in cpu seconds): About 500
- Method used in Query Construction

- Term Weighting (weights based on terms in topics)? : See below
- Other: Addition of terms from top documents retrieved by trial search using topic fields. Terms weighted based on term frequency in topics and in documents, collection frequency, and frequency in documents retrieved by trial search.

Automatically Built Queries (Ad-Hoc)

- Run Id: citya2
- Topic Fields Used: Title, narrative, description
- Average Computer Time to Build Query (in cpu seconds): About 2
- Method used in Query Construction
  - Term Weighting (weights based on terms in topics)? : See below
  - Other: Terms weighted based on term frequency in topics and in documents, and collection frequency.

Automatically Built Queries (Routing)

- Run Id: cityr1 cityr2
- Topic Fields Used: None
- Average Computer Time to Build Query (in cpu seconds): Thousands
- Method used in Query Construction
  - Terms Selected From
    - Only Documents with Relevance Judgments: Yes (only from relevant documents)
  - Term Weighting with Weights Based on terms in
    - Only documents with Relevance Judgments: Yes (only relevant documents).

Interactive Queries

- Initial Query Built Automatically or Manually: Manually
- Type of Person doing Interaction
  - Other: Information scientists and graduate students.
- Average Time to do Complete Interaction
  - CPU Time (Total CPU Seconds for all Iterations): Unknown
  - Clock Time from Initial Construction of Query to Completion of Final Query (in minutes): 39
- Average Number of Iterations: 1.5
- Average Number of Documents Examined per Iteration: 18.5
- Minimum Number of Iterations: 0
- Maximum Number of Iterations: 3
- What Determines the End of an Iteration: searcher decision
- Methods used in Interaction
  - Automatic Term Reweighting from Relevant Documents? : Yes
  - Automatic Query Expansion from Relevant Documents? : Yes
    - Only Top X Terms Added (what is X): Yes; 20
    - User Selected Terms Added: Yes
  - Manual Methods
    - Using Individual Judgment (No Set Algorithm)? : Yes

Searching

- Search Times
  - Run ID : citya1 citya2 cityr1 cityr2
  - Computer Time to Search (Average per Query, in CPU seconds): Varies enormously, from about 10 seconds

with 3 or 4 terms up to a few minutes. Would guess overall average about 60 on SS10

  - Run ID : cityi1
  - Computer Time to Search (Average per Query, in CPU seconds): Unknown

Machine Searching Methods

- Machine Searching Methods
  - Probabilistic Model? : Yes

Factors in Ranking

- Factors in Ranking
  - Term Frequency? : Yes
  - Inverse Document Frequency? : Yes
  - Other Term Weights? : From relevance information when available
  - Document Length? : Yes

Machine Information

- Machine Type for TREC Experiment: Suns. SS10, 4/330, IPX
- Was the Machine Dedicated or Shared: SS10 dedicated, others shared.
- Amount of Hard Disk Storage (in MB): About 11 GB for TREC
- Amount of RAM (in MB): 64, 40, 16 resp.
- Clock Rate of CPU (in MHz): N/K

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System: A lot, but very little which is TREC-specific.
- Given appropriate resources
  - Could your system run faster? : Of course.
  - By how much (estimate)? : Order of magnitude?
- Features the System is Missing that would be beneficial: Factors X and Y.

Significant Areas of System

- Brief Description of features in your system which you feel impact the system and are not answered by above questions:

  Both the databases had enough within-document information to allow searching sub-documents consisting of any run of algorithmically-determined paragraphs; this facility was used in the citya1 run.

System Summary and Timing
Organization Name: U. of California, Berkeley
List of Run ID's: Brkly6, Brkly7, Brkly8

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list: 592
- Stemming Algorithm: SMART STEMMER
- Term Weighting: YES

Statistics on Data Structures built from TREC Text

- Inverted index
  - Run ID : Brkly6, Brkly7
  - Total Storage (in MB): 659
  - Total Computer Time to Build (in hours): APPROX. 80
  - Automatic Process? (If not, number of manual hours): YES
  - Use of Term Positions? : NO
  - Only Single Terms Used? : YES
- Inverted index
  - Run ID : Brkly8
  - Total Storage (in MB): 364
  - Total Computer Time to Build (in hours): APPROX. 40
  - Automatic Process? (If not, number of manual hours): YES
  - Use of Term Positions? : NO
  - Only Single Terms Used? : YES

Query construction

Automatically Built Queries (Ad-Hoc)

- Topic Fields Used: TITLE, DESC, NARR
- Average Computer Time to Build Query (in cpu seconds): APPROX. 3 / QUERY
- Method used in Query Construction
  - Term Weighting (weights based on terms in topics)? : YES
  - Other: ABSOLUTE QUERY STEM FREQUENCIES, QUERY LENGTHS, AND IDF'S WERE USED IN WEIGHTING THE QUERY STEMS

Automatically Built Queries (Routing)

- Topic Fields Used: DOM, TITLE, DESC, NARR, CON, DEF, NAT TIME
- Average Computer Time to Build Query (in cpu seconds): APPROX. 100 / QUERY
- Method used in Query Construction
  - Terms Selected From
    - Topics: YES
    - Only Documents with Relevance Judgments: YES
  - Term Weighting with Weights Based on terms in
    - Topics: YES
    - Only documents with Relevance Judgments: YES

Manually Constructed Queries (Ad-Hoc)

- Topic Fields Used: TITLE, DESC, NARR
- Average Time to Build Query (in Minutes): 20-30
- Tools used to Build Query
  - Other Lexical Tools? : Thesaurus allowed but not used much.

- Method used in Query Construction
  - Addition of Terms not Included in Topic? : YES
    - Source of Terms: Boolean lookups in parallel collections

Searching

Search Times

- Run ID : Brkly6, Brkly7
- Computer Time to Search (Average per Query, in CPU seconds):
  APPROX 16 SECONDS PER QUERY TO SEARCH ANY ONE OF THE FIVE COLLECTIONS SEPARATELY.

- Run ID : Brkly8
- Computer Time to Search (Average per Query, in CPU seconds):
  APPROX. 300

Machine Searching Methods

- Machine Searching Methods
  - Probabilistic Model? : YES

Factors in Ranking

- Factors in Ranking
  - Term Frequency? : YES
  - Inverse Document Frequency? : YES
  - Document Length? : YES

Machine Information

- Machine Type for TREC Experiment: DEC STATION 5000/125
- Was the Machine Dedicated or Shared: Dedicated
- Amount of Hard Disk Storage (in MB): 4 GB
- Amount of RAM (in MB): 48
- Clock Rate of CPU (in MHz): 15 MHZ

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System:
  NONE EXCEPT FOR THE PROBABILISTIC LOGIC. THE BERKELEY SYSTEM IS AN EXPERIMENTAL PROTOTYPE ONLY, PROGRAMMED AS A MINIMAL MODIFICATION OF THE SMART SYSTEM (VERSION 10).

- Given appropriate resources
  - Could your system run faster? : NATURALLY

System Summary and Timing
Organization Name: Dortmund

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list:577 words
- Stemming Algorithm: Standard SMART stemmer
- Term Weighting: Yes
- Phrase Discovery? :
  - Kind of Phrase: Any pair of adjacent non-stopwords that occurred 25 times in D1
- Other Techniques for building Data Structures:we incorporated the manually assigned index-terms appearing in ziff material (the SGML-tag DESCRIPT) in the indexing process, because we considered it part of the documents. The SGML-tags MS and NS in the wsj material are not included.

Statistics on Data Structures built from TREC Text

- Inverted index
  - Total Storage (in MB):~460 MB for D3 (test set for routing)
  --~900 MB for D12 (learning set for routing and test set for ad-hoc)
  - Total Computer Time to Build (in hours): ~5 CPU hours for D3
  - ~10 CPU hours for D12
  - Use of Term Positions? : No
  - Only Single Terms Used? :No

Query construction

Automatically Built Queries (Ad-Hoc)

- Topic Fields Used:Everything except Definitions
- Average Computer Time to Build Query (in cpu seconds):~30/60 CPU seconds for each query
- Method used in Query Construction
  - Term Weighting (weights based on terms in topics)? :Yes
  - Phrase Extraction from Topics? :Yes

Machine Information

- Machine Type for TREC Experiment: Sparc 10
- Amount of Hard Disk Storage (in MB): 6300 MB
- Amount of RAM (in MB): 157 MB

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System: Several years

System Summary and Timing
Organization Name: Rutgers University, Interactive Searching
List of Run ID's: rutir1, rutir2

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures : PLEASE SEE THE SYSTEM SUMMARY AND TIMING FOR THE INQUERY SYSTEM FROM UNIVERSITY OF MASSACHUSETTS FOR THESE DATA

Statistics on Data Structures built from TREC Text: PLEASE SEE THE SYSTEM SUMMARY AND TIMING FOR THE INQUERY SYSTEM FROM UNIVERSITY OF MASSACHUSETTS FOR THESE DATA

Query construction

Manually Constructed Queries (Ad-Hoc)

- Topic Fields Used: ALL

Manually Constructed Queries (Routing)

- Average Time to Build Query (in Minutes): 15.48
- Type of Query Builder
  - Computer System Expert: YES (rutir2)
  - Searching Expert: YES (rutir1)
- Data Used for Building Query from
  - Training Topics: YES
  - All Training Documents: YES
  - Only documents with Relevance Judgments: YES
- Method used in Query Construction
  - Term Weighting? : YES (NO SEARCHER CONTROL)
  - Proximity Operators? : YES
  - Addition of Terms not Included in Topic? : YES
    - Source of Terms: SEARCHER
  - Other: SYNONYM OPERATOR

Interactive Queries

- Initial Query Built Automatically or Manually: MANUALLY
- Type of Person doing Interaction
  - System Expert: YES (rutir2)
  - Searching Expert: YES (rutir1)
- Average Time to do Complete Interaction ALL DATA FOR RUTIR1
  - Clock Time from Initial Construction of Query to Completion of Final Query (in minutes): 928.553
- Average Number of Iterations: 7.760
- Minimum Number of Iterations: 2
- Maximum Number of Iterations: 21
- What Determines the End of an Iteration: CLICKING THE "RUN QUERY" 011 BUTTON (OR SAVING THE "FINAL QUERY")
- Methods used in Interaction
  - Automatic Term Reweighting from Relevant Documents? : YES
  - Automatic Query Expansion from Relevant Documents? : YES
    - Only Top X Terms Added (what is X): 5
    - User Selected Terms Added: YES
  - Manual Methods
    - Using Individual Judgment (No Set Algorithm)? : YES

Searching

Search Times

- Run ID : rutir1
- Computer Time to Search (Average per Query, in CPU seconds): 19.288
- Component Times : NOTE THAT THESE ARE TIMES TO RUN THE QUERY MADE IN EACH ITERATION OF EACH SEARCH, FOR THE TRAINING CONDITION. COMPONENTS ARE SYSTEM TIME AND CPU TIME

Machine Searching Methods

- Machine Searching Methods
  - Probabilistic Model? : YES

Factors in Ranking

- Factors in Ranking
  - Term Frequency? : YES
  - Inverse Document Frequency? : YES
  - Proximity of Terms? : YES

Machine Information

- Machine Type for TREC Experiment: SUN10/51
- Was the Machine Dedicated or Shared: SHARED
- Amount of Hard Disk Storage (in MB): 6,000
- Amount of RAM (in MB): 96
- Clock Rate of CPU (in MHz): 50

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System: 3 PERSON-MONTHS
- Given appropriate resources
  - Could your system run faster? : YES
  - By how much (estimate)? : 50%

System Summary and Timing
Organization Name: Department of Industrial Engineering,
University of Toronto

List of Run ID's: stpat1

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list: 24 words
- Stemming Algorithm: No
  - Morphological Analysis: No

Statistics on Data Structures built from TREC Text

- Inverted index
  - Run ID : stpat1
  - Total Storage (in MB): about 370 MB (250MB TREC text plus 120MB of index)
  - Total Computer Time to Build (in hours): about 10 hours (elapsed time)
  - Automatic Process? (If not, number of manual hours): Yes
  - Use of Term Positions? : No
  - Only Single Terms Used? : No

Query construction

Manually Constructed Queries (Routing)

- Topic Fields Used: title, desc, smry, con
- Average Time to Build Query (in Minutes): 30 minutes
- Type of Query Builder
  - Domain Expert: no
  - Computer System Expert: Yes (perhaps). the queries were constructed by the two researchers for this project. In comparison with other TREC-3 researchers we might not be consider computer system experts.
- Tools used to Build Query
  - Other Lexical Tools? : Webster's online dictionary (hardly ever used in practice), Wordnet (available but not used)
- Data Used for Building Query from
  - Training Topics: No. We didn't use a separate set of training topics. We simply used the test topics on the training database for training.
  - All Training Documents: No. Only WSJ text were used.
  - Only documents with Relevance Judgments: Yes. We used these to construct measures of precision and recall that were shown to the user each time the query was modified. The user was also able to review the list of relevant documents that were identified by the experts in the training database.
- Method used in Query Construction
  - Term Weighting? : No.
  - Boolean Connectors (AND, OR, NOT)? : Yes
  - Proximity Operators? : Yes
  - Addition of Terms not Included in Topic? : Yes
    - Source of Terms: topic definitions, documents in training database, coordinate terms, user memory

Interactive Queries

- Initial Query Built Automatically or Manually: Manually

- Type of Person doing Interaction
  - Domain Expert: No
  - System Expert: Yes
- Average Time to do Complete Interaction
  - CPU Time (Total CPU Seconds for all Iterations): considerably less than 30 minutes, possibly between 1 and 5 minutes, but the exact value is not available. We didn't distinguish between CPU time and clock time in making our logs.
  - Clock Time from Initial Construction of Query to Completion of Final Query (in minutes): 32.44 minutes (on average)
- Average Number of Iterations: 8.9. We found it hard to define precisely what was an iteration, since this depends to a large part on the user's intention. In our system, the query was executed every time it was modified. However, we tried to assess an iteration as a "new" query rather than a simple adjustment which was one step in a sequence of adjustments to build a new iteration of the query.
- Average Number of Documents Examined per Iteration: 18.18
- Minimum Number of Iterations: 3
- Maximum Number of Iterations: 18
- What Determines the End of an Iteration: User evaluation of the effectiveness of the query followed by a change in query. Thus the definition of what an iteration is, is based entirely on the user, not the system.
  - Manual Methods
    - Using Individual Judgment (No Set Algorithm)? : Yes. In practice there was a tendency to construct a query that had high recall and then add terms to improve precision. If this didn't work the strategy was to identify subqueries with high precision and then OR them together to create a final query that had good precision and recall.
    - Following a Given Algorithm (Brief Description)? : No

Searching

Search Times

- Run ID : stpat1
- Computer Time to Search (Average per Query, in CPU seconds): not available. The CPU search time was of less interest to us than the elapsed time. The search time was determined by the text retrieval engine that we were using as an off-the-shelf component of our system. We believe that we could fairly easily substitute a different retrieval engine with approximately the same features and get similar results for our interactive experiments. The whole philosophy of our approach is that performance of this style of interactive search is largely driven by the properties and features of the user interface.
- Component Times : not applicable.

Machine Searching Methods

- Boolean Matching? : Yes
- Free Text Scanning? : Yes

Factors in Ranking

- Term Frequency? : Yes
- Document Length? : Yes
- Percentage of Query Terms which match? : Yes

Machine Information

- Was the Machine Dedicated or Shared: Dedicated
- Amount of Hard Disk Storage (in MB): approximately half a gigabyte used on a 2 gigabyte disk
- Amount of RAM (in MB): 32 MB
- Clock Rate of CPU (in MHz): 50

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System: Half a person year of non-professional student programming. No formal software engineering process.

- Given appropriate resources
  - Could your system run faster? : Yes
  - By how much (estimate)? :

  At least twice as fast. Even if the system ran twice as fast it probably wouldn't make much difference, as thinking time, and query formulation time are the major issues. Thus getting better functionality and simplifying the user's tasks are probably better areas to look at for increasing overall system efficiency. No effort has been put into the optimization problem. Instead we attempted to identify the most useful functionality for the user based on an iterative design approach. We never stayed with one iteration long enough to justify a major optimization effort. One feature of our system is that it was originally developed as a browsing system. The information retrieval tools were specifically added to meet the needs of TREC-3. These changes were made in a hurry and not fully tested prior to the TREC-3 evaluation, thus we expect that considerable improvements can be made to the system which should lead to greatly improved retrieval effectiveness.

- Features the System is Missing that would be beneficial: There are a number of traditional IR functions that would enhance the performance of the system greatly. These include truncation, a NOT operator, and more sophisticate relevance ranking methods.

Significant Areas of System

- Brief Description of features in your system which you feel impact the system and are not answered by above questions: We think that our system was at a general disadvantage in this evaluation because much of the query formulation that our system would be expected to excel in appeared to already have been done in the statement of the topic. For instance, the concept fields contained a number of highly diagnostic terms in several cases (e.g., chemotherapy and leukemia for topic 122). On the other hand, our system was not designed to provide relevance rankings. Thus the method that we constructed for expanded the retrieval set to 1000 documents to meet the TREC-3 requirements was extremely ad hoc. Discussions with Chris Buckley and others has convinced us that sophisticated relevance feedback algorithms will tend to be dominant in determining the overall effectiveness of a system for ranked data sets. Thus our strategy was to go with the simplest brute force method we could think of so that our results could

not be attributed to the performance of a particular relevance ranking algorithm or method. One of our goals in entering the TREC-3 system was to show how a browsing style of system could be used in text retrieval. We think that systems like ours would actually look a lot better if evaluations like TREC-3 were modified to consider the specific properties of interactive retrieval. We expect that our system would do a lot better in cases where: the search topic was not precisely stated or known ahead of time, the terminology and vocabulary tended to be inconsistent or imprecise, and where the emphasis was on getting a few good articles that answer specific questions rather than as large a fraction as possible of the set of potentially relevant documents.

System Summary and Timing
Organization Name: Verity Inc.
List of Run ID's: TOPIC1, TOPIC2, TOPIC3, TOPIC4

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list: No content stop words used in building the collections; some reserved words (i.e., SGML tags)removed
- Stemming Algorithm: Verity proprietary, but similar in spirit to Porter
  - Morphological Analysis: Limited to that provided by the stemmer
- Tokenizer? : No special tokens recognized for TREC

Statistics on Data Structures built from TREC Text

- Inverted index
  - Run ID : all runs
  - Total Storage (in MB): 1,504
  - Total Computer Time to Build (in hours): 112 hrs(elapsed time)
  - Automatic Process? (If not, number of manual hours): Yes
  - Use of Term Positions? : Yes
  - Only Single Terms Used? : Yes

Query construction

Interactive Queries

- Initial Query Built Automatically or Manually: Manually for routing test; automatically for ad hoc test
- Type of Person doing Interaction
  - System Expert: Some; we used a team of query builders, some of whom were proficient in the use of TOPIC
- Average Time to do Complete Interaction
  - CPU Time (Total CPU Seconds for all Iterations): No data
  - Clock Time from Initial Construction of Query to Completion of Final Query (in minutes): 116 mins (estimate) averaged across routing and adhoc queries; routing queries on average somewhat lower, adhoc queries on average somewhat higher
- Average Number of Iterations: No data
- Average Number of Documents Examined per Iteration: No data
- Minimum Number of Iterations: No data
- Maximum Number of Iterations: No data
- What Determines the End of an Iteration: the point at which the query is rerun and the results rescored
- Methods used in Interaction
  - Manual Methods
    - Using Individual Judgment (No Set Algorithm)? : Yes

Searching

Search Times

- Run ID : averaged over all runs
- Computer Time to Search (Average per Query, in CPU seconds): no data; average elapsed time is 3 mins (est.) but this varies widely with factors such as query complexity, field search constraints, data subsets being used, network load, etc.

- Component Times : no separate data available, but times are dominated by query matching and scoring

Machine Searching Methods

- Other: Verity's concept-based retrieval technology which uses a mixture of knowledge-based and probabilistic techniques

Factors in Ranking

- Term Frequency? : yes for some operators
- Semantic Closeness? : yes
- Proximity of Terms? : yes for some operators
- Percentage of Query Terms which match? : yes for some operators

Machine Information

- Machine Type for TREC Experiment: SPARC 2 used as file server
- Was the Machine Dedicated or Shared: Dedicated for TREC, but shared by all people doing the query building and experiments
- Amount of Hard Disk Storage (in MB): 4600
- Amount of RAM (in MB): 32MB
- Clock Rate of CPU (in MHz): Unknown

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System: TOPIC is COTS
- Given appropriate resources
  - Could your system run faster? : of course
  - By how much (estimate)? : hard to tell
- Features the System is Missing that would be beneficial: a capability that allowed us to focus on specific "zones" of documents would probably reduce the over generation problem we have

System Summary and Timing
Organization Name: West Publishing Co.
List of Run ID's: westp1 (Ad Hoc) westp2 (Routing)

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list: 290
- Stemming Algorithm: Modified Porter Stemmer
  - Morphological Analysis: Yes. There are 2789 words in our exception list. The remaining words are stemmed using the Porter Algorithm.
- Term Weighting: Yes.
- Phrase Discovery? : Yes.
  - Kind of Phrase: File of common phrases. (approx. 32000).
- Tokenizer? : A small collection of synonyms for company names, countries, states, and months is used.

Statistics on Data Structures built from TREC Text

- Inverted index
  - Total Storage (in MB): westp1 (adhoc) 523
    westp2 (routing) 380
  - Total Computer Time to Build (in hours):
    westp1 (adhoc) 42 hrs.
    westp2 (routing) 16 hrs.
  - Automatic Process? (If not, number of manual hours):
    westp1 (adhoc) yes
    westp2 (routing) yes
  - Use of Term Positions? : westp1 (adhoc) yes
    westp2 (routing) yes
  - Only Single Terms Used? : westp1 (adhoc) yes
    westp2 (routing) yes

Query construction

Automatically Built Queries (Ad-Hoc)

- Topic Fields Used: title, desc, narr
- Average Computer Time to Build Query (in cpu seconds):
  .5 secs per query. (25 seconds for 50 queries.)
- Method used in Query Construction
  - Term Weighting (weights based on terms in topics)? : yes.
  - Phrase Extraction from Topics? : yes.
  - Tokenizer? :
    - Patterns which are Tokenized: some company and country names.

Automatically Built Queries (Routing)

- Topic Fields Used: title, con
- Method used in Query Construction
  - Terms Selected From
    - Topics: yes
  - Term Weighting with Weights Based on terms in
    - Topics: yes. weight of 2 given to terms in title
  - Tokenizer: yes. some acronyms such as "ASK" and "RISC" were recognized.

Searching

Search Times

- Run ID: westp1
- Computer Time to Search (Average per Query, in CPU Seconds):
  Averaged 349 secs/query (clock time).

Machine Searching Methods

- Probabilistic Model? : Yes.

Factors in Ranking

- Term Frequency? : Yes.
- Inverse Document Frequency? : Yes.
- Other Term Weights? : Yes. Differential weights were assigned.

Machine Information

- Machine Type for TREC Experiment: SUN Sparc 5
- Was the Machine Dedicated or Shared: Shared.
- Amount of RAM (in MB): 64
- Clock Rate of CPU (in MHz): 70

System Comparisons

- Amount of "Software Engineering" which went into the
  Development of the System: A couple of years.
- Given appropriate resources
  - Could your system run faster? : Yes.

System Summary and Timing
Organization Name: Bellcore
List of Run ID's: lsir1, lsir2 (routing), laia0mw, lsia0mw20f (adhoc)

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list: yes, 571, SMART's list
- Stemming Algorithm:
  - Morphological Analysis: yes, SMART v11.0, modified lovins
- Term Weighting: yes, SMART's "ltc" for both documents and queries for both adhoc and routing runs
- Other Techniques for building Data Structures: yes
  LSI uses a k-dimensional vector space (k << nterms) for representation and retrieval. LSI begins with a term-document matrix and calculates the best k-dimensional approximation to this matrix using singular value decomposition (SVD). The number of dimensions, k, was about 350 for TREC-3.

Statistics on Data Structures built from TREC Text

- Other Data Structures built from TREC text
  - Run ID : lsir1, lsir2, laia0mw, lsia0mw20f
  - Type of Structure: k-dimensional vector representation (k << nterms); one vector for each term and for each document; k~350 Routing, a 78746 term by 38175 doc sample was used to construct the LSI space. Adhoc, a 82968 term by 69997 doc sample was used to construct the LSI space, and the remaining 672,358 CD-12 documents were folded in.
  - Total Storage (in MB): Adhoc 549 Mb. Routing 109 Mb.
  - Total Computer Time to Build (in hours): Routing, about 24 hours (SVD with k=346).Adhoc, about 20 hours (SVD with k=199) plus 2 hours for folding in 672k docs.
  - Automatic Process? (If not, number of manual hours): yes
  - Brief Description of Method: LSI analysis in brief:
    1. Create term-document matrix, with user-selected term weighting (used SMART v11.0 for this pre-processing).
    2. Compute the best reduced k-dimensional approximation to this matrix using singular value decomposition (SVD). For TREC-3 about 350 dimensions were used in this approximation.
    3. If necessary, fold in any terms or documents not in the original matrix. Necessary for Adhoc runs, not for Routing runs.

Query construction

Automatically Built Queries (Ad-Hoc)

- Topic Fields Used: All
- Average Computer Time to Build Query (in cpu seconds):
  about .1 sec / query (time to take weighted vector sum of query terms)
- Method used in Query Construction
  - Term Weighting (weights based on terms in topics)? : yes

Automatically Built Queries (Routing)

- Topic Fields Used: All (lsir1);
  None (lsir2 - uses only rel documents)
- Average Computer Time to Build Query (in cpu seconds):
  .1-.2 sec/query (time to take weighted vector sum of query terms, lsir1 or relevant documents, lsir2)
- Method used in Query Construction
  - Terms Selected From
    - Topics: yes, lsir1
    - Only Documents with Relevance Judgments: yes, lsir2 - used only relevant documents, ignored irrelevant documents
  - Term Weighting with Weights Based on terms in
    - All Training Documents: yes

Searching

Machine Searching Methods

- Vector Space Model? : Yes, a k-dimensional vector space (k << nterms), but matching does not use an inverted index. The cosine between the k-dim query vector and every k-dim document vector is computed. Adhoc - about 600 sec/query. Routing - about 0.1 sec/query.

Factors in Ranking

- Factors in Ranking
  - Term Frequency? : yes, for constructing term-doc matrix before SVD
  - Inverse Document Frequency? : yes, for constructing term-doc matrix before SVD
  - Semantic Closeness? : yes, the cosine between the query vector and the document vector in LSI-space determines rank
  - Document Length? : yes, for constructing the term-doc matrix before the SVD
  - Other: NOTE: the ranking is determined entirely by the cosine between the query vector and the document vector. Factors like term frequency, inverse document frequency, etc. enter indirectly in the sense that they determine the cell entries for the term-doc matrix which we approximate using the k-dimensional SVD vectors.

Machine Information

- Machine Type for TREC Experiment: Sparc 10
- Was the Machine Dedicated or Shared: Shared
- Amount of Hard Disk Storage (in MB): about 6 Gig
- Amount of RAM (in MB): 128-384 MB, depending on machine
- Clock Rate of CPU (in MHz): don't know

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System: Built as a research prototype - lots of flexibility at the cost of efficiency. About 1-2 person years.
- Given appropriate resources
  - Could your system run faster? : yes

- By how much (estimate)? : A factor of 2-3 would be easy for the pre-processing and SVD (tho these are one time costs). Query matching is slow (linear in the number of documents) and this is trivial to parallelize - improvements of 100 times were obtained using a MasPar
- Features the System is Missing that would be beneficial: Feature recognition (e.g., proper names, company names, geographic locations, dates, amounts of money would be especially useful for TREC); Phrases and other precision enhancing methods like proximity; Training in routing could use information about non-relevant documents; Perhaps handling "nots" would help.

System Summary and Timing
Organization Name: ETH Zurich, Switzerland
List of Run ID's: ETH001, ETH002 (adhoc), ETH003, ETH004 (routing)

Construction of Indices, Knowledge Bases, and other Data Structures 011

Methods Used to build Data Structures

- Length (in words) of the stopword list: 571
- Stemming Algorithm: suffix stripping (Porter, 1980) 011
- Term Weighting: (1+log(ff(phi_i,d_j))) (so-called lnc)
- Phrase Discovery? : yes 011
  - Kind of Phrase: two words without separator (punctuation etc.) in between
  - Method Used (statistical, syntactic, other): weak syntax rules
- Tokenizer? : yes
  - Patterns which are tokenized: words and phrases

Statistics on Data Structures built from TREC Text

- Other Data Structures built from TREC text
  - Run ID : link method
  - Type of Structure: links and files for direct access to ff of each document; link: phrase and a document list where it is
  - Total Storage (in MB): 670
  - Total Computer Time to Build (in hours): ca. 48h
  - Automatic Process? (If not, number of manual hours): yes
  - Brief Description of Method:
    extract phrases from documents;
    invert them;
    cut all phrases that are in less than three documents
      or in more than 7000 (about 1% of all documents;
      phrases as 'UNITED STATES' are cut this way)
      of all documents;
    cut all phrases that have a weight less than a limit c
- Other Data Structures built from TREC text
  - Run ID : passage retrieval method
  - Type of Structure: token files
  - Total Storage (in MB): 1723
  - Total Computer Time to Build (in hours): ca. 14h
  - Automatic Process? (If not, number of manual hours): yes
  - Brief Description of Method:
    extract features from TIPSTER disks,
    delete stopwords

Data Built from Sources Other than the Input Text

- Internally-built Auxiliary File
  - Domain (independent or specific): specific, basic method
  - Type of File (thesaurus, knowledge base, lexicon, etc.): mapping of features to a number
  - Total Storage (in MB): 10
  - Number of Concepts Represented: 500'000
  - Type of Representation: hash table
  - Total Computer Time to Build (in hours): ca. 10
  - Total Manual Time to Build (in hours): no manual time
- Internally-built Auxiliary File
  - Domain (independent or specific): specific, passage retrieval method

- Type of File (thesaurus, knowledge base, lexicon, etc.): HMM
- Total Storage (in MB): 6KB
- Number of Concepts Represented: 2
- Type of Representation: HMM
- Total Computer Time to Build (in hours): 120
- Total Manual Time to Build (in hours): no manual time

Query construction

Automatically Built Queries (Ad-Hoc)

- Topic Fields Used: all
- Average Computer Time to Build Query (in cpu seconds):
    basic method: msec
    link method: 60
    passage retrieval method: msec (same query as basic method)
- Method used in Query Construction
  - Term Weighting (weights based on terms in topics)? :
      (1+log(ff(phi_i,q)))*nidf(phi_i) (so-called ltn)
  - Tokenizer? : yes
    - Patterns which are Tokenized: words and phrases
  - Expansion of Queries using Previously-Constructed Data Structure? : yes
    - Structure Used: links

Automatically Built Queries (Routing)

- Topic Fields Used: all
- Average Computer Time to Build Query (in cpu seconds):
    basic method: msec
    link method: 60
    passage retrieval method: msec
- Method used in Query Construction
  - Terms Selected From
    - Topics: 101-150
    - All Training Documents: yes
  - Term Weighting with Weights Based on terms in
    - Topics: 101-150
    - All Training Documents: yes
  - Tokenizer
    - Patterns which are tokenized
      (dates, phone numbers, common patterns, etc): words and phrases
    - from Topics: 101-150
    - from All Training Documents: yes
  - Expansion of Queries using Previously-Constructed Data Structure:
    - Structure Used: links

Searching

Search Times

- Run ID : ETH001 (adhoc), ETH003 (routing)
- Computer Time to Search (Average per Query, in CPU seconds): msec
- Component Times :
    basic method: msec
    link method: msec

- Run ID : ETH002, ETH004
- Computer Time to Search (Average per Query, in CPU seconds): 600 seconds

- Component Times :
    basic method: msec
    link method: msec
    passage retrieval: 600 seconds

- Machine Searching Methods
  - Vector Space Model? : yes
  - Other: yes - HMM based passage extraction

Factors in Ranking

- Factors in Ranking
  - Term Frequency? : yes
  - Inverse Document Frequency? : yes
  - Position in Document? : yes
  - Proximity of Terms? : yes
  - Document Length? : yes

Machine Information

- Machine Type for TREC Experiment: SPARC MP690
- Was the Machine Dedicated or Shared: shared
- Amount of Hard Disk Storage (in MB): 8000
- Amount of RAM (in MB): 128
- Clock Rate of CPU (in MHz): 40

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System:
    basic method 200h;
    link method 380h;
    passage retrieval 300h

- Given appropriate resources
  - Could your system run faster? : yes
  - By how much (estimate)? : more than 10 times

- Features the System is Missing that would be beneficial:
  effectivity: Rel. feedback, query dependent training (routing)
  efficiency: optimized data structures (they leave less room for experiments)

Significant Areas of System

- Brief Description of features in your system which you feel impact the system and are not answered by above questions: none

System Summary and Timing
Organization Name: Queens College, City University of New York
List of Run ID's: Pircs1,Pircs2 Pircs3,Pircs4

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list: 630
- Stemming Algorithm: Porters algorithm
- Term Weighting: yes
- Phrase Discovery? :
  - Kind of Phrase: 2-word
  - Method Used (statistical, syntactic, other): statistical

Statistics on Data Structures built from TREC Text

- Special Routing Structures
  - Run ID : pircs3, pircs4
  - Type of Structure: Network of linked NODE and EDGE files capturing the query expansion terms and learnt weights
  - Automatic Process? (If not, number of manual hours): yes
  - Brief Description of Method: built from direct files of queries and documents and known relevant document information

- Other Data Structures built from TREC text
  - Run ID : Pircs1,pircs2,pircs3,pircs4
  - Type of Structure: Compressed, truncated direct file; network of linked NODE and EDGE files built during query time
  - Total Storage (in MB): direct file - about 80MB per 500MB raw text; network - about 60MB for 10 queries per 500MB textbase
  - Total Computer Time to Build (in hours): About 5 min per 10 query
  - Automatic Process? (If not, number of manual hours): Yes
  - Brief Description of Method: built from direct files of queries and documents

Data Built from Sources Other than the Input Text

- Internally-built Auxiliary File
  - Domain (independent or specific): independent
  - Type of File (thesaurus, knowledge base, lexicon, etc.): Stop words
  - Total Storage (in MB): .004
  - Number of Concepts Represented: 630
  - Type of Representation: array
  - Total Computer Time to Modify for TREC (if already built): none

- Internally-built Auxiliary File
  - Domain (independent or specific): independent
  - Type of File (thesaurus, knowledge base, lexicon, etc.): 2-word Phrases
  - Total Storage (in MB): .005
  - Number of Concepts Represented: 528
  - Type of Representation: array

Query construction

Automatically Built Queries (Ad-Hoc)

- Topic Fields Used: title, desc, narr; Boolean: title, desc

- Average Computer Time to Build Query (in cpu seconds): 3 sec
- Method used in Query Construction
  - Term Weighting (weights based on terms in topics)? : yes
  - Phrase Extraction from Topics? :yes, 2-word
  - Expansion of Queries using Previously-Constructed Data Structure? : yes
    - Structure Used: Network
  - Automatic Addition of Boolean Connectors or Proximity Operators? : yes

Automatically Built Queries (Routing)

- Topic Fields Used: title, desc, narr, con
- Average Computer Time to Build Query (in cpu seconds): 18
- Method used in Query Construction
  - Terms Selected From
    - Topics: yes
    - Only Documents with Relevance Judgments: yes
  - Term Weighting with Weights Based on terms in
    - Only Documents with Relevance Judgments: yes
  - Phrase Extraction from
    - Topics: yes
    - Only Documents with Relevance Judgments: yes
  - Expansion of Queries using Previously-Constructed Data Structure:
    - Structure Used: Network

Searching

Search Times

- Run ID : Pircs3,Pircs4
- Computer Time to Search (Average per Query, in CPU seconds):about 2-3 min clock time
- Component Times :
  Build network 4 min (per 10 query)
  Retrieval 12 min (per 10 query)
  Sort, merge reformat results 20 min

Machine Searching Methods

- Probabilistic Model? : yes
- Boolean Matching? : yes
- Neural Networks? : yes

Factors in Ranking

- Term Frequency? : yes
- Other Term Weights? : yes, within-doc term frequency, inverse collection term frequency.
- Proximity of Terms? : yes, 2 word phrases
- Document Length? : yes

Machine Information

- Machine Type for TREC Experiment: Sparc 10-30
- Was the Machine Dedicated or Shared: Dedicated
- Amount of Hard Disk Storage (in MB): 6000

- Amount of RAM (in MB): 128

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System: Efficiency improvement were made cutting retrieval time by a factor of 3.
- Given appropriate resources
  - Could your system run faster? :yes
  - By how much (estimate)? : probably half the time.
- Features the System is Missing that would be beneficial: ability to differentiate contexts

System Summary and Timing
Organization Name: Australian National University
List of Run ID's: padre1, padre2

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list: 0
- Term Weighting: Not used in data structure building
- Manually-Indexed Terms? : Removed from collection

Statistics on Data Structures built from TREC Text

- Inverted index
  - Total Storage (in MB): 1248, but note that inverted index was not actually used in submitted runs.
  - Total Computer Time to Build (in hours): 0.025
  - Automatic Process? (If not, number of manual hours): yes
  - Use of Term Positions? : yes
  - Only Single Terms Used? : yes

Query construction

Automatically Built Queries (Ad-Hoc)

- Topic Fields Used: Unknown
- Average Computer Time to Build Query (in cpu seconds): Unknown
- Method used in Query Construction
  - Term Weighting (weights based on terms in topics)? : yes
  - Phrase Extraction from Topics? : yes
  - Automatic Addition of Boolean Connectors or Proximity Operators? : yes

Manually Constructed Queries (Ad-Hoc)

- Topic Fields Used: all
- Average Time to Build Query (in Minutes): 30
- Type of Query Builder
  - Computer System Expert: yes - almost no experience with retrieval systems.
  - Other Lexical Tools? : Dictionary, thesaurus, books, spouse, friend who knew names of American restaurants.
- Method used in Query Construction
  - Term Weighting? : yes, based on perceived importance
  - Boolean Connectors (AND, OR, NOT)? : No
  - Proximity Operators? : Yes
  - Addition of Terms not Included in Topic? : Yes
    - Source of Terms: Dictionary, thesaurus, books, spouse, friend who new names of American restaurants.
  - Other: Set Operator: Union (related to but not the same as boolean OR)

Searching

Search Times
- Run ID : padre1
- Computer Time to Search (Average per Query, in CPU seconds): 38 (elapsed)

Search Times
  - Run ID : padre2
  - Computer Time to Search (Average per Query, in CPU seconds): 42

Machine Searching Methods

- Machine Searching Methods
  - Free Text Scanning? : yes

Factors in Ranking

- Factors in Ranking
  - Term Frequency? : yes
  - Inverse Document Frequency? : yes
  - Other Term Weights? : yes, assigned by searcher or generated
            automatically based on key word and key phrase ranks
  - Proximity of Terms? : yes
  - Document Length? : yes

Machine Information

- Machine Type for TREC Experiment: Fujitsu AP1000
- Was the Machine Dedicated or Shared: Dedicated
- Amount of Hard Disk Storage (in MB): 0 on AP1000, data loaded from
          fileserver over ethernet
- Amount of RAM (in MB): 8192
- Clock Rate of CPU (in MHz): 512 x 25 MHz

System Comparisons

- Amount of "Software Engineering" which went into the
  Development of the System: PADRE was developed from earlier
     (non-TREC-style) software (FTR) in six weeks (elapsed).
     FTR was developed intermittently in the author's spare
     time over 3 years with some help from a vacation student.
     Total effort on PADRE and ancestors: less than half a
     person-year.
- Given appropriate resources
  - Could your system run faster? : yes
  - By how much (estimate)? : factor of 100. PADRE can
       use inverted files but did not to demonstrate the viability
       and advantages of full text scanning on a large parallel machine.
- Features the System is Missing that would be beneficial: Improved
       query generation methods. More expert manual query generator.

System Summary and Timing
Organization Name: GMU / OIT
List of Run ID's: mason1

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list: None
- Stemming Algorithm: No
  - Morphological Analysis: No
- Term Weighting: No, trigram weighting was used instead

Query construction

Automatically Built Queries (Routing)

- Topic Fields Used: all
- Method used in Query Construction
  - Terms Selected From
    - Topics: Yes

Searching

Search Times

- Search Times
  - Run ID : mason1
  - Component Times : 4.5 hours for results to 50 queries

Machine Searching Methods

- Machine Searching Methods
  - Vector Space Model? : Yes
  - N-gram Matching? : Yes
  - Free Text Scanning? : Yes

Factors in Ranking

- Factors in Ranking
  - Other Term Weights? : within document
  - Document Length? : Yes
  - N-gram Frequency? : Yes

Machine Information

- Machine Type for TREC Experiment: Cray-Ymp
- Was the Machine Dedicated or Shared: Shared
- Amount of Hard Disk Storage (in MB): 60 GB
- Amount of RAM (in MB): 64 MB

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System: None
- Given appropriate resources
  - Could your system run faster? : Yes
  - By how much (estimate)? : Several orders of magnitude,

no secondary access structure was used.

- Features the System is Missing that would be beneficial: System at time of TREC did not use a centroid, did not have automatically generated thesaurus. We now have an inverted index and an automatically generated thesaurus.

System Summary and Timing
Organization Name: Logicon Operating Systems
List of Run ID's: losPA1

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list: 171.
- Controlled Vocabulary? : No.
- Stemming Algorithm: No.
- Term Weighting: No.
- Tokenizer? : Yes.
  - Patterns which are tokenized: All alphanumeric strings were tokenized. Only alphabetic tokens of length > 1 were kept. The following were then eliminated: stopwords; tokens serving as SGML tags; tokens occurring only once in the training documents.

Statistics on Data Structures built from TREC Text

- Other Data Structures built from TREC text
  - Run ID : losPA1
  - Type of Structure: Corpus word-frequency table.
  - Total Storage (in MB): 13.5
  - Total Computer Time to Build (in hours): 107
  - Automatic Process? (If not, number of manual hours): Yes.
  - Brief Description of Method: Process analyzed all documents on CDROM #1, and all documents with relevance judgements on CDROM #2. Each document was tokenized, and a count was kept with each token indicating the number of documents in which it occurred.

Query construction

Automatically Built Queries (Routing)

- Topic Fields Used: None.
- Average Computer Time to Build Query (in cpu seconds): CPU time was not measured. Average wall-clock time was 366 seconds.
- Method used in Query Construction
  - Terms Selected From
    - Only Documents with Relevance Judgments: Yes. Terms were selected only from relevant training documents.
  - Term Weighting with Weights Based on terms in
    - All Training Documents: Yes.
    - Only Documents with Relevance Judgments: Yes.
  - Tokenizer
    - from All Training Documents: Yes. See Data Structures.
    - from Documents with Relevance Judgments: Yes. Same as tokenizing rules for Data Structures.
  - Automatic Addition of Boolean connectors or Proximity Operators using information from
    - All Training Documents: Yes.
    - Only Documents with Relevance Judgments: Yes. The ten "most descriptive" terms were OR'ed for selection. Additional terms were then used for ranking.

Searching

Search Times

- Run ID : losPA1
- Computer Time to Search (Average per Query, in CPU seconds):
  CPU time was not measured. Average wall-clock time was 0.99 seconds per document.

Machine Searching Methods

- Free Text Scanning? : Yes. Search engine was the Logicon Message Dissemination System (LMDS), a COTS product.

Factors in Ranking

- Term Frequency? : Yes. Term frequency in the set of relevant training documents vs. the set of all training documents.
- Document Length? : Yes. Score from term weights was multiplied by the ratio of unique query tokens in document to total unique tokens in document.

Machine Information

- Machine Type for TREC Experiment: Sun SPARCstation IPC.
- Was the Machine Dedicated or Shared: Dedicated.
- Amount of Hard Disk Storage (in MB): 4,000 MB.
- Amount of RAM (in MB): 24 MB.
- Clock Rate of CPU (in MHz): 25 MHz.

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System: The search engine was the Logicon Message Dissemination System (LMDS), a COTS product. The query creation software and document ranking software were research prototypes integrated with LMDS via its API.

- Given appropriate resources
  - Could your system run faster? : Yes.
  - By how much (estimate)? : 50 to 80 percent.

- Features the System is Missing that would be beneficial:
  A word stemmer; AND logic in the query; an improved method for determining the optimum number of terms to use in ranking.

System Summary and Timing
Organization Name: Mayo
List of Run ID's: expst1 expst2

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list: 571
- Manually-Indexed Terms? : CO, IN and GV are used; the given abbreviation of each concept is used as the ID of the concept.

Statistics on Data Structures built from TREC Text

- Inverted index
  - Run ID : expst1
  - Total Storage (in MB): 123 MB
  - Total Computer Time to Build (in hours): 22 CPU hours
  - Automatic Process? (If not, number of manual hours): Yes
  - Only Single Terms Used? : No. The concepts in the CO, IN and GV fields are used, too.

- Special Routing Structures
  - Run ID : expst1
  - Type of Structure: sparse matrices, similar to inverted-file document indexing
  - Total Storage (in MB): 8 MB (plus 72 MB if counting the WSJ documents which are indexed for the AdHoc Test anyway)
  - Total Computer Time to Build (in hours): 0.13 CPU hours (plus 60 hours if counting the indexing time of the WSJ documents which is needed for the AdHoc Test anyway)
  - Automatic Process? (If not, number of manual hours): Yes
  - Brief Description of Method: We use Expert Network (ExpNet) to expand both routing queries and ad-hoc queries. Given a query, ExpNet identifies its Nearest Neighbors (NNs) among training queries, and then the terms in the documents which are related to the NNs are added to the given query. For the routing task, we set the number of NNs to one. That is, ExpNet behaves just like relevance feedback, and adds to each query the top-ranking terms (1000) in related documents of this query. A weighting factor of 0.45 was experimentally determined to scale the additional terms in the query expansion.

Statistics on Data Structures built from TREC Text

- Inverted index
  - Run ID : expst2
  - Total Storage (in MB): 72 MB
  - Total Computer Time to Build (in hours): 60 CPU hours
  - Automatic Process? (If not, number of manual hours): Yes
  - Only Single Terms Used? : No. The concepts in the CO, IN and GV fields are used, too.

- Special Routing Structures
  - Run ID : expst2
  - Type of Structure: sparse matrices, equivalent to inverted-file document indexing
  - Total Storage (in MB): 14 MB (plus 72 MB if counting the index

of the WSJ documents which are needed anyway)
- Total Computer Time to Build (in hours): 0.3 CPU hour (plus 60 hours if counting the indexing time of the WSJ documents which are needed anyway)
- Automatic Process? (If not, number of manual hours): Yes
- Brief Description of Method: We use Expert Network (ExpNet) to expand both routing queries and ad-hoc queries. Given a query, ExpNet identifies its Nearest Neighbors (NNs) among training queries, and then the terms in the documents which are related to the NNs are added to the given query. For the ad-hoc task, we set the number of NNs to 13. That is, ExpNet chooses 13 NNs (training queries) for each testing query, and adds 1000 top-ranking terms in the related documents to the given query. A weighting factor of 0.10 was experimentally determined to scale the additional terms in the query expansion.

Query construction

Automatically Built Queries (Ad-Hoc)

- Topic Fields Used: title, desc, narr, con
- Average Computer Time to Build Query (in cpu seconds): 871/50
- Method used in Query Construction
  - Term Weighting (weights based on terms in topics)? : Yes. TF*IDF is used, where TF is the with-in-topic-frequency of a term, and the IDF is derived from a document collection (WSJ documents in disk 1 and 2, and SJM documents in disc 3)
  - Heuristic Associations to Add Terms? : Yes, a query-query similarity (cosine value) is used to identify the NNs (13 training queries) of a given query, and the query-document relevance judgments of these NNs are used to expand the given query.

Automatically Built Queries (Routing)

- Topic Fields Used: title, desc, narr, con
- Average Computer Time to Build Query (in cpu seconds): 524/50
- Method used in Query Construction
  - Terms Selected From
    - Topics: yes
    - Only Documents with Relevance Judgments: yes
  - Term Weighting with Weights Based on terms in
    - Topics: yes
    - Only Documents with Relevance Judgments: yes
  - Heuristic Associations to Add Terms from
    - Topics: yes
    - Only Documents with Relevance Judgments: yes
  - Expansion of Queries using Previously-Constructed Data Structure:
    - Structure Used: sparse matrices (equivalent to inverted files, refer to the report for details).

Searching

Search Times

- Run ID : expst1

- Computer Time to Search (Average per Query, in CPU seconds): 438/50

- Run ID : expst2
- Computer Time to Search (Average per Query, in CPU seconds): 1290/50

Machine Searching Methods

- Run ID : expst1
- Vector Space Model? : yes
- Other: Expert Network (a Nearest Neighbor classification approach) is used in combination with a Vector Space Model.

- Run ID : expst2
- Vector Space Model? : yes
- Other: Expert Network (a Nearest Neighbor classification approach) is used in combination with a Vector Space Model.

Factors in Ranking

- Run ID : expst1
- Term Frequency? : yes
- Inverse Document Frequency? : yes

- Run ID : expst2
- Term Frequency? : yes
- Inverse Document Frequency? : yes

Machine Information

- Machine Type for TREC Experiment: SparcStation 10/30
- Was the Machine Dedicated or Shared: Shared (was also researcher's desktop)
- Amount of Hard Disk Storage (in MB): 3059 MB
- Amount of RAM (in MB): 32 MB
- Clock Rate of CPU (in MHz): 36 MHz

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System: our ExpNet is an research-oriented system; about 4 person-month has been spent on adopting it to the TREC materials and its scale-up

- Given appropriate resources
  - Could your system run faster? : yes

- Features the System is Missing that would be beneficial: simplification of data structure and system

System Summary and Timing
Organization Name: National Security Agency
List of Run ID's: ACQNT2 (= routing), ACQNT1(= adhoc)

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list: 0
- Term Weighting: yes
- Other Techniques for building Data Structures: uses n-grams to build vectors that characterize each document by the frequency of occurrence of those n-grams

Statistics on Data Structures built from TREC Text

- Inverted index
  - Run ID : ACQNT1
  - Total Computer Time to Build (in hours): approx 0.2
  - Automatic Process? (If not, number of manual hours): yes
  - Use of Term Positions? : no
  - Only Single Terms Used? : yes
- Inverted index
  - Run ID : ACQNT2
  - Total Computer Time to Build (in hours): approx 0.2
  - Automatic Process? (If not, number of manual hours): yes
  - Use of Term Positions? : no
  - Only Single Terms Used? : yes
- N-grams, Suffix arrays, Signature Files
  - Run ID : ACQNT1
  - Total Storage (in MB): generally less than 10
  - Total Computer Time to Build (in hours): less than 2
  - Automatic Process? (If not, number of manual hours): yes
  - Brief Description of Method: The n-grams in each document were tallied.
- N-grams, Suffix arrays, Signature Files
  - Run ID : ACQNT2
  - Total Storage (in MB): generally less than 10
  - Total Computer Time to Build (in hours): less than 2
  - Automatic Process? (If not, number of manual hours): yes
  - Brief Description of Method: The n-grams in each document were tallied.

Query construction

Automatically Built Queries (Ad-Hoc)

- Topic Fields Used: title, description, narrative
- Average Computer Time to Build Query (in cpu seconds): less than one
- Method used in Query Construction
  - Term Weighting (weights based on terms in topics)? : yes, where terms are n-grams in topics

Automatically Built Queries (Routing)

- Average Computer Time to Build Query (in cpu seconds): less than one

- Method used in Query Construction
  - Terms Selected From
    - Only Documents with Relevance Judgments: yes, from relevant documents
  - Term Weighting with Weights Based on terms in
    - Only Documents with Relevance Judgments: All the documents judged relevant to a particular topic were used to build n-gram-based query vectors

Searching

Search Times

- Run ID : ACQNT1
- Computer Time to Search (Average per Query, in CPU seconds): approx 2.5 secs per 100 MB of text

- Run ID : ACQNT2
- Computer Time to Search (Average per Query, in CPU seconds): approx. 4.8 secs per 100 MB of text

Machine Searching Methods

- Vector Space Model? : yes
- Probabilistic Model? : yes
- N-gram Matching? : yes

Factors in Ranking

- Term Frequency? : yes, where terms are n-grams
- Other Term Weights? : yes
- N-gram Frequency? : yes
- Other: n-gram frequencies are offset according to population means

Machine Information

- Machine Type for TREC Experiment: CRAY C 90
- Was the Machine Dedicated or Shared: shared
- Amount of Hard Disk Storage (in MB): 4,000
- Amount of RAM (in MB): 128
- Clock Rate of CPU (in MHz): 250

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System: one-half person year

- Given appropriate resources
  - Could your system run faster? : yes
  - By how much (estimate)? : with software improvements, 1 order of magnitude, with hardware implementation, perhaps 3 orders of magnitude

System Summary and Timing
Organization Name: NEC
List of Run ID's: virtu1, virtu2

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list: 430
- Stemming Algorithm: Yes
  - Morphological Analysis: Yes
- Term Weighting: Yes

Statistics on Data Structures built from TREC Text

- Inverted index : Yes
  - Run ID : virtu1, virtu2
  - Total Storage (in MB): 2056
  - Total Computer Time to Build (in hours): 214
  - Automatic Process? (If not, number of manual hours): Yes
  - Use of Term Positions? : Yes
  - Only Single Terms Used? : Yes

Data Built from Sources Other than the Input Text

- Internally-built Auxiliary File : Yes
  - Domain (independent or specific): independent
  - Type of File (thesaurus, knowledge base, lexicon, etc.):
    lexicon, inflection table
  - Total Storage (in MB): 4
  - Number of Concepts Represented: 143360
  - Type of Representation: look up table
  - Total Computer Time to Build (in hours): Not recorded
  - Total Computer Time to Modify for TREC (if already built): 0.5

Query construction

Automatically Built Queries (Ad-Hoc) : Yes

- Topic Fields Used: All
- Average Computer Time to Build Query (in cpu seconds): > 10 sec per query
- Method used in Query Construction
  - Term Weighting (weights based on terms in topics)? : Yes
  - Phrase Extraction from Topics? : Yes
  - Syntactic Parsing of Topics? : Yes
  - Proper Noun Identification Algorithm? : Yes
  - Tokenizer? : Yes
    - Patterns which are Tokenized: part of noun phrase identification
  - Automatic Addition of Boolean Connectors or Proximity Operators? : Yes

Manually Constructed Queries (Routing) : Yes

- Topic Fields Used: All
- Average Time to Build Query (in Minutes): 30 min per query
- Type of Query Builder
  - Domain Expert: No
  - Computer System Expert: Yes
- Data Used for Building Query from

- Training Topics: Yes
- All Training Documents: Yes
- Only Documents with Relevance Judgments: Yes

- Method used in Query Construction
  - Term Weighting? : Yes
  - Boolean Connectors (AND, OR, NOT)? : Yes
  - Proximity Operators? : Adjacency operator used
  - Addition of Terms not Included in Topic? : Yes
    - Source of Terms: relevant document

Searching

Search Times

- Run ID : virtu1, virtu2
- Computer Time to Search (Average per Query, in CPU seconds): 1200

Machine Searching Methods

- Vector Space Model? : Yes
- Probabilistic Model? : Yes
- Boolean Matching? : Yes partially

Factors in Ranking

- Factors in Ranking
  - Term Frequency? : Yes
  - Inverse Document Frequency? : Yes
  - Proximity of Terms? : Adjacency used
  - Document Length? : Yes

Machine Information

- Machine Type for TREC Experiment: sparc10
- Was the Machine Dedicated or Shared: shared
- Amount of Hard Disk Storage (in MB): 8000
- Amount of RAM (in MB): 90
- Clock Rate of CPU (in MHz): 40

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System: It was a first prototype. Three person month were spent.
- Given appropriate resources
  - Could your system run faster? : Yes
  - By how much (estimate)? : Main part of CPU time is disk I/O. If separate inverted files are used, 20-40% of CPU time may be decreased.
- Features the System is Missing that would be beneficial: Document Clustering, General Proximity Operation

Significant Areas of System

- Brief Description of features in your system which you feel impact the system and are not answered by above questions:

  Morphological analysis on making inverted files, Noun

phrase detection on automatic query Construction (ad-hoc),
Adjacent word detection

System Summary and Timing
Organization Name: Rutgers/APLab-Data Fusion
List of Run ID's: rutfu[r,a][1,2]

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list : As Inquery 2.0
- Controlled Vocabulary? : As Inquery 2.0
- Stemming Algorithm: As Inquery 2.0
  - Morphological Analysis: As Inquery 2.0
- Term Weighting: As Inquery 2.0

Manually Constructed Queries (Ad-Hoc)

- Topic Fields Used: all
- Average Time to Build Query (in Minutes): 3
- Type of Query Builder
  - Domain Expert: No. Simple entry of content words from topic, assembled into a Boolean AND of (ORs) structure.
- Method used in Query Construction
  - Boolean Connectors (AND, OR, NOT)? : Yes. AND OR

Manually Constructed Queries (Routing)

- Topic Fields Used: All
- Average Time to Build Query (in Minutes): 3
- Type of Query Builder
  - Domain Expert: No. Simple entry of content words from topic, assembled into a Boolean AND of (ORs) structure.
- Method used in Query Construction
  - Boolean Connectors (AND, OR, NOT)? : AND OR

Searching

Search Times
- Computer Time to Search (Average per Query, in CPU seconds): As for Inquery. But multiplied by 3 because 3 runs are combined in each fusion. The fusion program takes less than 1 minute to run per topic.

Machine Information

- Machine Type for TREC Experiment: For basic retrieval, as for the Inquery group. For Fusion. Sparc 10
- Was the Machine Dedicated or Shared: Shared

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System: For the fusion: Two person-weeks
- Given appropriate resources
  - Could your system run faster? : Yes
  - By how much (estimate)? : Factor 5 on fusion

System Summary and Timing
Organization Name: University of Minnesota
List of Run ID's: TeknosN05, TeknosN10, TeknosN15, TeknosN20,
TeknosN30, TeknosRel, TeknosW05, TeknosW10,
TeknosW15, TeknosW20, TeknosW30

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list: 213
- Stemming Algorithm:
  - Morphological Analysis: Personal Librarian default
- Term Weighting: Personal Librarian default

Statistics on Data Structures built from TREC Text

- Inverted index
  - Run ID : All Teknos runs. Load information for Wall Street Journal Disk 2. This is NOT the data set load used to produce the TREC results but is the only data set load that we were able to measure.
  - Total Storage (in MB): 328,904,027 bytes:
    222,485,339 data,
    106,418,688 for index

  - Total Computer Time to Build (in hours): 10 hours 16 minutes for Personal Librarian indexing.
  - Automatic Process? (If not, number of manual hours): Yes

- Knowledge Bases
  - Run ID : All Teknos runs
  - Total Storage (in MB): 1.31 MB
  - Automatic Process? (If not, number of manual hours): Manual. Did not measure time. Estimated 40 hours.
  - Use of Manual Labor
    - Mostly Manually Built using Special Interface: yes
  - Number of Concepts Represented: 10 pertaining to this experiment
  - Type of Representation: semantic network

Data Built from Sources Other than the Input Text

Query construction

Manually Constructed Queries (Ad-Hoc)

- Topic Fields Used: Description, narrative
- Method used in Query Construction
  - Proximity Operators? : yes
  - Addition of Terms not Included in Topic? :
    - Source of Terms: test data set (WSJ Disk 1)

Manually Constructed Queries (Routing)

- Average Time to Build Query (in Minutes): 8 hours (very rough estimate)
- Data Used for Building Query from
  - All Training Documents: yes

- Method used in Query Construction
  - Boolean Connectors (AND, OR, NOT)? : yes
  - Proximity Operators? : yes
  - Addition of Terms not Included in Topic? :
    - Source of Terms: training documents

Searching

Search Times

- Run ID : TeknosRel
- Computer Time to Search (Average per Query, in CPU seconds): 30

Machine Information

- Machine Type for TREC Experiment: Gateway 2000 66 MHz 486DX2 running Windows 3.11
- Was the Machine Dedicated or Shared: Dedicated
- Amount of Hard Disk Storage (in MB): 2 drives: 424 MB, 540 MB (real) compressed using Stacker 4.0
- Amount of RAM (in MB): 12 MB
- Clock Rate of CPU (in MHz): 66 MHz

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System: Indexing system is commercial. System to maintain knowledge base and expand conceptual graphs to boolean search expressions: 200+ hrs. (it evolved from a different project - this is the time I estimate it would have taken to develop from scratch).
- Given appropriate resources
  - Could your system run faster? : yes
  - By how much (estimate)? : unknown

System Summary and Timing
Organization Name: Paracel
List of Run ID's: FDF1 FDF2

Construction of Indices, Knowledge Bases, and other Data Structures

Statistics on Data Structures built from TREC Text

- Other Data Structures built from TREC text
  - Run ID : FDF1 FDF2
  - Type of Structure: word and phrase statistics table
  - Total Storage (in MB): 100 MB (uncompressed)
  - Total Computer Time to Build (in hours): (unrecorded)
  - Automatic Process? (If not, number of manual hours): yes
  - Brief Description of Method:

    Collection of in-document frequency, frequency and variance statistics for all corpus words and for all relevant document phrases. This process is only applied to the training text. For queries built one-at-a-time this process need take no additional storage.

Query construction

Automatically Built Queries (Routing)

- Average Computer Time to Build Query (in cpu seconds): ~200
- Method used in Query Construction
  - Terms Selected From
    - All Training Documents: yes
    - Only Documents with Relevance Judgments: yes
  - Term Weighting with Weights Based on terms in
    - All Training Documents: yes
    - Only Documents with Relevance Judgments: yes
  - Phrase Extraction from
    - Only Documents with Relevance Judgments: yes

Searching

Search Times

- Run ID : FDF1
- Computer Time to Search (Average per Query, in CPU seconds):
  1462.5 (host CPU)
  4547.9 (real)
- Component Times : As above. (The search engine takes queries and text and spits out scored hits. Collection and sorting time is negligible.)

- Run ID : FDF2
- Computer Time to Search (Average per Query, in CPU seconds):
  1068.7 (host CPU)
  3097.9 (real)
- Component Times : As above. (The search engine takes queries and text and spits out scored hits. Collection and sorting time is negligible.)

Machine Searching Methods

- Probabilistic Model? : yes
- Free Text Scanning? : yes, FDF-3 search engine

Factors in Ranking

- Term Frequency? : yes
- Inverse Document Frequency? : yes
- Other: term frequency variance, term recall, estimated query method performance

Machine Information

- Machine Type for TREC Experiment:
  Host: at various times, Sun SPARC-2, SPARC ELC, HP 9000/715
  Peripheral: 4 * FDF-3/15 w/ Seagate Elite-3 and Elite-9 disks

- Was the Machine Dedicated or Shared: shared
- Amount of Hard Disk Storage (in MB): 3 GB (actual max usage, uncompressed data from all CD-ROMs) 22 GB (max in configuration)
- Amount of RAM (in MB): 16 MB (32 MB swap)
- Clock Rate of CPU (in MHz): Hosts: various
  FDF-3: 12.5 MHz

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System: 0.25 person-year (on top of the underlying FDF-3 system)
- Given appropriate resources
  - Could your system run faster? : yes
- Features the System is Missing that would be beneficial:
  This is very difficult to answer. The system is currently under development. We believe that the underlying search engine has all the necessary facilities to perform the desired retrieval functions; we're currently working to make those facilities available to the user in appropriate form.

Significant Areas of System

- Brief Description of features in your system which you feel impact the system and are not answered by above questions:

  (a) ability to quickly re-run statistics collection.
  (b) easy reconfiguration of workbench.
  (c) automatic query selection (query-level data fusion) which also provides low-level performance data (and hence detailed performance comparisons).

General Comments:

There isn't place to put this in the form: we used a stop-word list for term suppression (173 words). This isn't used for searching, however, just for query construction. Also, we did statistical

phrase discovery -- but again, this was for query construction, not searching. CPU time is nearly meaningless. REAL TIME! Should include CPU type (which is listed elsewhere), types of disks involved, etc. Our "FDF1" run used >4500 sec REAL time, but <1500 sec CPU time, so REAL>3*CPU! This is crucial: the user wants to know how long it took to get the answer, not how long the CPU spent on the problem.011

System Summary and Timing
Organization Name: University of Neuchatel (Switzerland)
List of Run ID's: UniNE1, UniNE2

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Length (in words) of the stopword list: 571 (SMART)
- Stemming Algorithm: yes Lovins (SMART)
- Term Weighting: yes lnc for doc; ltc for query
- Other Techniques for building Data Structures: yes

Statistics on Data Structures built from TREC Text

- Inverted index
  - Run ID : UniNE1, UniNE2
  - Total Storage (in MB): 178.9
  - Total Computer Time to Build (in hours): 2.6
  - Automatic Process? (If not, number of manual hours): yes
  - Use of Term Positions? : no
  - Only Single Terms Used? : yes

- Other Data Structures built from TREC text
  - Run ID : UniNE2 only
  - Type of Structure: relevance judgments for query #1 to #150
  - Total Storage (in MB): 3.7 MB
  - Total Computer Time to Build (in hours): 0.12
  - Automatic Process? (If not, number of manual hours): yes
  - Brief Description of Method: for each document, we store the identifier of all documents found relevant in a previous request

Query construction

Automatically Built Queries (Ad-Hoc)

- Topic Fields Used: all
- Average Computer Time to Build Query (in cpu seconds): 0.5
- Method used in Query Construction
  - Term Weighting (weights based on terms in topics)? : yes (ltc SMART)

Searching

Search Times

- Run ID : UniNE1, UniNE2
- Computer Time to Search (Average per Query, in CPU seconds): 16.1 sec. for UniNE1, 19.3 sec. for UniNE2

Machine Searching Methods

- Vector Space Model? : yes (SMART)

Factors in Ranking

- Term Frequency? : yes
- Inverse Document Frequency? : yes
- Other Term Weights? : yes

- Other: yes

Machine Information

- Machine Type for TREC Experiment: SUN SPARCstation 10 model 51
- Was the Machine Dedicated or Shared: shared
- Amount of Hard Disk Storage (in MB): 6,144 MB
- Amount of RAM (in MB): 128 MB
- Clock Rate of CPU (in MHz): 50 MHz

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System: 4 months to understand SMART, 3 months to write our additional features in Smalltalk-80
- Given appropriate resources
  - Could your system run faster? : yes, because Smalltalk is interpreted
  - By how much (estimate)? : the improvement factor is unknown

System Summary and Timing
Organization Name: University of Central Florida
List of Run ID's: UCF101 and UCFSJM (each is a routing run).

Construction of Indices, Knowledge Bases, and other Data Structures

Methods Used to build Data Structures

- Controlled Vocabulary? : Yes.
- Stemming Algorithm: None
- Phrase Discovery? : Yes, but it is a manual process when training a filter.

- Kind of Phrase: Any kind of phrase (a sequence of words) that could be useful. For example, "chief executive officer", "due to", "back to committee", "plan that would insure Americans", and "shut down trading".
- Method Used (statistical, syntactic, other): Manual observation of viewed training text.
- Manually-Indexed Terms? : Each topic has its own knowledge base which is derived from an Entity Relationship (ER) schema for the topic. For each topic, the knowledge base primarily takes the form of one or more lists (files). There are two types of files. There is a synonym file for each structure component of the ER schema, and there is a domain file for each attribute specified in the ER schema. A phrase (a sequence of words) can be an entry in a domain or synonym file. Different forms of an entry (such as carry, carries, carried,...) are also put in these files. These files are initially built from information found in a dictionary, a thesaurus, or any specialized reference source. Training text is manually viewed to make modifications to these files. The knowledge base for a topic also includes another information (INF) file. The INF file specifies the size of a window for evaluating text, along with the importance of the individual domain and synonym files for determining relevancy of the text in the window.

Statistics on Data Structures built from TREC Text

- Knowledge Bases
  - Run ID : UCF101 and UCFSJM (each is a routing run).
  - Use of Manual Labor
- Special Routing Structures
  - Run ID : UCF101 and UCFSJM (each is a routing run).
  - Type of Structure: Hash table in memory to store entries in the synonym and domain files of a particular topic filter before beginning input text scan.
  - Total Storage (in MB): Insignificant.
  - Total Computer Time to Build (in hours): A few seconds.
  - Automatic Process? (If not, number of manual hours): Yes.
  - Brief Description of Method: Before a filter starts scanning input text documents, its synonym and domain files are read and each entry is placed in a memory resident hash table.
- Other Data Structures built from TREC text

Data Built from Sources Other than the Input Text

- Internally-built Auxiliary File
  - Domain (independent or specific): Domain specific, a set of synonym and domain files are built for each topic.
  - Type of File (thesaurus, knowledge base, lexicon, etc.): Each file is a list of words or phrases (a sequence of words). A synonym file is constructed for a component of an ER schema. A domain file is constructed for an attribute

in an ER schema. Alternate forms of words or phrases are also placed in these files.

- Total Storage (in MB): For the fifty routing topics, total storage for the synonym and domain files was 214K.
- Number of Concepts Represented: The concepts represented by a filter's synonym and domain files are ER schema entities, attributes, relationships, roles, subset predicates, specializations, generalizations, and categories.
- Type of Representation: An Entity Relationship (ER) Schema for a topic.

- Total Manual Time to Build (in hours): Manually building the synonym and domain files for a single topic ranged from one hour to sixteen hours, the average time was ten hours. However, we did not have enough time to build the best possible files.
- Use of Manual Labor
  - Mostly Manually Built using Special Interface: The files were manually built using only an editor. Initially, the files are established using reference material such as a dictionary, a thesaurus, or any specific reference book. The files were later modified after viewing training text. A few special interfaces were used during the training process.

- Internally-built Auxiliary File
- Domain (independent or specific): Domain specific, an information (INF) file. One is built for each topic.
- Type of File (thesaurus, knowledge base, lexicon, etc.): The INF file specifies the insertion criteria for a topic's ER schema.
- Total Storage (in MB): Small, about 100 bytes each.
- Number of Concepts Represented: The INF file specifies the size of a sliding window (the number of words) used to determine membership in specified combinations of synonym and domain files. The importance of each synonym and domain file is also indicated in the INF file.
- Type of Representation: The insertion criteria for an Entity Relationship schema.
- Total Manual Time to Build (in hours): A few seconds to establish one, but an hour or so of wait time to see how good the INF file was for the filter. We did not have enough time to build the best possible INF file for each of the filters.
- Use of Manual Labor
  - Mostly Manually Built using Special Interface: An INF file is manually built using only an editor. An INF file is usually modified after viewing training text. In an INF file, the window size and weights of individual synonym and domain files were also modified by observing successive performance evaluations over training text. This was done (when we had time) to obtain optimum performance of a filter over the training text. We did not have enough time to build optimum filters.

- Externally-built Auxiliary File

Manually Constructed Queries (Routing)

- Topic Fields Used: All.
- Average Time to Build Query (in Minutes): This is the time to sketch an ER schema for a topic (typically, one hour) plus the time to build synonym and domain files for the schema (average is ten hours) plus a minute or so to create the INF file for the topic.
- Type of Query Builder
  - Computer System Expert: The person constructing the synonym and domain files for a topic was an undergraduate student in a Computer Science database course.
- Data Used for Building Query from
  - Only Documents with Relevance Judgments: Yes, if we had time.
  - Other Sources: Hardcopy references (such as a dictionary, a thesaurus,

or a specialized reference book) were used. During training, some documents were retrieved that had no definite relevance judgment, so these documents were read and used if the student felt they were relevant.

- Method used in Query Construction
  - Term Weighting? : A weight can be assigned to each synonym file and each domain file of a filter.
  - Boolean Connectors (AND, OR, NOT)? : A form of AND and NOT is used when a combination of synonym and domain files is specified. A form of OR is used when different combinations of synonym and domain files are listed.
  - Proximity Operators? : The sliding window (number of words) to evaluate relevancy.
  - Addition of Terms not Included in Topic? : Yes!
    - Source of Terms: Any kind of reference material and viewed training text.

Searching

Search Times

- Search Times
  - Run ID : UCF101 and UCFSJM (each is a routing run).
  - Computer Time to Search (Average per Query, in CPU seconds): Since each routing query was a true filter that scanned across the entire document collection, we kept track of wall clock time. For all of Vol. 3, the fastest time was 1 hour and 23 minutes, while the slowest time was 2 hours and 45 minutes. For the SJMN directory, the fastest time was 23 minutes and the slowest time was 28 minutes. The difference in times was determined by the number of filters running on the network at the same time. This ranged from one filter to about eight filters running simultaneously when producing UCF101 and UCFSJM.
  - Component Times : No component times, just run the filter across the document collection.

Machine Searching Methods

- Machine Searching Methods
  - Boolean Matching? : Somewhat.
  - Free Text Scanning? : Yes.
  - Other: A window (number of words) to view was moved across a document collection and the window was evaluated in regard to words that satisfied the insertion criteria for an Entity Relationship (ER) schema of a topic description. This could be Conceptual Graph Matching.

Factors in Ranking

- Factors in Ranking
  - Term Frequency? : Yes.
  - Other Term Weights? : Yes. Each synonym or domain file can be assigned an integer "importance" determined by optimum performance over training text. We did not have enough time to determine optimum numbers.
  - Position in Document? : Yes, sliding window of words to evaluate.
  - Proximity of Terms? : Yes, sliding window of words to evaluate.
  - Other: 1. Number of synonym and domain files for a filter.
    2. Local evaluations (in the window) and a global evaluation of the entire document are used.
    3. Multiple combinations of synonym and domain files are allowed for a filter.

Machine Information

- Machine Type for TREC Experiment:
  For training:
  1. The Vol. 1 CD was copied to the hard drive of a PC running Linux (a public domain version of Unix) and functioning as an NFS node.
  2. The Vol. 2 CD was copied to the hard drive of a SPARCserver 690MP (4 processors).
  3. Students ran filters and viewed training text from 32 RISC 6000 machines across a network.

  For UCF101 and UCFSJM runs:
  1. The Vol. 3 CD was copied to the hard drive of a SPARCserver 690MP (4 processors).
  2. Most filters were run on the SPARCserver 690MP (a few ran on the RISC 6000 machines).
- Was the Machine Dedicated or Shared: Shared, except for the NFS node running Linux.
- Amount of Hard Disk Storage (in MB): We had access to 1000 MB on the NFS node, and 1000 MB on the SPARCserver.
- Amount of RAM (in MB): 8 MB on each of the 32 RISC 6000 machines.
16 MB on the NFS node.
128 MB on the SPARCserver.
- Clock Rate of CPU (in MHz): 33 MHz for the NFS node. Not known for the RISC 6000 machines. Not known for the SPARCserver 690MP.

System Comparisons

- Amount of "Software Engineering" which went into the Development of the System:
80 hours: Purchase and install hardware and establish network access.
160 hours: Design, code, and test the basic filter scanner.
40 hours: Design, code, and test a few utilities for training.
40 hours: Figure out the rules for drawing an "atomic" ER diagram.
100 hours: Establish an ER schema diagram for each of the routing topics.
500 hours: Establish synonym and domain files for the routing topics for the two submitted runs.
- Given appropriate resources
  - Could your system run faster? : Yes.
  - By how much (estimate)? : It is a function of how many machines are available for running a filter, and how much traffic the network will tolerate. It might be possible to put a filter on each processor of a machine like the MASPAR, and in four iterations, filter the documents on Vol. 3 in about four minutes.
- Features the System is Missing that would be beneficial:
1. A human-computer dialog interface to automate the development of an ER "atomic" schema from a person with a search request.
2. Access to electronic dictionaries, thesauri, and reference material for initial filter construction from the ER schema.
3. Utility programs to help train the filters using training documents and relevancy judgments.
4. An interface for filter modification during interactive queries.

Significant Areas of System

- Brief Description of features in your system which you feel impact the system and are not answered by above questions:

APPENDIX C

COMPARISON BETWEEN TREC2 and TREC3

Karen Sparck Jones, November 15, 1994

The following table presents Precision performance at Document Cutoff 30 for both Trec 2 and Trec 3. The Trec 2 figures are taken from KSJ, 'Reflections on TREC' (prepared for the issue of Information Processing and Management on TREC, also published as Computer Laboratory TR 347, 1994). The Trec 3 figures are from the Trec 3 conference working papers. The data are only for full runs, not Category B.

The conventions are as for Trec 2: figures are not rounded, performance is assigned to 'blocks', teams per block are in Proceedings order, the best of two official runs is taken where there are two, regardless of the retrieval method used. However I have indicated the Trec 3 interactive figures separately, as a different evaluation methodology was applied.

DOCUMENTS 30

	AD HOC		ROUTING		INTERACTIVE (ROUTING)
	Trec 2	Trec 3	Trec 2	Trec 3	Trec 3
>= 60		Amherst City Berkeley	Cornell Dortmund	City	
>= 55	Amherst HNC VT	Cornell Mead	City Berkeley Amherst Bellcore CMU	Amherst Cornell Berkeley Dortmund Bellcore CUNY	
>= 50	Cornell Berkeley Dortmund CMU Verity Siemens CUNY	VT Westlaw ETH CUNY	Rutgers HNC GE TRW Verity Siemens	CMU Westlaw Logicon TRW Florida	
>= 45	City Bellcore ETH CITRI Conquest	NYU CMU RMIT RutgersK	VT	Xerox NYU Verity ETH NSA NEC	RutgersB Verity
>= 40					City

Notes:

1) Changed players - Trec 2 only: GE, Conquest
Trec 3 only: Westlaw, Logicon, Xerox, NSA, NEC
also RMIT, Trec 3, was part of the CITRI group, Trec 2
Rutgers had two groups, RutgersB(elkin) and RutgersK(antor)

2) P >= 45 was not shown for Trec 2 Routing in 'R on T': it is included here both to cover new players and for comparison with interactive

3) I do not include Minnesota as I do not know which of their many runs to use

COMMENTS:

1) Ad hoc performance shows a slight rise of the top level from Trec 2 to Trec 3.

2) Routing top performance is similar for Trec 2 and Trec 3.

3) Both ad hoc and routing show a slight drift up over classes, but rather more for ad hoc than for routing: with ad hoc it applies overall, with routing really only for lower blocks.

4) Ad hoc performance for Trec 3 is comparable with routing for Trec 3.

5) On the whole the better performing teams for Trec 3 are those for Trec 2: the odd downward movement (where not a goof) probably reflects a willingness to experiment.

6) Some teams that did poorly for Trec 2 (below the bottom of the table) got it together enough for Trec 3 to be with the main pack.

7) Some new players performed respectably (though one, NSA, perhaps less well than for comfort).

8) One team, NYU, showed ability to do NLP on the full scale, not just in group B.

9) Interactive performance was absolutely poor.

OVERALL REMARKS:

1) The General Findings for Trec 2 given in 'R on T' continue to apply in Trec 3.

2) Many different approaches give similar performance.

3) Good performance is obtained with variants of the statistical approach souped up with phrases etc.

4) Routing performance seems to have reached a plateau.

5) The improvement in adhoc performance must be attributed to the stability of the queries (or query types) and the large volumes of training data, as well as the honed character (even after concept removal) of the queries.

6) (Setting aside any evaluation methodology issues) the poor interactive performance must be attributed to the fact that the queries were already well worked up, and mainstream ad hoc and routing performance is good because of quality queries and lots of training data.

7) ALL of these points are broad brush ones: there may be significant differences within performance blocks, and also none between members of adjoining blocks. Without accepting the degree of similarity between different runs that Jean tague's Scheffe tests show, and also recognising that a Precision difference between 45 and 60 at Document cutoff may be not only statistically significant, but meaningful to a user, I think a conservative view of performance differences is in order.

8) IN PARTICULAR, all of Trec 2 and Trec 3 is within a very specific retrieval context; and as many of the successful systems are complex overlays on simple bases, it is essential to get a better view of how the Trec IR environment is influencing the settings for the many parameters that are often involved in these systems.

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)

☆ U.S. GOVERNMENT PRINTING OFFICE: 1995 — 386-627/37913

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.

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.

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 nongovernment). 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