Improving Accuracy and Run-Time Performance for TREC-4

David A. Grossman
Office of Information Technology
3E09 Plaza B
Washington, DC
dgrossm1@mason1.gmu.edu

David O. Holmes
AT&T Global Information Solutions
Rockville, MD
David.Holmes@washingtondc.attgis.com

Ophir Frieder*
Department of Computer Science
George Mason University
Fairfax, VA
ophir@cs.gmu.edu

Matthew D. Nguyen
Coherent Design Systems
Centreville, Virginia
mnguye6@osf1.gmu.edu

Christopher E. Kingsbury
Hughes Applied Information Systems
Landover, Maryland
ckingsbu@osf1.gmu.edu

Abstract

For TREC-4, we enhanced our existing prototype that implements relevance ranking using the AT&T DBC-1012 Model 4 parallel database machine to support the entire document collection. Additionally, we developed a special purpose IR prototype to test a new index compression algorithm and to provide performance comparisons to the relational approach.

We submitted official results for both automatic and manual adhoc queries for the entire 2GB English collection and the provided Spanish collection. Additionally, we submitted results using n-grams to process the corrupted data. In addition to implementing the vector-space model, we experimented with query reduction based on term frequency. Query reduction was shown to result in dramatically improved run-time performance and, in many cases, resulted in little or no degradation of precision/recall.

1 Introduction

For TREC-4, we implemented relevance ranking queries using SQL on an AT&T DBC-1012 (formerly Teradata) parallel database machine [1]. Additionally, we implemented a special purpose IR prototype to test a new index compression algorithm and to provide performance comparisons to the relational approach.

We submitted official results for the entire 2GB English collection, both for automatic and manual adhoc queries and against the Spanish collection. We also submitted results using n-grams to process the corrupted data.

*This work supported in part by the National Science Foundation under contract number IRI-9357785.

In addition to implementing the vector-space model, we experimented with query reduction based on term frequency. Query reduction was shown to dramatically affect run-time performance without losing a significant amount of precision/recall.

We will briefly describe the implementation of our relational prototype and our special-purpose prototype in Section 2. More detailed descriptions are found in [9, 12]. Sections 3 and 5 will describe the results obtained for our English and Spanish submissions. Section 4 describes our corrupted data results, and Section 6 contains our conclusions and future work.

2 Implementation of the Inverted Index

We developed two separate implementations, a parallel relational approach and a special purpose IR approach.

2.1 A Parallel DBMS Approach to IR

Our approach treats the problem as an application of a relational database system. While parallel implementations of relational database systems are common, parallel implementations of information retrieval (IR) systems are rare. Implementing information retrieval as a relational database application provides a portable, parallel means of implementing information retrieval algorithms.

We model an inverted index with a relation DOC_TERM(*doc_id, term, tf*). A relation, QUERY (*query, term, tf*), indicates the terms in the query and their frequency in the query. DOC (*doc_id, doc_name, doc_weight*) contains the document name and the normalized weight for each document. QUERY_WEIGHT (*query, query_weight*) contains the normalized query weight for each query. Finally, an IDF(*term,idf*) relation stores the inverse document frequency for each term.

Given these relations, the following DBC-1012 SQL will compute a cosine similarity coefficient for a given query: *query_number*:

Ex: 1
```
SELECT a.query, c.doc_name, SUM(a.tf * b.tf * e.idf * e.idf) / SQRT(d.query_weight * c.doc_weight)
  FROM QUERY a, DOC_TERM b, DOC c, QUERY_WEIGHT d, IDF e
 WHERE a.term = b.term AND
       a.term = e.term AND
       b.docid = c.docid AND
       a.query = d.query AND
       a.query = query_number
 GROUP BY a.query, c.doc_name, d.query_weight, c.doc_weight
 ORDER BY 2 DESC
```

For additional details the reader is referred to [7]. The query in Example 1 is of fixed syntactic length. We implemented the cosine query on the AT&T parallel database machine. We implemented the simple dot product on *Microsoft SQL Server V4.2, Sybase SQL Server System 10,* and *Oracle 7.* Practical experimentation shows that this query performs well on each of these systems.

2.2 Special Purpose IR Prototoype

We developed a special-purpose IR system to test a new index compression algorithm. Additionally, we wanted to learn more about the implementation details of an IR system. Our system implements relevance ranking using the popular vector-space model [14].

Although compression of text has been extensively studied [2, 8], we have found little work in the area of compression of inverted indexes. Linoff and Stanfill, the most recent published work on the compression of inverted indexes, combine three techniques to compress their index [11]. We implemented this algorithm, referred to as NS compression, as well as a new algorithm, byte-aligned (BA), and compared the differences. Run-time performance for index creation and compression ratio of both TREC-3 and TREC-4 data are given below:

Run Time Performance (index construction)

Compression Type	TREC-3	TREC-4
none	3:48:47	2:38:59
ns	3:50:58	3:14:47
ba	2:56:42	3:06:58

Compression Ratio

Compression Type	TREC-3	TREC-4
none	.466 : 1	.457 : 1
ns	.128 : 1	.121 : 1
ba	.163 : 1	.157 : 1

We have found that the BA compression algorithm results in faster creation of the inverted index, but yields slightly less compression than NS compression.

Query run-time performance is give in Figure 1. Note that run-time is very slightly improved with BA. All performance numbers given here were obtained when running Sun Solaris 2.4 on an 18 processor SUN Sparc 2000. The data was spread across the processors, but the algorithms implemented were sequential.

3 English Results

3.1 Automatic

We submitted both manual and automatic results for the Category A data. All results for Category A data were submitted using the relational IR prototype. Each section of the corpus was loaded into a corresponding relation, and a larger query to UNION all the different relations was implemented.

In addition to simply loading terms, we also loaded phrases which were recognized with a crude phrase parser. A phrase was defined as a two-term sequence that did not contain a punctuation mark or a stop word.

The topics were parsed in the same fashion and both terms and phrases were incorporated into the queries. Phrase inverse document frequency (IDF) was computed as if the phrase was a single term. All terms other than stop words were used in the query.

3.2 Manual

Our automatic queries returned a union of all documents that contained one or more terms related to a query. The manual implementation restricted answer sets, in many cases, to an intersection of documents.

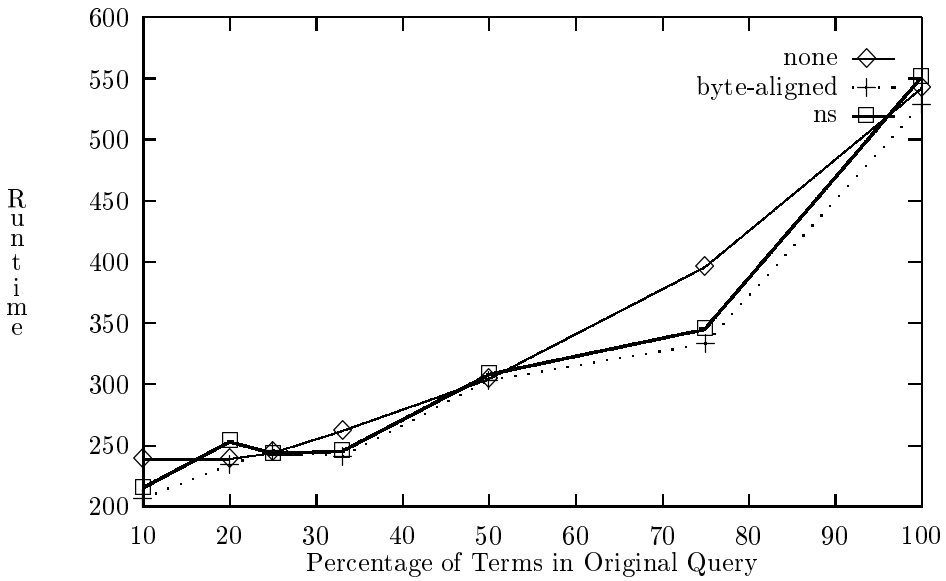

Figure 1: Query Performance (TREC-4, 2GB)

The manual queries were generated after careful study of the query terms. Related terms were placed in at most three sets, and the SQL was generated such that a document was only ranked if it contained at least one term from each set. An optional fourth set contained terms which were not required, but increased the relevance of a document if they were present. Many sets contained lexical variants of the term as we did not perform any stemming. Each set roughly corresponded to a concept.

For example, on topic 203, our answer set was an intersection of two lists. The first set contained the terms {*recycle, recycled, recycling*}. The second set contained the term {*tires*}. Qualifying documents mentioned one of the three forms of *recycle* and referenced *tires*.

To increase the relevance of documents about the economic benefit of recycling tires, we included terms and phrases such as *car tires*, *automobile tires*, *cost effective* and *profitable* to the optional fourth set. The inclusion of a document into the answer set was not dependent upon these extra terms and phrases, however, their presence affected the ranking of the document within the answer set.

Finally, world knowledge was used to generate query terms. Topic 205 requests information about paramilitary activity in the United States. For this topic, one set contained {*militia, militias*} and another set was {*Idaho, Oklahoma*} in an effort to ensure that certain states in which the militia have been active were searched.

3.3 Results

Our calibration against TREC-3 data indicated that a threshold of 100 provides the best results for English data. Hence, we used a threshold of 100 for our official results. The hypothesis for the parallel approach was that queries would run in a balanced fashion; that is, the workload for each processor would be approximately equal. This balance is critical if the approach is to be truly scalable. Table 1 indicates the amount of Processor Load Imbalance (PLIB) for CPU and DISK I/O time: $\left(\frac{max-min}{min}\right)$ measured at each query threshold. For all workloads, the processors are fifteen

Table 1: Percent of Processor Imbalance

threshold	*CPU Time*	*Disk I/O)*
10	15.0	1.5
20	12.0	1.5
25	10.4	1.2
33	9.4	0.9
50	2.0	0.7

percent or less out of balance. Given that the workload is balanced evenly among the existing processors, if processors are added, response time will be reduced. Due to resource limitations, it was not possible to empirically validate this scalability hypothesis.

Our overall results for English are given below:

	Avg. Precision	Above Median	Below Median	Equal Median
Automatic	.1385	10	37	2
Manual	.2216	25	22	2

4 Corrupted Data Results

For corrupted data, we tested the use of n-grams and the effect of reducing the number of n-grams in the query based on an automatically generated *query threshold*. All results for corrupted data were obtained using our special-purpose IR prototype.

N-grams have been used for detection of spelling errors [13, 15, 19] and text compression [16]. A survey of text compression and spelling checking may be found in [18]. More recently, n-grams have been used to determine the authorship of documents [10]. Yochum has shown n-grams are effective in implementing a high-speed document routing system [17].

The first similarity measures based on n-grams were done by Ray D'Amore and Clinton Mah in the early 1980's [6]. More recently, at TREC-3, Damashek expanded on D'Amore and Mah's work by implementing a five-gram based measure of relevance [5]. Damashek's algorithm is entirely language independent and relies upon the vector space model but computes relevance using centroid vectors.

Also at TREC-3, Cavnar implemented a vector-space approach using n-grams [4]. The effectiveness of his approach and the claim that n-grams should have resilience to errors resulted in our decision to use n-grams as part of the corrupted data track for TREC-4.

In TREC-3, we illustrated the effectiveness of automatically reducing query size based on term frequency. First, the terms in the query are sorted and only a percentage of these terms are used. We found that for higher thresholds, relatively useless terms are added to the query and there is little, if any, improvement in precision and recall.

We trained for the real data in TREC-4 (both WSJ data on disk 2 and SJMN data on disk 3) by implementing various n-grams and query thresholds against only the WSJ data on disk 2 and running the TREC-3 queries against this data. We tried n-grams of both size three and four and thresholds of 10, 20, 25, 33, 50, 75, and 100 percent of the original query. With trigrams, we obtained between 700 and 800 relevant documents for the various thresholds (increasing for thresholds from 10 to 33 and decreasing for 50 to 100). With 4-grams we found between 800 and 900 relevant documents with the threshold value increasing up to 75 and then decreasing. Hence, we selected a query threshold of 75 and 4-grams for our official results.

Our results for corrupted data are given below:

	Avg. Precision	Above Median	Below Median	Equal Median
Baseline	.1401	11	19	0
10 Percent	.1153	23	26	0

5 Spanish Results

For the Spanish data, we used the special purpose IR system to obtain our automatic results, and the same method that was used for the English queries was implemented to obtain the manual results.

5.1 Automatic Results

We developed a Spanish stop word list by identifying the top 500 most frequent terms and asking a Spanish linguist to determine which ones were really not so common across the language that they should be in a stop list. We calibrated our approach using the first twenty-five topics and the relevance assessments from TREC-3. Results for trigrams, 4-grams, and 5-grams at thresholds of 10, 20, 25, 33, 50, 75, and 100 are given in Figure 2.

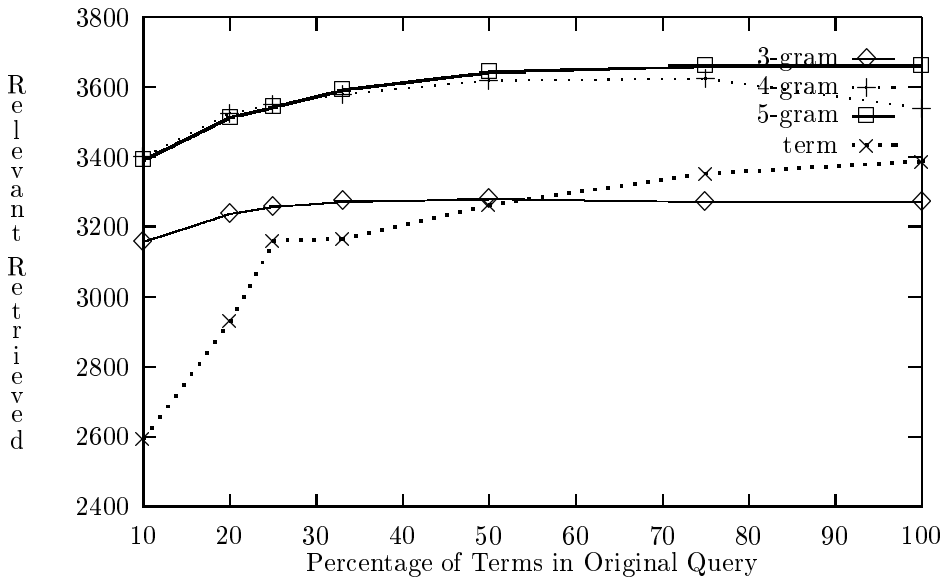

Figure 2: Spanish Relevant Retrieved

We found that our best results occurred with a threshold of 100 using 5-grams. We submitted official results using these parameters.

5.2 Manual Results

After having the queries translated by a Spanish linguist and key terms identified, we grouped the terms in sets and ran the same type of manual queries as done for the English data.

5.3 Results

Our results for Spanish data are given below:

	Avg. Precision	Above Median	Below Median	Equal Median
Manual	.1708	12	11	2
Automatic	.1722	14	6	5

Interestingly, the automatically generated queries performed better than the manual queries. The manual queries used terms, while the automatic used n-grams. The use of n-grams appears to provide a type of automatic stemming as frequently occuring prefixes and suffixes are ignored. Additionally, our lack of Spanish language knowledge made it difficult to develop sophisticated manual queries.

6 Conclusions and Future Work

Given that this was our first year as a Category A participant, we see much room for improvement. Although about half of our queries for both corrupted data and the manual English data were over the median of all participants, half were not.

Our manual queries surprised us with their accuracy. Clearly our automated system needs more improvements, such as passages and relevance feedback. With these improvements, it is likely that our manual queries will improve as well. We are working on enhancing the prototype to expand the manual queries based on relevance feedback.

For corrupted data, we were pleased that the n-gram approach showed resilience to errors. More queries for corrupted data were above the median than below. We will continue testing on the 20 percent corrupted data.

It was interesting that an n-gram approach was calibrated as superior to a term-based approach for Spanish. We look forward to official results to determine whether or not the effects we observed during calibration occurred on the TREC-4 data.

Overall, we were much improved from TREC-3. During TREC-3, we only used the category B data and our precision/recall was a mediocre .0860 and .1356, respectively. Only one of our fifty queries was above the median. Our worst results this year were vastly superior to those submitted last year (our lowest number above the median is ten). We will try to keep up this pace and improve as much in TREC-5.

References

[1] AT&T Global Information Systems. *Teradata DBC-1012 Concepts and Facilities*, March 1992.

[2] T.C. Bell, J.G. Cleary, and I.H. Witten. *Text Compression.* Prentice Hall, Englewood Cliffs, NJ, 1990.

[3] D. Blair. An extended relational retrieval model. *Information Processing and Management*, 1988.

[4] W. Cavnar. N-grams in trec-3. In *Proceedings of the Third Annual Text REtrieval and Evaluation Conference*, 1994.

[5] M. Damashek. Gauging similarity via n-grams: Text sorting, categorization, and retrieval in any language. Submitted to Science, 1994.

[6] R. D'Amore and C. Mah. One-time complete indexing of text: Theory and practice. *8th International Conferencee on Research and Development in Information Retrieval*, pages 155–164, 1985.

[7] D. Grossman, O. Frieder, D. Holmes, and D. Roberts. Integrating structured data and text: A relational approach. To appear in the Journal of the American Society of Information Science, 1995.

[8] P.C. Gutmann and T.C. Bell. A hybrid approach to text compression. In *Proceedings of the Data Compression Conference DCC '94*, 1994.

[9] D. Grossman O. Frieder D. Holmes and D. Roberts. Integrating structured data and text: A relational approach. Submitted to the Journal of the American Society for Information Science, August 1995.

[10] B. Kjell, A. Woods, and O. Frieder. Discrimination of authorship using visualization. *Information Processing and Management*, 30(1):141–150, January 1994.

[11] G. Linoff and C. Stanfill. Compression of indexes with full positional information in very large text databases. *Proceedings of the Sixteenth Annual ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 88–95, 1993.

[12] D. Grossman O. Frieder M. Nguyen and C. Kingsbury. Implementation of an information retrieval system. To be Submitted to Software Practice and Engineering, October, 1995.

[13] J. Pollock and A. Zamora. Automatic spelling correction in scientific and scholarly text. *Communications of the ACM*, 27(4):358–358, April 1984.

[14] G. Salton, C.S. Yang, and A. Wong. A vector-space model for information retrieval. *Communications of the ACM*, 18, 1975.

[15] Lars Erik Thorelli. Automatic correction of errors in text. *BIT*, 2:45–62, 1962.

[16] E.J. Yannakoudakis, P. Goayal, and J.A. Huggill. The generation and use of text fragments for data compression. *Information Processing and Management*, 18(1):15–21, 1982.

[17] J. Yochum. A high-speed text scanning algorithm utilizing least frequent trigrahs. *IEEE Proceedings on New Directions in Computing Symposium*, pages 114–121, 1985.

[18] E.M. Zamora. Survey of spelling correcting methods. *ACM Computing Surveys*, 1983.

[19] E.M. Zamora, J.J. Pollock, and A. Zamora. The use of trigram analysis for spelling error detection. *Information Processing and Management*, 17(6):305–316, 1981.