RC 21040 (log #94205) 12/1/97
Computer Science 8 pages

IBM Research Report

The GURU System in TREC-6

E. W. Brown, H. A. Chong

IBM T. J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden · T.J. Watson · Tokyo · Zurich

The GURU System in TREC-6

Eric W. Brown Herb A. Chong
IBM T. J. Watson Research Center
P.O. Box 704, Yorktown Heights, NY 10598
{brown, herbie}@watson.ibm.com

December 1, 1997

1 Introduction

As the on-line world grows and increases its role in our daily lives, the problems of searching, categorizing, and understanding textual information become ever more important. While researchers and practitioners have made much progress in these areas over the last thirty years, anyone who has gone to the World Wide Web seeking information and returned with more frustration than answers can attest that much work remains. Today's important issues cover topics such as scalability, user interfaces, and techniques that exploit the unique hypermedia features of Web environments.

To support our research in these areas, the Text Analysis and Advanced Search department at the IBM T. J. Watson Research Center has developed an experimental probabilistic text retrieval system called Guru[4]. Guru was originally built to explore new probabilistic ranking algorithms, and now serves as a test-bed for much of our text analysis, search, and categorization work. Guru may be run as a stand-alone system or in a client/server configuration. The Guru indexer performs minimal case and hyphen normalization, but otherwise indexes all words (including stop words) in their original form. The index includes document, paragraph, sentence, and word-in-sentence positional information for each word occurrence in the document collection.

At search time, queries are input to Guru in a free-text format. Stop words are eliminated from the query and morphological variants for each query term are automatically generated and added as synonyms to the query term. Syntax is provided that allows the user to control morphological expansion and stop word elimination. Guru ranks documents using a probabilistic algorithm that considers the frequency statistics of the query terms in individual documents and the collection as a whole. Guru also considers *lexical affinities* (LAs), which are co-occurrences of two terms within a given distance. These automatically identified "phrases" are ranked higher than instances of the component words occurring outside of the LA distance.

Our purpose in participating in TREC-6 is four-fold. First, we continue to refine the base probabilistic ranking algorithm in Guru and wish to evaluate its performance on a large, standard test set. Second, we are developing a prototype user interface and seek initial feedback and guidance for further development. Third, we are interested in text search scalability as an issue orthogonal to the basic problems of search and categorization and seek feedback on initial attempts to address this issue. Fourth, hypermedia domains, such as the World Wide Web, are an increasingly important arena for application of text analysis and search technology. Such domains, however, pose a challenge for evaluation since both search and navigation must be considered by the evaluation metric. We hope that this issue will be addressed by the TREC community with the ultimate goal of defining appropriate evaluation metrics and building suitable test collections. Toward these ends, we are participating in the Ad-hoc Task, the Interactive Track, and the Very Large Corpus Track of TREC-6.

2 Ad-hoc

Our focus in the Ad-hoc Task was to evaluate the performance of our core ranking algorithm. Our TREC-5 results[5] suggested that Guru was using LA scores in a sub-optimal fashion. Most of our pre-submission work involved determining more appropriate settings for the LA distance and the weight of LA contribution to overall document score. A series of experiments was run on the TREC-5 data over which these parameters were varied. Figure 1 shows the average non-interpolated precision obtained by varying LA distance from 1 to 5 and LA weight from 0 to 1. The plot indicates that our probabilistic ranking formula should give LA terms a weight of 0.1 relative to single terms, and the LA distance should be 5. Note, however, that performance is more sensitive to LA weight than LA distance, and the difference in performance between LA distances 1 and 5 is marginal. This is a useful result, since a larger LA distance yields more LA terms for scoring, increasing the processing required to evaluate a query. If similar effectiveness can be obtained with a shorter LA distance, then the system will run faster.

For TREC-6 we submitted two runs, both in the automatic query construction category. One run (ibmg97a) was generated using the topic description field only, while the second run (ibmg97b) was generated from the topic description plus topic title. The script used to generate the queries extracts text from the appropriate topic fields, strips out certain stop phrases and words (based on previous TREC topics), removes punctuation, and produces a query suitable for input to Guru. Most stop words are left in the query at this stage since they are counted in the LA distance when identifying LA terms. Guru ultimately removes stop words from the query using a list of approximately 250 stop words. The queries were run with an LA weight of 0.1 and an LA distance of 5. Note that Guru currently performs no automatic query expansion or relevance feedback.

The results from our two submitted runs are summarized in Table 1. Combining the topic title with the topic description yields a significant improvement over using the topic

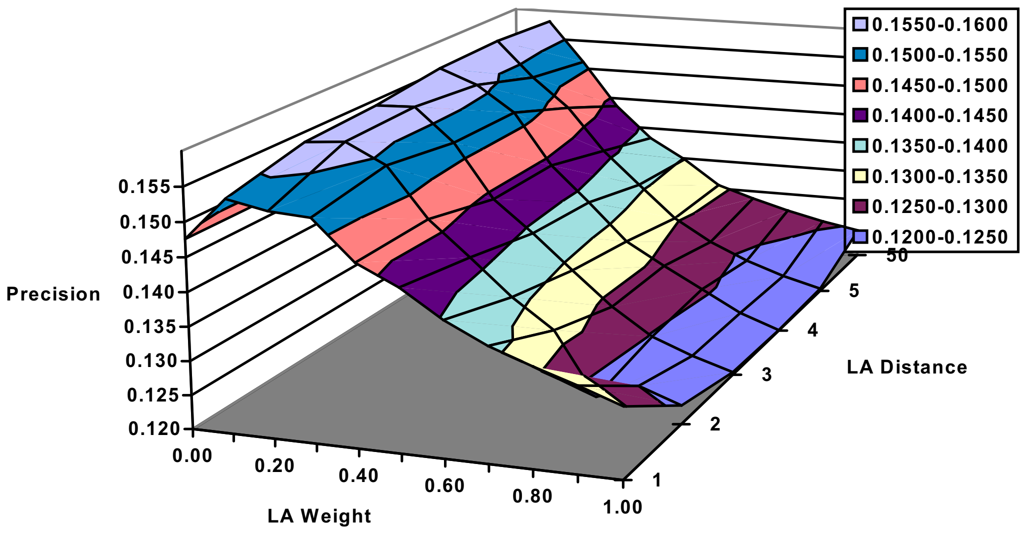

Figure 1: Average precision over varying LA weight and distance

description alone. A quick analysis of the query topics indicates that a number of the topics (e.g., 308, 311, 312, 316, 328) have significant key words, phrases, or unique morphological variations that appear in the title but not in the description. Including these words in the query is usually beneficial. We note, however, that performance deteriorated significantly on at least one query (314) when the title was included in the query. The reason for this has not yet been determined, though the average precision over all TREC-6 participants on topic 314 was worse in the Automatic Long-topics Task than in the Automatic Short-topics Task (Ad-hoc Category A).

Of the 16 participants in the Ad-hoc Automatic Category A Long-topics Task, Guru produced the best average precision (non-interpolated) for 7 of the 50 topics. Guru performed above the median average precision on 29 topics, and below the median average precision on 21 topics. So far we have found no strong correlations between query characteristics and Guru performance, although the topics for which Guru performed best tend to have a relatively small number of relevant documents. This might suggest that inclusion of the topic narrative or the use of automatic query expansion or relevance feedback techniques by other participants tended to reduce their precision. Alternatively, this might suggest that the incorporation of automatic query expansion or relevance feedback techniques into Guru would improve our performance. Of course, we prefer the second hypothesis and plan to prove it in the future.

Interpolated Precision			
Recall	ibmg97a	ibmg97b	
0.00	0.5709	0.6659	(+16.6)
0.10	0.3557	0.4516	(+27.0)
0.20	0.3080	0.3842	(+24.7)
0.30	0.2578	0.3191	(+23.8)
0.40	0.2180	0.2614	(+19.9)
0.50	0.1716	0.2255	(+31.4)
0.60	0.1331	0.1929	(+44.9)
0.70	0.0642	0.1404	(+118.7)
0.80	0.0283	0.0613	(+116.6)
0.90	0.0086	0.0383	(+345.3)
1.00	0.0071	0.0234	(+229.6)
Ave. prec. (non-interp)	0.1727	0.2309	(+33.7)

Table 1: Ad-hoc automatic category A results (a = short, b = long)

3 VLC Track

We can attack the execution performance issues associated with large text collections at a variety of levels. Low level techniques, such as Smeaton and van Rijsbergen[7], Buckley and Lewit[2], and Brown[1], use thresholds and constrained candidate document sets to reduce the amount of work performed in the core ranking algorithm. Higher level techniques, such as Stanfill[8], Tomasic and Garcia-Molina[9], and Cahoon and McKinley[3], use parallel or distributed architectures to scale IR system performance. We opted for a high level approach in the VLC Track.

We ran our VLC Track tests using an experimental distributed search system that performs collection fusion across distributed document collections. Our system can distribute queries to an arbitrary number of search servers in parallel and merge the results into a single hit-list. Result merging was performed without rank normalization, working under the assumption that the documents were distributed in such a way that collection wide term statistics were approximately consistent across all search servers. Guru was used as the search server in all cases, with an LA weight of 0.1 and an LA distance of 1.

For our VLC runs we used the same queries as were used to produce our Ad-hoc ibmg97a run (automatically generated from topic descriptions only with no automatic feedback or query expansion). Our Baseline run was conducted on a single RS/6000 43P/140. The full VLC run was conducted on a network of six workstations, including four RS/6000 43P/140's, one RS/6000 C20, and one RS/6000 42T connected by a 16Mbit token ring. The data for the full run was stored on only four of the six workstations due to disk space limitations on two of the machines. Figure 2 shows the distributed architecture and distribution of data.

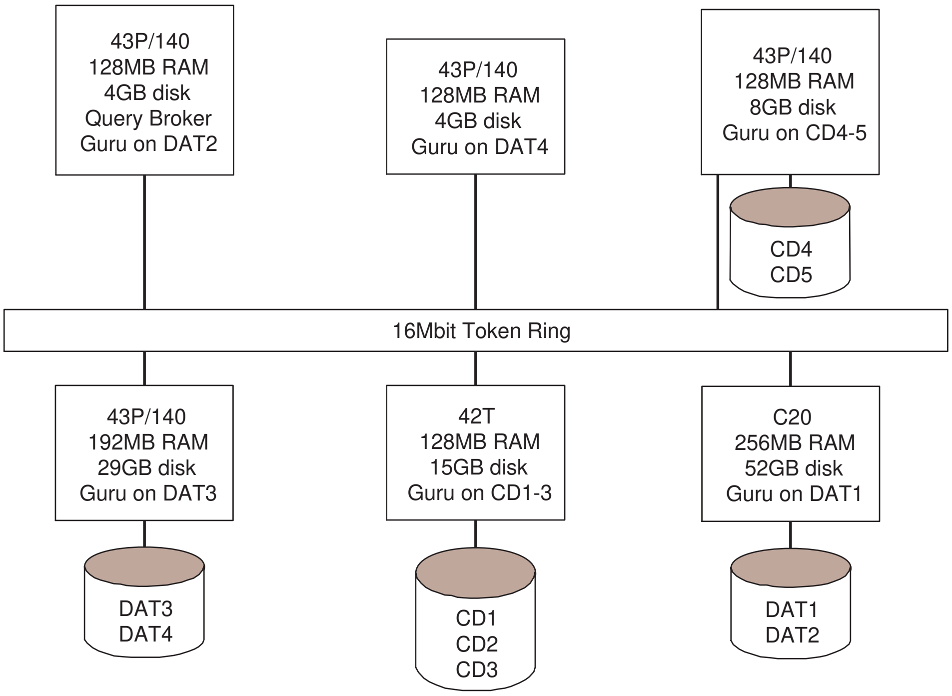

Figure 2: VLC system architecture, showing distribution of data and processing

Collection	Search Servers	Batch Time (min.)	Ave. Prec. @ 20
Baseline (2.02GB)	1	16.5	0.275
Full VLC (17.8GB)	6	47.2	0.361

Table 2: VLC results

Our VLC results are summarized in Table 2. The batch time reported is the time to process all 50 queries in a single batch. Note that our Full VLC results were obtained on an incomplete VLC collection. Our indexer was unable to parse the long web server log files contained in the AUNI collection on DAT3, causing that collection indexing run to fail. Also, a large portion of the NEWS08 collection on DAT4 failed to unload from our tape and was not indexed. Unfortunately these errors were not detected in time to make the necessary corrections before the runs had to be submitted.

Our results are generally encouraging. Average precision at twenty documents actually improves from the Baseline to the Full VLC. We would like to conclude that this validates our collection fusion approach with no rank normalization. However, all seven participants in the track experienced a similar effectiveness improvement from Baseline to Full VLC. More analysis is required to determine if this improvement is due more to characteristics of the collections than quality of the retrieval systems.

Our system found an average of seven relevant documents in the top twenty documents returned. This is six fewer than the best performing system in the track, which found an average of thirteen. We note, however, that our queries did not include the topic titles. As discussed in Section 2, adding topic titles to the queries significantly improves performance. Unfortunately, evaluating the results of queries with topic titles now using the relevance judgments produced by the VLC track would be inconclusive since the coverage of those judgments is so small. Without having submitted title plus description runs to the assessors we can only predict, based on our experience in the Ad-hoc task, that our effectiveness would improve.

The execution performance of our system is satisfactory. Under ideal circumstances, we would expect that using 9 machines configured similarly to the baseline machine (each searching data stored on local disk) would allow us to search 18 GB in approximately the same amount of time as required to search the Baseline data (i.e., 16.5 minutes). This ideal architecture was not available, however, forcing us to distribute the Full VLC data such that the largest individual collection searched was over 4 GB, two of the six servers accessed their index data from remote disk, and two of the six servers had to additionally act as file servers. This allows the scalability of a single search server to limit the performance improvement obtainable in the distributed architecture. The Guru search server is a prototype designed for exploring ranking algorithms. As such, execution performance is not a priority. In spite of this, we achieve execution speeds that scale well with collection size. The single machine (Baseline) system requires 9.6 milliseconds/megabyte/query, while the six machine distributed (Full VLC) system requires 3.1 milliseconds/megabyte/query. These results were obtained with the machines in multi-user mode and the runs were made during the evening when the machines and network were lightly loaded.

Our participation in the VLC Track at TREC-6 was motivated by two goals: 1) to obtain preliminary performance data on our prototype distributed search system, and 2) to contribute to the development of a large, realistic test set with meaningful queries and relevance judgments. We have achieved both of these goals to a certain degree. Attempting to index and search 20 GB of data produced valuable feedback on our prototype and identified a number of opportunities for further work. As a participant, we contributed to the track discussions and submitted documents to the judging pool. Ultimately, the relevance judgments for the VLC track appear to have less utility than originally hoped. While the participants can use them to evaluate their submitted runs, the relevance judgments should not be used to evaluate runs that did not contribute to the judging pool. The coverage of the relevance judgments is simply too small, rendering any effectiveness evaluations inconclusive.

4 Interactive Track

Our participation in the Interactive Track is described in detail in Schmidt-Wesche *et al.*[6] To conduct our Interactive Track experiments, a number of search and text analysis systems were used, including IBM's NetQuestion search engine and Guru. NetQuestion is a hybrid Boolean and probabilistic free-text ranking system. It uses a version of the Guru ranking algorithm to perform probabilistic free-text ranking.

5 Summary

This is the second year in which the Text Analysis and Advanced Search department at the IBM T. J. Watson Research Center has participated in the TREC conference. For TREC-6 we have expanded our participation beyond the Ad-hoc task to include the Interactive and VLC tracks. While we continue to improve our core ranking algorithm, our efforts this year were focused mainly on the Interactive and VLC tracks. In the future, we hope to see TREC activity in the hypermedia domain. Information retrieval in a hypermedia environment is a unique combination of search and navigation. The creation of appropriate hypermedia test collections and metrics would be of great value to this research community.

Acknowledgements

Thanks go to Paul Jensen for helping prepare the VLC data.

References

[1] E. W. Brown. Fast evaluation of structured queries for information retrieval. In *Proc. of the 18th Inter. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 30–38, Seattle, WA, July 1995.

[2] C. Buckley and A. F. Lewit. Optimization of inverted vector searches. In *Proc. of the 8th Inter. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 97–110, June 1985.

[3] B. Cahoon and K. McKinley. Performance evaluation of a distributed architecture for information retrieval. In *Proc. of the 19th Inter. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 110–118, Aug. 1996.

[4] Y. S. Maarek and F. A. Smadja. Full text indexing based on lexical relations. In *Proc. of the 12th Inter. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 198–206, Cambridge, MA, June 1989.

[5] Y. Ravin. The GURU system in TREC-5. In D. Harman and E. Voorhees, editors, *The Fifth Text REtrieval Conference (TREC-5)*, Gaithersburg, MD, 1997. National Institute of Standards and Technology Special Publication 500-238.

[6] B. Schmidt-Wesche, R. Mack, C. Cesar, and D. VanEsselstyn. IBM search UI prototype evaluation at the interactive track of TREC6. In D. Harman and E. Voorhees, editors, *The Sixth Text REtrieval Conference (TREC-6)*, Gaithersburg, MD, 1998. National Institute of Standards and Technology Special Publication.

[7] A. F. Smeaton and C. J. van Rijsbergen. The nearest neighbour problem in information retrieval. An algorithm using upperbounds. In *Proc. of the 4th Inter. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 83–87, Oakland, CA, 1981.

[8] C. Stanfill. Parallel computing for information retrieval: Recent developments. Technical Report TR-69 DR88-1, Thinking Machines Corporation, Cambridge, MA, Jan. 1988.

[9] A. Tomasic and H. Garcia-Molina. Performance of inverted indices in distributed text document retrieval systems. Technical Report STAN-CS-92-1434, Stanford University Department of Computer Science, 1992.