IBM Research Activities at TREC

Eric W. Brown[*] David Carmel[†] Martin Franz[‡] Abraham Ittycheriah[‡]
Tapas Kanungo[§] Yoelle Maarek[†] J. Scott McCarley[‡] Robert L. Mack[*]
John M. Prager[*] John R. Smith[*] Aya Soffer[†] Jason Y. Zien[†]

IBM Research Division

Draft v1.3, September 15, 2003[1]

1 Introduction

Several groups in IBM's Research Division have participated in TREC, with differing goals. This chapter summarizes some of their activities and the conclusions they reached. It should be emphasized that there was no central plan in these activities – individual groups decided to participate in the way that made sense to them at the time. In effect, participation in TREC was a tool they used to further their diverse research agendas. In some cases, groups participated once or twice in order to explore some issue; in other cases, regular participation in certain TREC tracks has been a central component of a group's research agenda.

In this chapter we will first describe IBM Research's participation in the TREC tracks most relevant to the classic document retrieval task, and in the tracks related to user interfaces for search and cross-language document retrieval. Then we will outline the extensive series of submissions made to the Question Answering track, and finally discuss IBM Research's activities in multimedia retrieval at TREC.

2 Document Retrieval

The Guru [1] information retrieval system, first developed by Maarek during her doctoral work at the Technion [2], and then significantly extended at IBM, after she joined the T.J. Watson Research Lab, has been the basis for several TREC submissions from IBM Research groups. Different versions of Guru have emerged as the original code has been rewritten and adapted. The two following sections describes work done with one such version, which retained the original name, and was used in submissions to the Ad-hoc track and the Very Large Corpus (VLC) track. Then, a submission to the Web track using a Java version, called Juru, is described, followed by a discussion of a submission to the Homepage Finding track, using a different system. Finally, the IR system used by the Natural Language Systems group is briefly described.

[*]T.J. Watson Research Center, 19 Skyline Drive, Hawthorne NY 10532, USA
[†]Haifa Research Laboratory, University Campus, Carmel Mountains, Haifa, 31905, Israel
[‡]IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, USA
[§]Almaden Research Center, 650 Harry Road San Jose, CA 95120, USA
[1]Collated by Alan D. Marwick, *marwick@us.ibm.com.*

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)
Avg. prec. (non-interp)	0.1727	0.2309	(+33.7)

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

2.1 Guru at TREC-5 and TREC-6 [2]

Guru was used in submissions to the ad-hoc tracks of TREC-5 and TREC-6, and to the Very Large Corpus (VLC) track of TREC-6 (see next sub-section). The motivation for participating in these tracks was to evaluate the probabilistic text retrieval algorithms in this version of Guru on a large, standard test collection, to explore issues of scale and performance, and to verify that Guru provided a sufficient level of retrieval effectiveness to support other research systems in which it was used as a component.

The version of Guru used in this work 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 in the collection as a whole. Guru also considers *lexical affinities* (LAs), as multiple-word indexing units in addition to regular single words. LAs were first introduced by Saussure in 1947 to represent the correlation between words co-occurring in a given language and then restricted to a given document for IR purposes [1]. LAs are identified by looking at pairs of words found in close proximity to each other.

2.1.1 The Ad-hoc track at Trec-6

Our participation in TREC-5 served mainly as a learning experience for the group concerned in how to handle a relatively large (for 1996) text collection and follow the TREC experimental methodology. Here we limit our discussion to TREC-6 results [3].

Our focus in the TREC-6 Ad-hoc Task was to evaluate the performance of our core ranking algorithm.

[2] Eric Brown

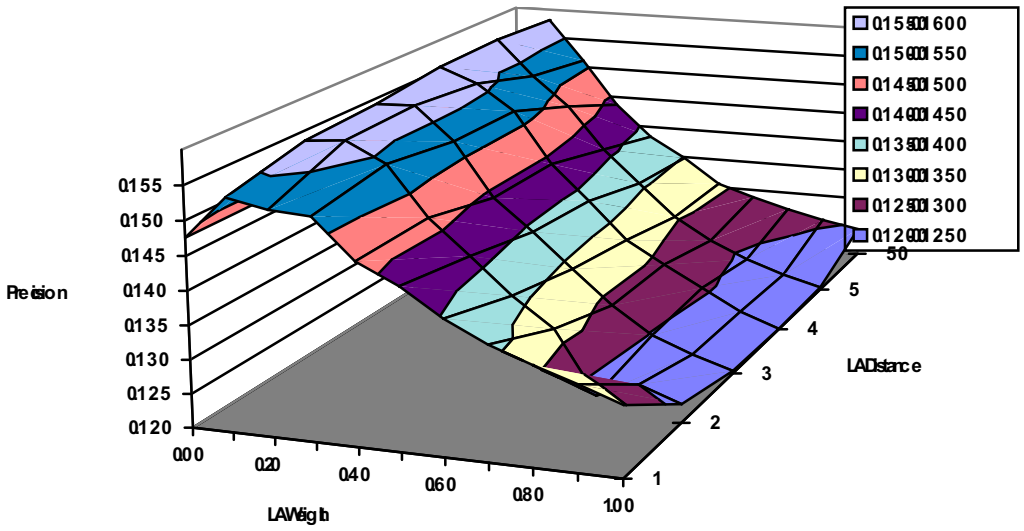

Figure 1: Average precision as a function of LA weight and distance

Our TREC-5 results [4] 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. We ran a series of experiments 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 words, and the weight given to LAs in the ranking algorithm 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 performed 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 yielded a significant improvement over using the topic description alone. A quick analysis of the query topics indicated 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 was usually beneficial. 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

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

topics. Guru performed above the median average precision on 29 topics, and below the median average precision on 21 topics.

Overall we felt that this was a satisfactory result, considering that the submitted runs did not use query expansion or relevance feedback. The results we obtained by participating in TREC allowed us to obtain the information we were seeking as to the optimum LA weight and length for the version of Guru we were using.

2.1.2 The VLC Track at Trec-6

The Very Large Collection (VLC) track gave us an opportunity to evaluate an approach to using a distributed system to search large collections of text that was being considered in our group at the time. 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 [5], Buckley and Lewit [6], and Brown [7], 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 [10], 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.

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 were generally encouraging. Average precision at twenty documents actually improved from the Baseline to the Full VLC. Our system found an average of seven relevant documents in the top twenty documents returned. This was 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

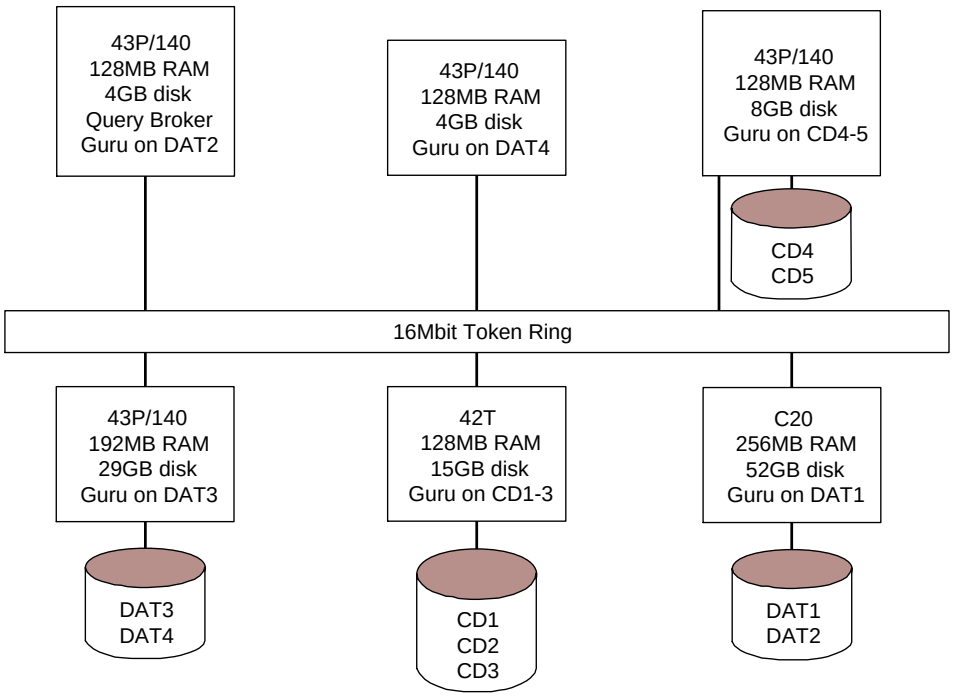

Figure 2: VLC system architecture, showing distribution of data and processing

Section 2.1.1, adding topic titles to the queries significantly improved performance.

The execution performance of our system was 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. 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 allowed the scalability of a single search server to limit the performance improvement obtainable in the distributed architecture. In spite of this, we achieved 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.

Although the absolute performance times reported here pale in comparison to the performance achieved by modern Web search engines, our results at the time demonstrated that a distributed search system (in particular, a network of workstations) was a viable approach to achieving scalable search on large text collections.

2.2 Juru in the Web Track

2.2.1 Juru at TREC 10 [11]

Juru participated in TREC for the first time in 2001 (TREC 10). Juru is a full text search engine purely written in Java. It was developed at IBM's research lab in Haifa and is based on the Guru search engine [1].

Following the classical inverted index approach, Juru creates an index by associating terms with the documents that contain them, and then storing this mapping in inverted files for efficient retrieval. For Web data, Juru uses a description database that provides for each page p in the collection a set of descriptions extracted from other pages that cite (i.e., link to) p. Juru indexes every page based on its content as well as its set of descriptions. Simulations showed that using descriptions as indexing units for HTML pages improved precision by about 20%. Juru's query evaluation is based on the ranking process of the SMART system [12]. Juru makes use of Lexical affinities (LAs), in order to improve search precision (see Section 2.1).

The main goal of our experiments was to validate a novel pruning method, first presented at [13], that significantly reduces the size of the index with very little influence on the system's precision. By reducing the index size, it becomes feasible to index large text collections such as the Web track's data on low-end machines. Furthermore, using our method, Web search engines can significantly decrease the burden of storing, caching, or backing up extremely large indices by discarding entries that have almost no influence on search results. For the TREC 10 experiments we used a lossy pruning method that prunes the index at the posting level. The idea is to remove those postings whose potential contribution to the relevance score of a document is so small that their removal will have little effect on the accuracy of the system.

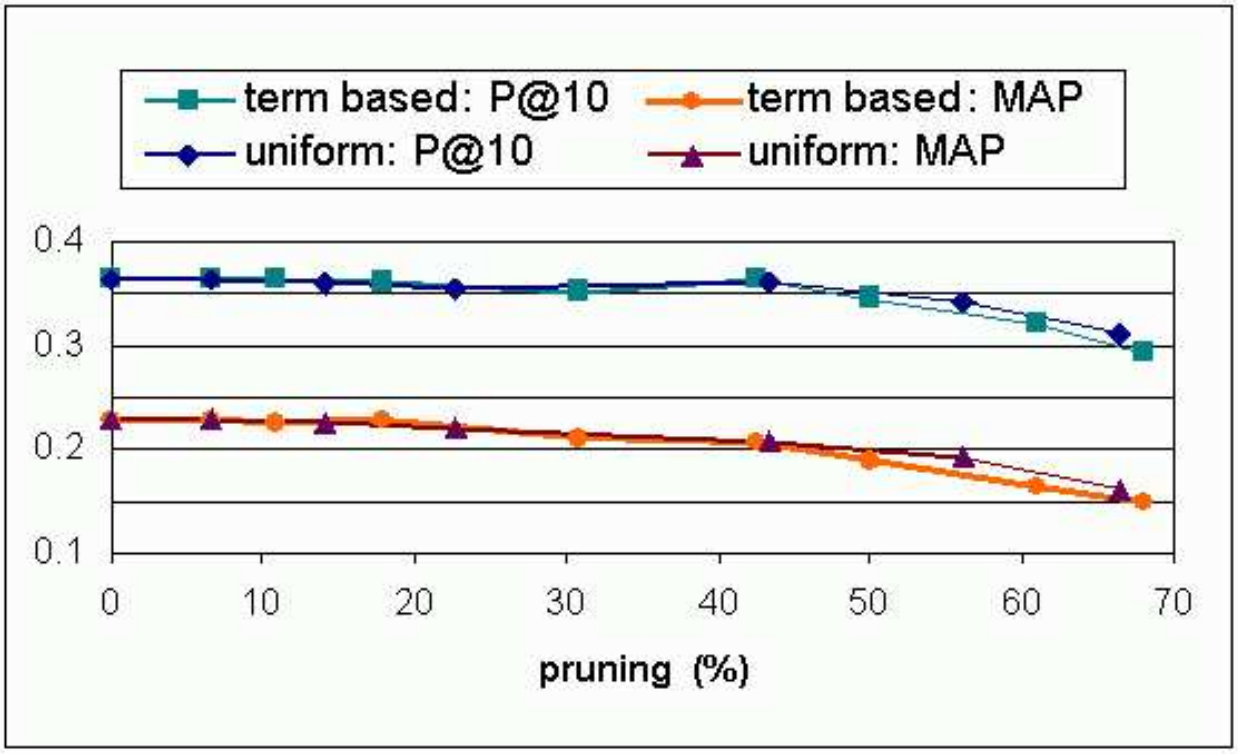

Figure 3: The impact of pruning on precision as measured by mean average precision and precision at 10 documents.

Our experiments in TREC 10 tested the impact of pruning on the search results. We created a sequence of pruned indices using several variations of our pruning algorithms. For each index we ran 50 queries, constructed automatically from the titles of topics 501-550. Our first experiment tested the effect of pruning on the precision of the results. Figure 3 shows the impact of pruning on precision as measured by mean average precision (MAP) and precision at 10 documents (P@10). From these tests, it is apparent that P@10

remains more or less stable up to 40% pruning. Although there is a slight decrease in average precision at 30% pruning, a significant loss of MAP occurs only at 40% pruning.

In addition to validating the pruning methods, the Web Track results also demonstrated the overall high quality of the Juru search engine. Juru achieved excellent results ranking first in precision@10 and second in average precision. Especially interesting in the context of the index pruning experiments is the fact that Juru's four runs which included up to 35% pruning ranked 1-4 in terms of precision@10.

2.2.2 Juru at Trec 2002 [14]

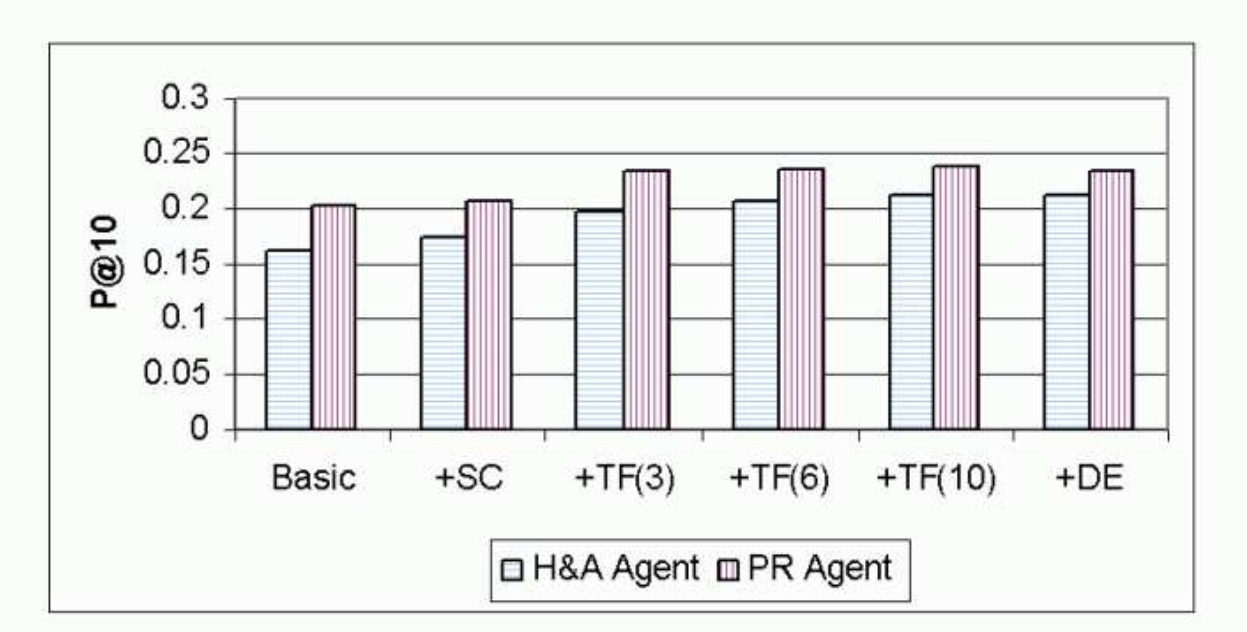

Figure 4: Filter contribution to Juru's P@10 at Trec 2002.

For Trec 2002, a new "Topic Distillation" task was proposed for the Web track. Topic distillation involves finding a list of key resources for a particular topic. According to the Web-Track guidelines, a key resource is a page which would be included, if someone were to build a short list of key pages in a topic area. Our main goal was to experiment with the Knowledge Agent (KA) technology [15], previously developed at our Lab, for this particular task. The knowledge agent approach was designed to enhance Web search results by utilizing domain knowledge and link analysis. We experimented with two type of agents, the H&A agent which uses hub and authority computation [16], and the PR agent which uses Page-Rank computation [17] for link analysis. The set of top 10 results returned by the agent was further distilled by a set of filters:

- Site compression (SC) filter: ensure that the top-10 results of each query will indeed be diverse.
- Title filter (TF): the similarity of a page title to the topic title is used as another relevance filtering mechanism.
- Duplicate elimination filter (DE): invoked to filter out duplicate results.

Our experiments tested the specific contribution of the various distillation filters. Figure 4 presents the average P@10 for each run. For the Basic run, no filter was invoked. For the +SC run, the SC filter was invoked on the results of the Basic run. For The +TF(k) runs, the TF filter was invoked on the results of the +SC run, varying parameter k - the number of titles to filter. Finally, for the +DE run, the DE filter was invoked on the results of the TF(10) run. We performed the experiments using both the original KA algorithm (H&A agents) and the static ranking algorithm (PR agents). We can clearly see the contribution of the SC and TF filters. From these results, it is clear that the PR agent with maximal TF filtering (k= 10) indeed achieved the best results, and was ranked fourth among all participants.

2.3 Integrating Link Structure and Content Information for the Homepage Finding Task[3]

Web documents have an interesting feature that distinguishes them from other document datasets — they have inter-document links. TREC evaluations focussed on two most common modes in which user uses a web search tool to find information i) information navigation and ii) content finding. In information navigation, the user would like to get to the homepage that has links to the information the user is looking for. Thus while the homepage itself may not have the information the user is looking for, it has the links to the content pages. This was evaluated under the "homepage finding" task. The "Adhoc" evaluation task, on the other hand, focussed on finding pages that actually contain the formation the user is looking for.

The IBM Almaden system focussed on the Home Page Finding task at trec 10 combined both the link and content information in a unique way to score documents [18]. The system consists of three components: the indexer, the DocRanker, and the query engine. The indexer stems and tokenizes the documents and records the presence of various attributes: capitalization, presence in title or bulleted lists, color, term and document frequency, etc. Furthermore, to take advantage of contextual cues on a page, we made use of heading and title information by giving more weight to term occurrences in those contexts. Next, the DocRanker ranks documents by extracting the link structure of the crawled pages, then computing the SameSite function [19], and finally computes the PageRank [17], on the graph. Our PageRank calculations use a weighted graph where the weight of a link was 1.0 if the link was composed of two pages from different sites, and 0.0001 if they were on the same site, to deemphasize self-references. The query engine retrieves the filtered set of documents and ranks them using our integrated ranking function.

Although PageRank [17] provides useful information for scoring, it is unclear how this information should be combined into the page-content-based scoring function of Okapi/Inquery [20]. We propose a new scoring function that augments the TF*IDF model with document ranking. Let us examine the Inquery/Okapi function's TF component in detail. The component $1.5|D_d|/A$ is the only portion of the function that contains document-related information. Here $|D_d|$ is the size of the document and A is the average size of documents. This component provides a bias based on the importance (that is, size) of a document. We propose incorporating document rank (DocRank) ρ_d into this component. ρ_d is the scaled ordinal rank of a page. For instance, if the document is the the 3rd ranked document in a collection of N documents, $\rho_d = 3/N$. Note that in particular, the DocRank is not the actual PageRank value. Rather, the use of ordinal rank provides a smoother, more gradually changing value than the actual PageRank. Also, it is obvious that any algorithm that can generate an ordering of documents could be used in place of PageRank for our purposes. We experimented with two forms of combining of combining DocRank with the document component of the score: i) multiplicative: $1.5\rho_d|D_d|/A$ and and ii) additive: $\alpha\rho_d + 1.5|D_d|/A$, where α is a user-specified constant. After experimentation we found the additive form was more effective. We set $\alpha = 10.0$. Rest of the Inquery/Okapi model was unchanged.

[3]Tapas Kanungo and Jason Y. Zien

Table 3: Home Page Finding Results

Metric	Rank not used	Rank used
Average reciprocal rank over 145 topics	0.382	0.611
Number of topics for which entry pages found in top 10	90 (62.1%)	113 (77.9%)
Number of topics for which no entry pages was found	17 (11.7%)	15 (10.3%)

Results for the Home Page Finding task are shown in Table 3. The results show clearly that when our DocRank scoring is used, both the average reciprocal rank and the top ten scoring method showed substantial improvement. Using linkage information was a clear win with Home Page Finding.

In summary, we introduced a novel scoring function that combines TF*IDF scoring with link-based ranking. Our experiments showed that this combined scoring method was exceptionally well-suited to Home Page Finding.

3 The Interactive Track at TREC 6[4]

In 1997, NIST initiated a "Interactive Track" (IT) aimed at evaluating the effectiveness of end-user interfaces (UI's) for search, and IBM Research participated. The goal of the interactive track was (and still is) to compare peoples' search behavior when using different search UI's built by participating institutions, and with a standard NIST search UI (and search engine). The rationale for the IT is that the usability of a search function depends not only on the quality of the underlying search technology (as measured in other tracks), but also on how these search capabilities are made available to end users in the search user interface.

The interactive track specified an experimental design appropriate for behavioral evaluation using human participants. The design specified a corpus (Wall Street Journal news articles), a set of search scenarios, criterion for successful task completion, guidelines about the experience and background of participants, and the experimental procedure, involving instructions, and so on. Within this framework, IBM Research's participation focused on search UI features that were of interest to internal IBM studies of IBM's external Web site as it was at that time. We were especially interested in obtaining evidence for the usefulness of emerging "Web" search conventions involving the use of "+" and "-" operators. The overall track results are reported in detail in the TREC Proceedings for 1998 [21], and IBM's quantitative performance on accuracy and completion rate for search tasks, is reported in Schmidt-Wesche, Mack, Cesar and VanEsselstyn [22]. Although the reported rank of IBM's submission was near the bottom of twelve participants, the differences between the twelve systems were not statistically significant. That is, the variability attributable to the experimental manipulation (search system and UI), was not sufficiently different from that due to "chance" (uncontrolled sources of variability, such as individual differences, differences in lab and experimenter context, etc.) Thus TREC 6 was a learning experience for the research community interested behavioral issues for search. Of more ultimate value for IBM was qualitative evidence for how people interpreted the Web search syntax. People's statements made spontaneously and in debriefing supported the interpretation we designed for the syntax. It also helped us to design on-line hints and tips for the search syntax of IBM's external search site. IBM has not participated in subsequent Interactive tracks, although they have continued. The search goals and UI issues for IBM's internal and external Web sites are sufficiently different from the requirements of the TREC IT that we were not able to test those issues at TREC. However, our participation in the first

[4]Bob Mack

interactive track was quite useful for helping us investigate questions relevant to IBM's Web search methods.

4 Cross Language Retrieval[5]

The following subsections describe the activities of the Natural Language Systems group at the IBM T. J. Watson Research Laboratory in Yorktown Heights.

4.1 Monolingual IR Techniques

Our monolingual IR system is a common subsystem for our Multilingual IR, Spoken Document Retrieval, and Question Answering systems; as a stand-alone system, it also performed strongly in the Ad-hoc tasks in TREC [23, 24, 25] and scaled to the Very Large Corpus in TREC-6. The input to our system is prepared using several text pre-processing components including a decision tree tokenizer, a part-of-speech tagger, and a morphological analyzer instead of the typical lexical stemmer. After removing stop words, word and bigram features were indexed for scoring. Query-document relevance scores were computed using a two-pass strategy. The first pass used an Okapi-like [26] $tf \cdot idf$ scoring formula. The second pass was based on a query expansion approach. We experimented with a variety of techniques for constructing expanded queries, including k-Nearest Neighbors [23], a probabilistic model of document generation [27], and an LCA-like [28] approach. The second pass scoring was performed using either an Okapi-like [26] formula or the probabilistic model. Other techniques we experimented with included scoring with passage/document combination, sigmoidal adjustment factor for document length, "stop-characters" instead of stop-words in monolingual Chinese, and suppressing the scores of correlated features.

4.2 Multilingual IR: English, French, German, and Italian

For the TREC-7 and TREC-8 CLIR tasks we constructed a system to retrieve English, French, German, and Italian documents given English or French queries. Building a multilingual (as opposed to a bilingual) system is an exercise in leveraging bilingual resources of varying quality and types [24, 25]. No single solution is ideal for all language pairs.

Between English and French, we trained statistical machine translation models (IBM Model 1, [29]) on the Canadian parliamentary proceedings. We captured local context by incorporating a trigram language model to produce a "Fast MT" system ([30]) which could translate large document collections at high speed. We used this system to investigate the merits of query translation versus document translation. We expected query translation to be more fragile, because queries are short and there is little opportunity for the IR system to recover from mistranslations. On the other hand, documents are longer, and there is more opportunity for at least some instances of a word to be translated correctly. In practice, we found that a combination system outperforms either [31]. Thus our TREC English-French system mixed the scores from a query translation system that translated English queries into the docuemnt language French and performed Okapi-based scoring in French, and the scores from a document translation system that translated the document collection offline from French into English, and performed Okapi-based scoring in English. We combined the two systems with a linear combination of scores. Our system to retrieve English documents from French queries was identical, with French and English interchanged.

Incorporating German and Italian documents into the system posed a new challenge: we lacked a good parallel corpus for these languages, and an artificial German-English parallel corpus produced with a commercial MT system was found to be of little use. However the retrieval corpus itself contained a valuable

[5] Martin Franz and J. Scott McCarley

trilingual resouce: the SDA (a Swiss newswire agency) part of the corpus contained French, German and Italian news stories which frequently reported the same events in different languages, even though the stories were not translations of each other. We did not know which stories in one language should be paired with which stories in another langauge. We refer to a corpus with this structure as a *comparable* corpus, to distinguish it from a parallel corpus. Although we did not know which pairs of articles referred to the same event, we used words with identical spellings in both languages (typically names) to find reasonable candidate pairs, and constrained the search by the dates of the article. We then trained a translation model based on these pairs and extracted a bilingual dictionary from this translation model. We then used the word-pairs in this bilingual dictionary to find more candidate document pairs and repeated the entire process. This procedure produced translation models suitable for use in our French-German and French-Italian retrieval systems.

In order to handle English queries on German and Italian documents, we used French as a *pivot* language. We used the English query in the E-F retrieval system to find French documents, then used these French documents to formulate a French query which we issued to the F-G and F-I retrieval systems to find German and Italian documents. This approach worked better than approaches involving combined translation models, such as translating German documents to French and then translating the translations into English.

The TREC evaluation requires systems to output a single ranked list of documents across all four languages for each query. We converted retrieval scores to probabilities by observing that precision was approximately a linear function of log(rank) and estimating a straight line fit separately for each langauge pair.

We note that although English was the strongest query language of most of the contestants, our system produced strong performances with either English or French as the query language — in fact our single best run in TREC-8 was with French queries.

4.3 English-Chinese and English-Arabic CLIR

In order to explore Asian language processing issues, the TREC-9 cross-language task switched from searching European languages to searching traditional Chinese documents with English queries. A particular difficulty with Chinese text is that words are not space-delimited: some form of linguistic signal processing is essential before selecting the basic retrieval units. We built a Chinese segmenter that segmented the text into short words (< 5 characters.) We also experimented with indexing overlapping bigrams of Chinese characters, without regard to whether the bigrams spanned word boundaries [32]. Both approaches yield reasonable performance (it varies from query set to query set which approach produces the better performance, so we used a linear mixture.) An important lesson we learned from this evaluation was how the quality of retrieval varies with the quality and quantity of the parallel corpus used as training data. We found that a system built with a moderate quantity of FBIS translations of news stories outperformed a system built with larger quantities of Hong Kong news and Hong Kong laws translations. Other differences in training data, such as mismatch between simplified and traditional Chinese, were found to have smaller effects.

The TREC-11 cross-language task searched Arabic documents with English queries. The challenge with Arabic is that it is a highly inflected language. We again built an Arabic stemmer and used the morphemes as the basic retrieval units. We confirmed earlier observations [33] that monolingual Arabic retrieval is highly sensitive to appropriate stemming, and assorted character normalizations, whereas crosslingual retrieval is much less so, apparently because many variations are seen in the English-Arabic UN parallel corpus that we use as training data [34].

5 Question Answering

5.1 Predictive Annotation Approach[6]

When it was announced in 1998 that there would be a Question-Answering (QA) track in TREC-8, we were eager to participate due to a desire amongst some of us to develop search technology which would get closer to what the end-user wanted, which in many cases was answers to fact-seeking questions. IBM groups have continued to participate in the QA track. In this section, we describe an approach to the problem based on pre-annotation of the corpus and on indexing, while in the next section a different approach will be discussed.

We realized, as did most or all other QA efforts, that the key to answering questions was identifying the answer-type called for by natural-language questions, and also recognizing instances of such types in text corpora. We already had the Guru search engine [4] as already described, and a named-entity recognizer Textract [35, 36]. For the work described in this section we adapted both of these for the QA task, and in addition we developed other modules. The major innovation that came out of the first stages of this effort was Predictive Annotation [37].

With Predictive Annotation, we run Textract on the corpus prior to indexing, and then index not only the original corpus terms but also the semantic labels produced by the named entity recognizer (e.g. PERSON$, COUNTRY$, DATE$). This then enables us to include in the bag-of-words sent to the search engine the semantic label(s) representing the sought answer type. This approach was expected to greatly reduce the size of the hit-list needed to retrieve documents with answers of the right type. With a smaller (but more precise!) hit-list there would be less noise due to a smaller number of candidate answers. Indeed, subsequent experiments and modelling have both shown that for our system an optimum hit-list size is around 10 passages.

For QA, we adapted the Guru document retrieval engine by making some extensions that we felt would be particularly useful for the QA task, and called it GuruQA. These modifications followed the intuition that, unlike a typical "ad-hoc" query for which one or more documents were to be sought, in QA the answers to questions (especially fact-based questions) would often be found in just one or a small number of sentences. Thus, GuruQA returns passages, not documents; to achieve this, the GuruQA query syntax requires the user to specify the number of sentences in the passages to be searched. Furthermore, unlike in the ad-hoc track, where the strategy is to reward documents for having more occurrences of query terms, a sufficient answer sentence would likely only have a single instance of each query term. Thus the ranking algorithm of GuruQA scored a passage by adding up the number of matching query terms in it (each term could only match once), multiplying the contribution of each term by a weight specified in the call. We used a very simple approximation to idf weighting: common words (i.e. non-proper names) had a relative weight of 1, proper names a relative weight of 2, and the semantic labels indicating the sought answer type a relative weight of 4. In addition, a density contribution to the score was computed, based on the reciprocal of the size of the sub-passage spanning all of the matching query terms.

5.1.1 TREC-8

For TREC-8 we submitted two runs for each of the 50-byte and 250-byte tasks, although we had only trained our system on the 50-byte task. These runs differed in the Answer Selection algorithm, namely the algorithm to select candidate answers from the passages returned in the GuruQA hit-list [38]. The ANSEL algorithm used straight linear regression based on features such as search engine score, number of words in segment

[6] John Prager

that were not in the query, average distance of query words in the segment from beginning of segment, and so on. The WERLECT algorithm used similar features but combined in a more ad-hoc fashion. Our TREC8 results [39] were as follows. The ANSEL runs were uniformly 10% better than WERLECT, and achieved an MRR of 0.319 in the 50-byte task and 0.430 in the 250-byte (these scores positioned us at 4th out of 20 and 10th out of 25 respectively).

5.1.2 TREC-9

Our TREC-9 entry used the same system as for TREC-8 but with a series of minor improvements. Using the TREC-8 data for training, we determined that a passage window size of 1-3 sentences and hit-lists of 10 documents gave optimal performance. We added more semantic classes and refined the recognition patterns for the existing ones. We noticed that in TREC-8 our system did particularly poorly with definition questions of the form "What is X", since with that kind of question we were unable to identify in advance the desired answer type, thus losing the advantages of Predictive Annotation. We remedied this to some degree by introducing the THING$ class for nouns, but to a greater extent by using the AT&T document set that NIST made available to QA participants who did not have their own search engine. Our use of this set was merely to boost scores for answer passages that GuruQA had found if they were from documents that were also in the AT&T set for that particular question [40]. The effect this had on ultimate system performance was to boost the 50-byte task score by 13% and the 250-byte score by 8%, for the subset of questions (31% of the total) for which this technique was applied.

The TREC-9 questions are generally considered to be harder than TREC-8 because they were extracted from independent query logs, rather than being back-formulated from the corpus. There was thus much more of a vocabulary matching problem. Nevertheless, our scores were comparable to the previous year. Our MRR scores were 0.315 in the 50-byte task and 0.425 in the 250 byte task, putting us 4th out of 35 and 5th out of 43 respectively.

5.1.3 TREC-10

We continued the incremental improvements for TREC-10, but made a couple of qualitative changes in order to introduce more NLP. We replaced our Answer Selection module with one which used fewer features, but with more reliance on linguistic relationships - a research area we have continued to work on up to the present. For definition questions, we developed a WordNet-based algorithm we called Virtual Annotation [41], which found the most commonly used hypernym for the term in question and incorporated that into the search. Unfortunately, our reliance on this method was undercut by what in our view was inconsistent judging, especially in the area of granularity. For example, the assessors accepted "treatment" as an answer to "What is acupuncture?" and "tennis" for "What is Wimbledon?", but disallowed "disorder" for "What is cerebral palsy?" and "gland" for "What is a thyroid" (see [42]). A combination of this problem, and a bug that crept into the system just prior to the evaluation, caused us to get an MRR of 0.26 and 0.29 in two runs (in the 50-byte task; the 250 byte task was dropped). We also submitted a run in which we combined our results with that of the IBM Natural Language Systems group in a voting algorithm (weighting our answers relative to theirs in a 3:2 ratio). This combined run had an MRR of 0.36, which was an improvement over the prior year but we slipped in rank to 10th out of 92 submitted runs.

5.1.4 TREC-2002

For TREC-2002, the task was made more difficult through (1) limiting submissions to single answers rather than the top 5, and (2) a requirement that the answer string be "exact", rather than a text fragment that con-

tained the answer. To address these challenges we made significant improvements to the Answer Selection module, which through parsing both the question and the hit-list passages, scored answers based on the strengths of syntactic relationships of terms, rather than their mere existence. We also began the task (still underway) of changing our system to perform question-specific processing when questions of certain classes were detected. One such specific process examined the question focus in "Where is X" type questions: if X can be identified as a specific geographic entity (such as city or country for example), then the set of allowable answer types would be restricted to containing entities (state/country or continent/ocean/world-region respectively).

We began to centralize our ontological resources behind a service layer, and initially used it for answering questions whose answers would be most naturally found in a list or a table - for example, questions about capitals or names of animal young. Due to the limited implementation of this mechanism at the time of the evaluation, it was unclear if it made any difference either way, but subsequent improvements have proven its utility.

The use of Cyc as part of our system was another addition whose benefit for the evaluation was limited due to the infancy of the implementation. We had observed in prior years that some of our candidate answers were ridiculous even though they were of the right type (e.g. an answer to "What is the diameter of the Earth?" was "4 ft"). We had undertaken a relationship with Cycorp under the AQUAINT program, and decided to use Cyc at first as a "sanity checker" to eliminate candidate answers that were clearly "insane". Our initial efforts in this direction were to have Cyc test that answers to numerical questions were within a reasonable range. Unfortunately, Cyc's coverage of such ranges at the time of the evaluation was insufficient to be of much help.

The final major change that we made for TREC-2002 was to extend the use of alternative corpora/answering agents. The first dimension explored here was to perform the search against alternative text indices, and perform answer selection against the combined set of passages. A particular set of passages was denoted as "primary" if a final answer could be selected from it, or as "support" if an answer found there could only boost the score of the same answer found in a primary passage. The primary corpus for this evaluation was the new AQUAINT corpus, while for support we used the old TREC QA corpus plus a subset of an encyclopedia. Our runs showed a 20% relative improvement in answers found, and about 15% in average precision. Using the passages from our sister SMT group instead of the encyclopedia we achieved a slight improvement in these figures. The second dimension that we explored was to use the final answers from the SMT group to recalculate the confidence in our answers, and hence their position in the submitted ranking. Our results showed an improvement in average precision of a further 10%.

The average precision measure, also called *Confidence Weighted Score*, introduced for this evaluation, was an attempt to grade systems for knowing what they know. Systems were to reorder their 500 responses according to confidence, and correct answers nearer the head of the list would be given a greater weight than those following. This measure combines both a system's intrinsic question answering ability with its confidence of its abilities; in [43] we separate out the latter contribution as *Ranking Ability*, for which one of our submissions scored second-best overall.

5.1.5 TREC-2003

Our work for TREC2003 centered primarily on three aspects of our system: (1) making engineering changes to enforce a more rigorous distinction between the different agents that we used, to allow easier switching on and off either by end-users or automatically depending on question type, (2) a heavier use of Cyc and other external structured knowledge sources, in particular tables of facts downloaded from the Web, to aid with questions about popular culture, and (3) a new agent to implement our QA-by-Dossier approach to

answering definitional questions. In the Dossier agent, the original question is transformed into a dozen or more subsidiary questions ("When was X born?", "What occupation did X have?" etc.), the answers from which are collected, filtered by a machine-learned threshold and returned as a dossier or profile of the subject of the original question.

5.2 Statistical Question Answering[7]

In TREC-9, IBM developed a statistical system for question answering (SQA)[44], [45]; this was the first fully data-driven TREC QA system. The system used a pipeline architecture similar to other systems performing question answering but used maximum entropy to model both the answer tagging (classifying the answer phrase) and answer selection (selecting the best answer chunk from a list of answers.) The system models the distribution $p(c|a,q)$, which attempts to measure the c, correctness or more traditionally the relevance, of the answer to the question. Further, we introduced a hidden variable representing the class of the answer, e, (answer tag/named entity) as follows

$$\begin{array}{ll} p(c|q,a) & = \sum_e p(c,e|q,a) \\ & = \sum_e p(c|e,q,a)p(e|q,a). \end{array} \tag{1}$$

These distributions are modeled using a maximum entropy formulation [46]. Human judgements of question answer pairs form the training data. $p(e|q,a)$ is the answer tag model which predicts from the question and proposed answers, the entity that both satisfy. The training data for the answer tag model is 13000 questions which have been annotated with 31 categories. In this model, we make the assumption that the entity being sought by the question is independent of the answer. The joint effect of the question and answer is modeled in the answer selection model.

$p(c|e,q,a)$ is the Answer Selection Model. Given the question, the answer and an entity as predicted by the answer tag model, we seek to model the correctness of this configuration. This model is tractable because we have judgement data on 892 questions from the TREC-8 and TREC-9 as well as 4000 trivia questions. In formulating this model, note that there is no shortage of negative data, although the most useful question-answer pairs are those that are only nearly correct but need to be rejected. In order to generate the nearly correct data, a basic question answering system (without the statistical answer selection model) is used to propose answer candidates. The output is then judged by humans and these judgements form the training data for the answer selection algorithm.

The question is first analyzed by the answer tag model and the top answer tag is used by the answer selection model. Simultaneously, the question is expanded using LCA [28] with an encyclopedia, and the top 1000 documents are retrieved. From these documents, the top 100 sentences are chosen that (a) maximize the question word match, (b) have the desired answer tag, (c) minimize the dispersion of question words, and (d) have similar syntatic structures as the question. From the top 100 sentences, named entity and parse chunks are extracted. The chunks are then ranked using the maximum entropy answer selection model and the best chunk is output as the answer.

Subsequent systems in TREC-10 and TREC-11 [47] explored richer features such as (a) machine translation based answer ranker, (b) whether a candidate answer occured on the web, and (c) answer patterns for question answering.

[7] Abraham Ittycheriah

6 Multimedia Retrieval

IBM Research groups have participated in the Trec tracks on Spoken Document Retrieval, and Video Retrieval. Their participation is outlined in this section.

6.1 Spoken Document Retrieval[8]

In the TREC-6 spoken document retrieval task, the documents to be retrieved were VOA broadcast recordings. IBM provided an automatic transcription of the broadcasts to other participants [48] in order to provide a standard baseline and to encourage participation by other sites that did not have their own speech recognition systems. We used a large vocabulary research variant of the IBM ViaVoice speech recognizer to produce automatic transcripts of the audio. The IBM Audio-Indexing system was a combination of this automatic transcription and a text-based retrieval system. The audio indexing system was also used as a platform to address the open vocabulary problem by developing a search-time phonetic fast match [49] that ran 2,400 times faster than the speaking rate. The intuition was that a significant fraction of news centers around newly-important words unknown to the speech recognizer, for example "Netanyahu" in the year of TREC-6. Unfortunately the TREC queries contained few out-of-vocabulary words, so their effect on retrieval performance could not be addressed. The text retrieval system was based on the system that we used for the ad hoc track.

6.2 Video Retrieval at TREC-10[9]

The TREC-10 conference saw the first running of the video retrieval track. The benchmark consisted of a corpus of approximately 11 hours of video and 74 query topics. The challenge was to build a system capable of automatically analyzing and indexing the video, and of retrieving results for the 74 semantic queries. Example topics included "retrieve video clips showing the lunar rover vehicle," "retrieve clips showing launch of the space shuttle," and "retrieve clips of beach scenes." The use of automatic speech recognition was not emphasized, rather, the track focussed on the use of content-based retrieval techniques.

The IBM Research team developed a system for automatic and interactive content-based retrieval of video using visual features and statistical models [50]. The system used automatic methods for shot boundary detection and key-frame selection [51]. It indexed the key-frames of the video shots using MPEG-7 visual descriptors based on color histograms, color composition, texture and edge histograms [52]. The automatic searches were computed by matching descriptors of query content to those of the target content. The system also used statistical models for classifying events (fire, smoke, launch), scenes (greenery, land, outdoors, rock, sand, sky, water), and objects (airplane, boat, rocket, vehicle, faces). The classifiers were used to generate labels and corresponding confidence scores for each shot. The features and models were then used together for answering interactive searches where the user constructed query/filter pipelines that cascaded content-based and model-based searches. This allowed integration of multiple searches using different methods for each topic, for example, to retrieve "shots that have similar color to this image, have label 'outdoors' and show a 'boat."'

The overall results showed that the content-based/model-based approach performed relatively well compared to other systems. The team explored automatic speech recognition and text indexing as a baseline. In some cases the speech provided better results, for example, to retrieve "clips that deal with floods." In other cases, the content-based retrieval was better, for example, to retrieve "shots showing grasslands." In

[8] J. Scott McCarley
[9] John R. Smith

two cases, the best result was obtained by combining speech and content-based/model-based methods, for example, to retrieve "clips of Perseus high altitude plane." The results show promise in particular for the approach based on statistical modeling for video content classification. The overall results show that improvements are still needed in terms of absolute retrieval effectiveness in order to deploy usable systems. By creating the video retrieval benchmark, NIST is helping to accelerate the necessary technology development.

7 Conclusions

The work by groups in IBM Research that has been surveyed here shows some of the variety of scientific problems that were tackled, and perhaps gives some insights into the variety of motivations for participating in TREC as a way to tackle them. TREC has proven to be a valuable forum in which IBM Research has contributed to improved understanding of search, while at the same time the insights obtained by participating in TREC have helped to improve IBM's products and services. This fruitful association looks likely to continue: TREC continues to play a central role in the research agendas of several groups, and Research is a participant in new tracks, such as Genomics.

8 Acknowledgements

This work was supported in part by the Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Program under contract number MDA904-01-C-0988.

References

[1] Yoelle 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.

[2] Y.S. Maarek, D.M. Berry, and G.E. Kaiser. An information retrieval approach for automatically constructing software libraries. *Transactions on Software Engineering*, 17(8), August 1991.

[3] Eric W. Brown and Herb A. Chong. The GURU system in TREC-6. In E. Voorhees and D.K. Harmon, editors, *The Sixth Text Retrieval Conference (TREC-6)*, number NIST Special Publication 500-240, pages 535–540, Gaithersburg, MD., 1998. Information Technology Laboratory, National Institute of Standards and Technology.

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

[5] Alan 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.

[6] Chris Buckley and Alan 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.

[7] Eric 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.

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

[9] Anthony Tomasic and Hector 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.

[10] Brendon Cahoon and Kathryn 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, August 1996.

[11] D. Carmel, E. Amitay, M. Herscovici, Y. S. Maarek, Y. Petruschka, and A. Soffer. Juru at TREC 10 - Experiments with Index Pruning. In *Proceeding of Tenth Text REtrieval Conference (TREC-10)*. National Institute of Standards and Technology (NIST), 2001.

[12] C. Buckley, A. Singhal, M. Mitra, and G. Salton. New retrieval approaches using SMART: TREC 4. In *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pages 25–48, Gaithersberg, Maryland, November 1996.

[13] David Carmel, Doron Cohen, Ronald Fagin, Eitan Farchi, Michael Herscovici, Yoelle S. Maarek, and Aya Soffer. Static index pruning for information retrieval systems. In *Proceedings of the 24th International ACM SIGIR Conference*, pages 41 – 50, New Orleans, LA., September 2001.

[14] E. Amitay, D. Carmel, A. Darlow, R. Lempel, and A. Soffer. Topic Distillation with Knowledge Agents. In *Proceeding of the eleventh Text REtrieval Conference (TREC-2002)*. National Institute of Standards and Technology (NIST), 2002.

[15] Y. Aridor, D. Carmel, R. Lempel, Y. Maarek, and A. Soffer. Knowledge agents on the web. In *Proceedings of the 4th International Workshop on Cooperative Information Agents, CIA 2000, (Lecture Notes in Artificial Intelligence LNAI 1860)*, pages 15–26, Boston, MA, July 2000. Springer.

[16] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, volume 25-27, pages 668 – 677, January 1998.

[17] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, Stanford, CA, November 1999. http://dbpubs.stanford.edu/pub/1999-66.

[18] Tapas Kanungo and Jason Y. Zien. Integrating link structure and content information for ranking web documents. In Ellen Voorhees and Donna Harman, editors, *The Tenth Text REtrieval Conference (TREC-10)*, page 237, Gaithersburg, MD, 2002. National Institute of Standards and Technology.

[19] Soumen Chakrabarti, Byron Dom, David Gibson, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Spectral filtering for resource discovery. In *Proceedings of the SIGIR 98 Workshop on Hypertext Information Retrieval for the Web*, August 1998.

[20] J. Allan, M. Connell, W. B. Croft, F. F Feng, D. Fisher, and X. Li. INQUERY and TREC-9. In *Proceedings of TREC-9*, 2000.

[21] E. M. Voorhees and D. K. Harman, editors. Department of Commerce, National Institute of Standards and Technology, November 1997.

[22] Birgit Schmidt-Wesche, Robert Mack, Christian Lenz Cesar, and David VanEsselstyn. IBM search UI prototype evaluation at the interactive track of TREC-6. In E. Voorhees and D.K. Harmon, editors, *The Sixth Text Retrieval Conference (TREC-6)*, number NIST Special Publication 500-240, pages 517–534, Gaithersburg, MD., 1998. Information Technology Laboratory, National Institute of Standards and Technology.

[23] Ernest P. Chan, Santiago Garcia, and Salim Roukos. Trec-5 ad hoc retrieval using k nearest-neighbors re-scoring. In *Proceedings of the Fifth Text REtrieval Conference*, 1997.

[24] Martin Franz, J. Scott McCarley, and Salim Roukos. Ad hoc and multilingual information retrieval at IBM. In *Proceedings of the Seventh Text REtrieval Conference*, 1999.

[25] Martin Franz, J. Scott McCarley, and R. Todd Ward. Ad hoc, cross-language and spoken document information retrieval at IBM. In *Proceedings of the Eighth Text REtrieval Conference*, 2000.

[26] S.E. Robertson, S. Walker, S. Jones, MM. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *Proceedings of the Third Text REtrieval Conference*, 1995.

[27] Ernest P. Chan, Santiago Garcia, and Salim Roukos. Probabilistic modeling for information retrieval with unsupervised training data. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, 1998.

[28] J. Xu and W.B. Croft. Query expansion using local global document analysis. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4–11, 1996.

[29] P. Brown, S. Della Pietra, V. Della Pietra, and R. Mercer. The mathematics of statistical machine translation : Parameter estimation. *Computational Linguistics*, 19:263–311, 1993.

[30] J.S. McCarley and S. Roukos. Fast document translation for cross-language information retrieval. In D. Farwell., E. Hovy, and L. Gerber, editors, *Machine Translation and the Information Soup*, page 150, 1998.

[31] J.S. McCarley. Should we translate the documents or the queries in cross-language information retrieval? In *37th Annual Meeting of the Association for Computational Linguistics*, 1999.

[32] Martin Franz, J. Scott McCarley, and Wei-Jing Zhu. English-chinese information retrieval at IBM. In *Proceedings of the Ninth Text REtrieval Conference*, 2001.

[33] Jinxi Xu, Alexander Fraser, and Ralph Weischedel. Trec 2001 cross-lingual retrieval at bbn. In *Proceedings of the Tenth Text REtrieval Conference*, 2001.

[34] Martin Franz and J. Scott McCarley. Arabic information retrieval at IBM. In *Proceedings of the Eleventh Text REtrieval Conference*, 2003.

[35] N. Wacholder, Y. Ravin, and M. Choi. Disambiguation of proper names in text. In *Proceedings of Fifth Conference on Applied Natural Language Processing*, pages 202–208, March 1997.

[36] R. Byrd and Y. Ravin. Identifying and extracting relations in text. In *Proceedings of NLDB 99*, Klagenfurt, Austria., 1999.

[37] J. Prager, E. Brown, A. Coden, and D. Radev. Question answering by predictive annotation. In *Proceedings SIGIR 2000*, pages 184–191, Athens, Greece, 2000.

[38] D. Radev, J. Prager, and V. Samn. Ranking suspected answers to natural language questions using predictive annotation. In *6th Conference on Applied Natural Language Processing*, pages 150–157, Seattle, May 2000.

[39] J. Prager, D. Radev, E. Brown, A. Coden, and V. Samn. The use of predictive annotation for question answering in trec. In Ellen Voorhees and Donna Harman, editors, *The Eighth Text REtrieval Conference (TREC-8)*, Gaithersburg, MD, 2000. National Institute of Standards and Technology.

[40] J. Prager, E. Brown, D. Radev, and Krzysztof Czuba. One search engine or two for question-answering. In Ellen Voorhees and Donna Harman, editors, *The Ninth Text REtrieval Conference (TREC-9)*, Gaithersburg, MD, 2001. National Institute of Standards and Technology.

[41] John Prager, Dragomir Radev, and Krzysztof Czuba. Answering what-is questions by virtual annotation. In *Proceedings Human Language Technology conference*, San Diego, CA, 2001.

[42] John Prager, Jennifer Chu-Carroll, and Krzysztof Czuba. Use of wordnet hypernyms for answering what-is questions. In Ellen Voorhees and Donna Harman, editors, *The Tenth Text REtrieval Conference (TREC-10)*, Gaithersburg, MD, 2002. National Institute of Standards and Technology.

[43] Jennifer Chu-Carroll, John Prager, Krzysztof Czuba Christopher Welty, and David Ferrucci. A multi-strategy and multi-source approach question answering. In Ellen Voorhees and Donna Harman, editors, *The Eleventh Text REtrieval Conference (TREC-10)*, pages 281–288, Gaithersburg, MD, 2003. National Institute of Standards and Technology.

[44] Abraham Ittycheriah, Martin Franz, Wei-Jing Zhu, Adwait Ratnaparki, and Richard Mammone. IBM's statistical question answering system. *TREC-9 Proceedings*, pages 60–65, 2000.

[45] Abraham Ittycheriah, Martin Franz, Wei-Jing Zhu, Adwait Ratnaparki, and Richard Mammone. Question answering using maximum entropy components. *The Second Meeting of the North American Chapter of the Association of Computational Linguistics, Pittsburgh, PA*, pages 33–39, 2001.

[46] Adam L. Berger, Vincent Della Pietra, and Stephen Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[47] Abraham Ittycheriah. *Trainable Question Answering Systems*. PhD Thesis, Department of Electrical and Computer Engineering, Rutgers - The State University of New Jersey, 2001.

[48] Satya Dharanipragada, Martin Franz, and Salim Roukos. Audio indexing for broadcast news. In *Proceedings of the Seventh Text REtrieval Conference*, 1999.

[49] Satya Dharanipragada and Salim Roukos. A multi-stage algorithm for spotting new words in speech. *IEEE Transactions on Speech and Audio Processing*, 10:542, 2002.

[50] J. R. Smith, S. Basu, C.-Y. Lin, M. Naphade, and B. Tseng. Integrating features, models, and semantics for content-based retrieval. In *Proc. Multimedia Content-based Indexing and Retrieval (MMCBIR) workshop*, Rocquencourt, FR, September 2001.

[51] *IBM CueVideo Toolkit Version 2.1, http://www.almaden.ibm.com/cs/cuevideo/.* Download at http://www.ibm.com/alphaworks.

[52] J. R. Smith. MPEG-7 standard for multimedia databases. In *ACM Intl. Conf. on Management of Data (SIGMOD)*, Santa Barbara, CA, May 2001. Tutorial.