Building Effective Queries In Natural Language Information Retrieval

Tomek Strzalkowski[1], Fang Lin[1], Jose Perez-Carballo[2] and Jin Wang[1]

[1]GE Corporate Research & Development

1 Research Circle, Niskayuna, NY 12309

[2]School of Communication, Information and Library Studies, Rutgers University

4 Huntington Street, New Brunswick, NJ 08903

ABSTRACT

In this paper we report on our natural language information retrieval (**NLIR**) project as related to the recently concluded 5th Text Retrieval Conference (TREC-5). The main thrust of this project is to use natural language processing techniques to enhance the effectiveness of full-text document retrieval. One of our goals was to demonstrate that robust if relatively shallow NLP can help to derive a better representation of text documents for statistical search. Recently, we have turned our attention away from text representation issues and more towards query development problems. While our NLIR system still performs extensive natural language processing in order to extract phrasal and other indexing terms, our focus has shifted to the problems of building effective search queries. Specifically, we are interested in query construction that uses words, sentences, and entire passages to expand initial topic specifications in an attempt to cover their various angles, aspects and contexts. Based on our earlier results indicating that NLP is more effective with long, descriptive queries, we allowed for long passages from related documents to be liberally imported into the queries. This method appears to have produced a dramatic improvement in the performance of two different statistical search engines that we tested (Cornell's SMART and NIST's Prise) boosting the average precision by at least 40%. In this paper we discuss both manual and automatic procedures for query expansion within a new stream-based information retrieval model.

1. INTRODUCTION

A typical (full-text) information retrieval (IR) task is to select documents from a database in response to a user's query, and rank these documents according to relevance. This has been usually accomplished using statistical methods (often coupled with manual encoding) that (a) select terms (words, phrases, and other units) from documents that are deemed to best represent their content, and (b) create an inverted index file (or files) that provide an easy access to documents containing these terms. A subsequent search process will attempt to match preprocessed user queries against term-based representations of documents in each case determining a degree of relevance between the two which depends upon the number and types of matching terms. Although many sophisticated search and matching methods are available, the crucial problem remains to be that of an adequate representation of content for both the documents and the queries.

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

The simplest word-based representations of content, while relatively better understood, are usually inadequate since single words are rarely specific enough for accurate discrimination, and their grouping is often accidental. A better method is to identify groups of words

that create meaningful phrases, especially if these phrases denote important concepts in the database domain. For example, "joint venture" is an important term in the Wall Street Journal (WSJ henceforth) database, while neither "joint" nor "venture" are important by themselves.

There are a number of ways to obtain "phrases" from text. These include generating simple collocations, statistically validated N-grams, part-of-speech tagged sequences, syntactic structures, and even semantic concepts. Some of these techniques are aimed primarily at identifying multi-word terms that have come to function like ordinary words, for example "white collar" or "electric car", and capturing other co-occurrence idiosyncrasies associated with certain types of texts. This simple approach has proven quite effective for some systems, for example the Cornell group reported (Buckley et al., 1995) that adding simple bigram collocations to the list of available terms can increase retrieval precision by as much as 10%.

Other more advanced techniques of phrase extraction, including extended N-grams and syntactic parsing, attempt to uncover "concepts", which would capture underlying semantic uniformity across various surface forms of expression. Syntactic phrases, for example, appear reasonable indicators of content, arguably better than proximity-based phrases, since they can adequately deal with word order changes and other structural variations (e.g., "college junior" vs. "junior in college" vs. "junior college"). A subsequent regularization process, where alternative structures are reduced to a "normal form", helps to achieve a desired uniformity, for example, "college+junior" will represent a college for juniors, while "junior+college" will represent a junior in a college. A more radical normalization would have also "verb object", "noun rel-clause", etc. converted into collections of such ordered pairs. This head+modifier normalization has been used in our system, and is further described in this paper. It has to be noted, however, that the use of full-scale syntactic analysis is severely pushing the limits of practicality of an information retrieval system because of the increased demand for computing power and storage. At the same time, while the gain in recall and precision has not been negligible (we recorded 10-20% increases in precision), no dramatic breakthrough has occurred either.[1]

1. Currently, the state-of-the art statistical and probabilistic IR system perform at about 20-40% precision range for arbitrary ad-hoc retrieval tasks.

This state of affairs has prompted us take a closer look at the term selection and representation process. Our earlier experiments demonstrated that an improved weighting scheme for compound terms, including phrases and proper names, leads to an overall gain in retrieval accuracy. The fundamental problem, however, remained to be the system's inability to recognize, in the documents searched, the presence or absence of the concepts or topics that the query is asking for. The main reason for this was, we noted, the limited amount of information that the queries could convey on various aspects of topics they represent. Therefore, we started experimenting with manual and automatic query building techniques. The purpose was to devise a method for full-text query expansion that would allow for creating exhaustive search queries such that: (1) the performance of any system using these queries would be significantly better than when the system is run using the original topics, and (2) the method could be eventually automated or semi-automated so as to be useful to a non-expert user. Our preliminary results from TREC-5 evaluations show that this approach is indeed very effective.

In the rest of this paper we describe the overall organization of our TREC-5 system, and then discuss some experiments that we performed and their results, as well as our future research plans.

2. STREAM-BASED INFORMATION RETRIEVAL MODEL

Our NLIR system encompasses several statistical and natural language processing (NLP) techniques for robust text analysis. These has been organized together into a "stream model" in which alternative methods of document indexing are strung together to perform in parallel. Stream indexes are built using a mixture of different indexing approaches, term extracting and weighting strategies, even different search engines. The final results are produced by merging ranked lists of documents obtained from searching all stream indexes with appropriately preprocessed queries, i.e., phrases for phrase stream, names for names stream, etc. The merging process weights contributions from each stream using a combination that was found the most effective in training runs. This allows for an easy combination of alternative retrieval and routing methods, creating a meta-search strategy which maximizes the contribution of each stream. Both Cornell's SMART version 11, and NIST's Prise search engines were used as basic engines.[2]

Our NLIR system employs a suite of advanced natural language processing techniques in order to assist the sta-

tistical retrieval engine in selecting appropriate indexing terms for documents in the database, and to assign them semantically validated weights. The following term extraction methods have been used; they correspond to the indexing streams in our system.

1. Eliminate stopwords: original text words minus certain no-content words are used to index documents.
2. Morphological stemming: we normalize across morphological word variants (e.g., "proliferation", "proliferate", "proliferating") using a lexicon-based stemmer.
3. Phrase extraction: we use various shallow text processing techniques, such as part-of-speech tagging, phrase boundary detection, and word co-occurrence metrics to identify stable strings of words, such as "joint venture".
4. Phrase normalization: we identify "head+modifier" pairs in order to normalize across syntactic variants such as "weapon proliferation", "proliferation of weapons", "proliferate weapons", etc. into "weapon+proliferate".
5. Proper names: we identify proper names for indexing, including people names and titles, location names, organization names, etc.

Among the advantages of the stream architecture we may include the following:

- stream organization makes it easier to compare the contributions of different indexing features or representations. For example, it is easier to design experiments which allow us to decide if a certain representation adds information which is not contributed by other streams.
- it provides a convenient testbed to experiment with algorithms designed to merge the results obtained using different IR engines and/or techniques.
- it becomes easier to fine-tune the system in order to obtain optimum performance
- it allows us to use any combination of IR engines without having to modify their code at all.

While our stream architecture may be unique among IR systems, the idea of combining evidence from multiple sources has been around for some time. Several researchers have noticed in the past that different systems may have similar performance but retrieve different documents, thus suggesting that they may complement one another. It has been reported that the use of different sources of evidence increases the performance of a system (see for example, Callan et al., 1995; Fox et al.,1993; Saracevic & Kantor, 1988).

3. STREAMS USED IN NLIR SYSTEM

3.1 Head-Modifier Pairs Stream

Our most linguistically advanced stream is the *head+modifier pairs* stream. In this stream, documents are reduced to collections of word pairs derived via syntactic analysis of text followed by a normalization process intended to capture semantic uniformity across a variety of surface forms, e.g., "information retrieval", "retrieval of information", "retrieve more information", "information that is retrieved", etc. are all reduced to "retrieve+information" pair, where "retrieve" is a head or operator, and "information" is a modifier or argument.

The pairs stream is derived through a sequence of processing steps that include:

- Part-of-speech tagging
- Lexicon-based word normalization (extended "stemming")
- Syntactic analysis with the TTP parser (cf. Strzalkowski & Scheyen, 1996)
- Extraction of head+modifier pairs
- Corpus-based decomposition/disambiguation of long noun phrases.

Syntactic phrases extracted from TTP parse trees are head-modifier pairs. The head in such a pair is a central element of a phrase (main verb, main noun, etc.), while the modifier is one of the adjunct arguments of the head. It should be noted that the parser's output is a predicate-argument structure centered around main elements of various phrases. The following types of pairs are considered: (1) a head noun and its left adjective or noun adjunct, (2) a head noun and the head of its right adjunct, (3) the main verb of a clause and the head of its object phrase, and (4) the head of the subject phrase and the main verb. These types of pairs account for most of the syntactic variants for relating two words (or simple phrases) into pairs carrying compatible semantic content. This also gives the pair-based representation sufficient flexibility to effectively capture content elements even in complex expressions. Long, complex phrases are similarly decomposed into collections of pairs, using corpus statistics to resolve structural ambiguities.

2. SMART version 11 is freely available, unlike the more advanced version 12.

3.2 Linguistic Phrase Stream

We used a regular expression pattern matcher on the part-of-speech tagged text to extract ***noun groups*** and *proper noun sequences*. The major rules we used are:

1. a sequence of modifiers (vbn|vbg|jj) followed by at least one noun, such as: "cryonic suspend", "air traffic control system";
2. proper noun(s) modifying a noun, such as: "u.s. citizen", "china trade";
3. proper noun(s) (might contain '&'), such as: "warren commission", "national air traffic controller".

In these experiments, the length of phrases was limited to maximum 7 words.

3.3 Name Stream

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

3.4 Other Streams used

The **stems stream** is the simplest, yet, it turns out, the most effective of all streams, a backbone in our multi-stream model. It consists of stemmed non-stop single-word tokens (plus hyphenated phrases). Our early experiments with multi-stream indexing using SMART suggested that the most effective weighting of this stream is lnc.ltc, which yields the best average precision, whereas lnc.ntc slightly sacrifices the average precision, but gives better recall (see Buckley, 1993).

We used also a **plain text stream**. This stream was obtained by indexing the text of the documents "as is" without stemming or any other processing and running the unprocessed text of the queries against that index.

Finally, some experiments involved the **fragments stream.** This was the result of spliting the documents of the STEM stream into fragments of constant length (1024 characters) and indexing each fragment as if it were a different document. The queries used with this stream were the usual stem queries. For each query, the resulting ranking was filtered to keep, for each document, the highest score obtained by the fragments of that document.

Table 1 shows relative performance of each stream tested for this evaluation. Note that the standard stemmed-word representation (stems stream) is still the most efficient one, but linguistic processing becomes more important in longer queries. In this evaluation, the short queries are one-sentence search directives such as the following:

What steps are being taken by governmental or even private entities world-wide to stop the smuggling of aliens.

The long queries, on the other hand, contain substantially more text as the result of full-text expansion described in section 5 below.

TABLE 1. How different streams perform relative to one another (11-pt avg. Prec)

STREAM	short queries	long queries
Stems	0.1682	0.2626
Phrases	0.1233	0.2365
H+M Pairs	0.0755	0.2040
Names	0.0844	0.0608

4. STREAM MERGING STRATEGY

The results obtained from different streams are list of documents ranked in order of relevance: the higher the rank of a retrieved document, the more relevant it is presumed to be. In order to obtain the final retrieval result, ranking lists obtained from each stream have to be combined together by a process known as merging or fusion. The final ranking is derived by calculating the combined relevance scores for all retrieved documents. The following are the primary factors affecting this process:

1. document relevancy scores from each stream
2. retrieval precision distribution estimates within ranks from various streams, e.g., projected precision between ranks 10 and 20, etc.;
3. the overall effectiveness of each stream (e.g. measured as average precision on training data)
4. the number of streams that retrieve a particular document, and
5. the ranks of this document within each stream.

Generally, a more effective stream will more effect on shaping the final ranking. A document which is retrieved at a high rank from such a stream is more likely to end up ranked high in the final result. In addition, the performance of each stream within a specific range of ranks is taken into account. For example, if phrases stream tends to pack relevant documents into top 10-20 retrieved documents (but not so much into 1-10) we would give premium weights to the documents found in this region of phrase-based ranking, etc. Table 2 gives some additional data on the effectiveness of stream merging. Further details are available in a TREC conference article.

TABLE 2. Precision improvements over stems-only retrieval

Streams merged	short queries %change	long queries %change
All streams	+5.4	+20.94
Stems+Phrases+Pairs	+6.6	+22.85
Stems+Phrases	+7.0	+24.94
Stems+Pairs	+2.2	+15.27
Stems+Names	+0.6	+2.59

Note that again, long text queries benefit more from linguistic processing.

5. QUERY EXPANSION EXPERIMENTS

5.1 Why query expansion?

The purpose of query expansion in information retrieval is to make the user query resemble more closely the documents it is expected to retrieve. This includes both content, as well as some other aspects such as composition, style, language type, etc. If the query is indeed made to resemble a "typical" relevant document, then suddenly everything about this query becomes a valid search criterion: words, collocations, phrases, various relationships, etc. Unfortunately, an average search query does not look anything like this, most of the time. It is more likely to be a statement specifying the semantic criteria of relevance. This means that except for the semantic or conceptual resemblance (which we cannot model very well as yet) much of the appearance of the query (which we can model reasonably well) may be, and often is, quite misleading for search purposes. Where can we get the right queries?

In today's information retrieval systems, query expansion usually pertains content and typically is limited to adding, deleting or re-weighting of terms. For example, content terms from documents judged relevant are added to the query while weights of all terms are adjusted in order to reflect the relevance information. Thus, terms occurring predominantly in relevant documents will have their weights increased, while those occurring mostly in non-relevant documents will have their weights decreased. This process can be performed automatically using a relevance feedback method, e.g., Roccio's (1971), with the relevance information either supplied manually by the user (Harman, 1988), or otherwise guessed, e.g. by assuming top 10 documents relevant, etc. (Buckley, et al., 1995). A serious problem with this content-term expansion is its limited ability to capture and represent many important aspects of what makes some documents relevant to the query, including particular term co-occurrence patterns, and other hard-to-measure text features, such as discourse structure or stylistics. Additionally, relevance-feedback expansion depends on the inherently partial relevance information, which is normally unavailable, or unreliable.

Other types of query expansions, including general purpose thesauri or lexical databases (e.g., Wordnet) have been found generally unsuccessful in information retrieval (cf. Voorhees & Hou, 1993; Voorhees, 1994)

An alternative to term-only expansion is a full-text expansion which we tried for the first time in TREC-5. In our approach, queries are expanded by pasting in entire sentences, paragraphs, and other sequences directly from *any* text document. To make this process efficient, we first perform a search with the original, unexpanded queries (short queries), and then use top N (10, 20) returned documents for query expansion. These documents are not judged for relevancy, nor assumed relevant; instead, they are scanned for passages that contain concepts referred to in the query. Expansion material can be found in both relevant and non-relevant documents, benefitting the final query all the same. In fact, the presence of such text in otherwise non-relevant documents underscores the inherent limitations of distribution-based term reweighting used in relevance feed-

back. Subject to some further "fitness criteria", these expansion passages are then imported verbatim into the query. The resulting expanded queries undergo the usual text processing steps, before the search is run again.

Full-text expansion can be accomplished manually, as we did initially to test feasibility of this approach, or automatically, as we tried in later with promising results. We first describe the manual process focussing on guidelines set forth in such a way as to minimize and streamline human effort, and lay the ground for eventual automation. We then describe our first attempt at automated expansion, and discuss the results from both.

The initial evaluations indicate that queries expanded manually following the prescribed guidelines are improving the system's performance (precision and recall) by as much as 40%. This appear to be true not only for our own system, but also for other systems: we asked other groups participating in TREC-5 to run search using our expanded queries, and they reported nearly identical improvements. At this time, automatic text expansion produces less effective queries than manual expansion, primarily due to a relatively unsophisticated mechanism used to identify and match concepts in the queries.

5.2 Guidelines for manual query expansion

We have adopted the following guidelines for query expansion. They were constructed to observe realistic limits of the manual process, and to prepare ground for eventual automation.

1. NLIR retrieval is run using the 50 original "short" queries.
2. Top 10 documentss retrieved by each query are retained for expansion. We obtain 50 expansion sub-collections, one per query.
3. Each query is manually expanded using phrases, sentences, and entire passages found in any of the documents from this query's expansion subcollection. Text can both added and deleted, but care is taken to assure that the final query has the same format as the original, and that all expressions added are well-formed English strings, though not necessarily well-formed sentences. **A limit of 30 minutes per query in a single block of time is observed.**
4. Expanded queries are sent through all text processing steps necessary to run the queries against multiple stream indexes.
5. Rankings from all streams are merged into the final result.

There are two central decision making points that affect the outcome of the query expansion process following the above guidelines. The first point is how to locate text passages that are worth looking at -- it is impractical, if not downright impossible to read all 10 documents, some quite long, in under 30 minutes. The second point is to actually decide whether to include a given passage, or a portion thereof, in the query. To facilitate passage spotting, we used simple word search, using key concepts from the query to scan down document text. Each time a match was found, the text around (usually the paragraph containing it) was read, and if found "fit", imported into the query. We experimented also with various "pruning" criteria: passages could be either imported verbatim into the query, or they could be "pruned" of "obviously" undesirable noise terms. In evaluating the expansion effects on query-by-query basis we have later found that the most liberal expansion mode with no pruning was in fact the most effective. This would suggest that relatively self-contained text passages, such as paragraphs, provide a balanced representation of content, that cannot be easily approximated by selecting only some words.

5.3 Automatic Query Expansion

Queries obtained through the full-text manual expansion proved to be overwhelmingly better than the original search queries, providing as much as 40% precision gain. These results were sufficiently encouraging to motivate us to investigate ways of performing such expansions automatically.

One way to approximate the manual text selection process, we reasoned, was to focus on those text passages that refer to some key concepts identified in the query, for example, "alien smuggling" for query 252 below.

The key concepts (for now limited to simple noun groups) were identified by either their pivotal location within the query (in the Title field), or by their repeated occurrences within the query Description and Narrative fields. As in the manual process, we run a "short" query retrieval, this time retaining 100 top documents retrieved by each query. An automated process then scans these 100 documents for all paragraphs which contain occurrences, including some variants, of any of the key concepts identified in the original query. The paragraphs are subsequently pasted verbatim into the query. The original portion of the query may be saved in a special field to allow differential weighting. Finally, the expanded queries were run to produce the final result.

The above, clearly simplistic technique has produced some interesting results. Out of the fifty queries we tested, 34 has undergone the expansion. Among these 34 queries, we noted precision gains in 13, precision loss in 18 queries, with 3 more basically unchanged. However, for these queries where the improvement did occur it was very substantial indeed: the average gain was 754% in 11-pt precision, while the average loss (for the queries that lost precision) was only 140%. Overall, we still can see a 7% improvement on all 50 queries (vs. 40%+ when manual expansion is used).

Our experiments show that selecting the right paragraphs from documents to expand the queries can dramatically improve the performance of a text retrieval system. This process can be automated, however, the challenge is to devise more precise automatic means of "paragraph picking".

6. SUMMARY OF RESULTS

In this section we summarize the results obtained from query expansion and other related experiments.

An automatic run means that there was no human intervention in the process at any time. A manual run means that some human processing was done to the queries, and possibly multiple test runs were made to improve the queries. A **short query** is derived using only one section of a TREC-5 topic, namely the DESCRIPTION field. A **full query** is derived from any or all fields in the original topic. A **long query** is obtained through our full-text expansion method (manual, or automatic). An example TREC-5 query is show below; note that the Description field is what one may reasonably expect to be an initial search query, while Narrative provides some further explanation of what relevant material may look like. The Topic field provides a single concept of interest to the searcher; it was not permitted in the short queries.

<top>

<num> Number: 252

<title> Topic: Combating Alien Smuggling

<desc> Description:

What steps are being taken by governmental or even private entities world-wide to stop the smuggling of aliens.

<narr> Narrative:

To be relevant, a document must describe an effort being made (other than routine border patrols) in any country of the world to prevent the illegal penetration of aliens across borders.

</top>

Table 3 summarizes selected runs performed with our NLIR system on TREC-5 database using queries 251 through 300. Table 4 gives the performance of Cornell's (now Sabir Inc.) SMART system version 12, using advanced Lnu.ltu term weighting scheme, and query expansion through automatic relevance feedback (rel.fbk), on the same database and with the same queries. Sabir used our long queries to obtain long query run. Note the consistently large improvements in retrieval precision attributed to the expanded queries.

TABLE 3. Precision improvement in NLIR system

PREC.	short queries	full queries	long queries auto.	long queries man.
11pt. avg	0.1478	0.2078	0.2220	0.3176
%change		+41.0	+50.0	+115.0
@10 docs	0.1578	0.2044	0.2089	0.3156
%change		+30.0	+32.0	+100.0
@100 doc	0.0544	0.0696	0.0709	0.0998
%change		+28.0	+30.0	+83.0
Recall	0.59	0.65	0.64	0.77
%change		+10.0	+8.5	+31.0

TABLE 4. Results for Cornell's SMART

PREC,	short queries	full queries	full queries rel.fbk	long queries man.
11pt.avg	0.1499	0.2142	0.2416	0.2983
%change		+43.0	+62.0	+99.0
@5 docs	0.2178	0.2889	0.2756	0.3600
%change		+33.0	+27.0	+65.0
@100 docs	0.0578	0.0709	0.0771	0.0904
%change		+23.0	+33.0	+56.0
Recall	0.58	0.64	0.70	0.73
@change		+10.0	+21.0	+26.0

7. CONCLUSIONS

We presented in some detail our natural language information retrieval system consisting of an advanced NLP module and a `pure' statistical core engine. While many problems remain to be resolved, including the question of adequacy of term-based representation of document content, we attempted to demonstrate that the architecture described here is nonetheless viable. In particular, we demonstrated that natural language processing can now be done on a fairly large scale and that its speed and robustness have improved to the point where it can be applied to real IR problems.

The main observation to make is that natural language processing is not as effective as we have once hoped to obtain better indexing and better term representations of queries. Using linguistic terms, such as phrases, head-modifier pairs, names, or even simple concepts does help to improve retrieval precision, but the gains remained quite modest. On the other hand, full text query expansion works remarkably well. Our main effort in the immediate future will be to explore ways to achieve at least partial automation of this process. An initial experiment in this direction has been performed as part of NLP Track (genlp3 run), and the results are encouraging.

ACKNOWLEDGEMENTS. We would like to thank Donna Harman of NIST for making her PRISE system available to us since the beginning of TREC. Will Rogers and Paul Over provided valuable assistance in installing updated versions of PRISE. We would also like to thank Ralph Weischedel for providing the BBN's part of speech tagger. We acknowledge the following members of our TREC-5 team who participated in the query expansion experiments: Louise Guthrie, Jussi Karlgren, Jim Leistensnider, Troy Straszheim, and Jon Wilding. This paper is based upon work supported in part by the Advanced Research Projects Agency under Tipster Phase-2 Contract 94-FI57900-000, Tipster Phase-3 Contract 97-FI56800-000, and the National Science Foundation under Grant IRI-93-02615.

REFERENCES

Buckley, Chris. 1993. "The Importance of Proper Weighting Methods." Proc. of ARPA's Human Language Technology Workshop. pp. 349-352.

Buckley, Chris, Amit Singhal, Mandar Mitra, Gerard Salton. 1995. "New Retrieval Approaches Using SMART: TREC 4". Proceedings of the Third Text REtrieval Conference (TREC-4), NIST Special Publ.

Callan, James, Zhihong Lu, and Bruce Croft. 1995. Searching Distributed Collections with Inference Networks." Proceedings of SIGIR-95. pp. 21-28.

Fox, Ed, Prabhakar Kushik, Joseph Shaw, Russell Modlin and Durgesh Rao. 1993. "Combining Evidence from Multiple Searches.". Proc. of First Text Retrieval Conference (TREC-1). NIST Spec. Publ. 500-207. pp. 319-328.

Harman, Donna. 1988. "Towards interactive query expansion." Proceedings of ACM SIGIR-88, pp. 321-331.

Roccio, J. 1971. "Relevance Feedback in Information Retrieval." In G. Salton (ed), "The SMART Retrieval System. Prentice-Hall, pp. 313-323.

Saracevic, T., Kantor, P. 1988. "A Study of Information Seeking and Retrieving. III. Searchers, Searches, and Overlap". Journal of the American Society for Information Science. 39(3):197-216.

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

Strzalkowski, Tomek, Jose Perez-Carballo and Mihnea Marinescu. 1995. "Natural Language Information Retirieval: TREC-3 Report." Proceedings of the Third Text REtrieval Conference (TREC-3), NIST Special Publication 500-225, pp. 39-53.

Strzalkowski, Tomek, Jose Perez-Carballo and Mihnea Marinescu. 1996. "Natural Language Information Retirieval: TREC-4 Report." Proceedings of the Third Text REtrieval Conference (TREC-4), NIST Special Publ.

Strzalkowski, Tomek. 1995. "Natural Language Information Retrieval" Information Processing and Management, Vol. 31, No. 3, pp. 397-417. Pergamon/Elsevier.

Strzalkowski, Tomek, and Peter Scheyen. 1996. "An Evaluation of TTP Parser: a preliminary report." In H. Bunt, M. Tomita (eds), Recent Advances in Parsing Technology, Kluwer Academic Publishers, pp. 201-220.

Voorhees, Ellen. 1994. "Query Expansion Using Lexical-Semantic Relations." Proc. of SIGIR-94, pp. 61-70.

Voorhees, Ellen, Yuan-Wang Hou. 1993. "Vector Expansion in a Large Collection." Proc of First Text Retrieval Conference (TREC-1). NIST Spec. Pub. 500-207. pp. 343-351.