Enhancing Detection through Linguistic Indexing and Topic Expansion

Tomek Strzalkowski, Gees C. Stein and G. Bowden Wise
GE Corporate Research & Development
1 Research Circle
Niskayuna, NY 12309
strzalkowski@crd.ge.com

Abstract

Natural language processing techniques may hold a tremendous potential for overcoming the inadequacies of purely quantitative methods of text information retrieval. Under the Tipster contracts in phases I through III, GE group has set out to explore this potential through development and evaluation of new text processing techniques. This work resulted in some significant advances and in a better understanding on how NLP may benefit IR. Tipster research has laid a critical groundwork for future work.

In this paper we summarize GE work on document detection in Tipster Phase III. Our summarization research is described in a separate paper appearing in this volume.

Background

The main thrust of this project has been to demonstrate that robust if relatively shallow NLP can help to derive better representation of text documents for indexing and search purposes than any simple word and string-based methods commonly used in statistical full-text retrieval. This was based on the premise that linguistic processing can uncover certain critical *semantic* aspects of document content, something that simple word counting cannot do, thus leading to more accurate representation. The project's progress has been rigorously evaluated in a series of five Text Retrieval Conferences (TREC's) organized by the U.S. Government under the guidance of NIST and DARPA. Since 1995, the project scope widened substantially to include several parallel efforts at GE, Rutgers, Lockheed Martin Corporation, New York University, University of Helsinki, and Swedish Institute for Computer Science (SICS). We have also collaborated with SRI International during TREC-6. At TREC we demonstrated that NLP can be done efficiently on a very large scale, and that it can have a significant impact on IR. At the same time, it became clear that exploiting the full potential of linguistic processing is harder than originally anticipated.

Table 1: Performance gains attributed to NLP indexing vs. query length

	T-2: 115 terms		T-3: 70 terms		T-4: 10 terms	
RUNS	Base	+NL	Base	+NL	Base	+NL
Prec.	0.22	0.31	0.22	0.27	0.20	0.22
change		+40%		+20%		+10%

Not surprisingly, we have noticed that the amount of improvement in recall and precision which we could attribute to NLP, appeared to be related to the quality of the initial search request, which in turn seemed unmistakably related to its length (cf. Table 1). Long and descriptive queries responded well to NLP, while terse one-sentence search directives showed hardly any improvement. This was not particularly surprising or even new, considering that the shorter queries tended to contain highly discriminating words in them, and that was just enough to achieve the optimal performance. On the other hand, comparing various evaluation categories at TREC, it was also quite clear that the longer queries just did better than the short ones, no matter what their level of processing. Furthermore, while the short queries needed no better indexing than with simple words, their performance remained inadequate, and one definitely could use better queries. Therefore, we started looking into ways to build full-bodied search queries, either automatically or interactively, out of users' initial search statements.

TREC-5 (1996), therefore, marks a shift in our approach away from text representation issues and towards query development problems. While our TREC-5 system still performs extensive text processing in order to extract phrasal and other indexing terms, our main focus moved on to query construction using words, sentences, and entire pas-

sages to expand initial search specifications in an attempt to cover their various angles, aspects and contexts. Based on the observations that NLP is more effective with highly descriptive queries, we designed an expansion method in which entire passages from related, though not necessarily relevant documents were quite liberally imported into the user queries. This method appeared to have produced a dramatic improvement in the performance of several different statistical search engines that we tested boosting the average precision by anywhere from 40% to as much as 130%. Therefore, topic expansion appears to lead to a genuine, sustainable advance in IR effectiveness. Moreover, we show in TREC-6 and TREC-7 that this process can be automated while maintaining the performance gains.

The other notable new feature of our TREC-5 system is the *stream architecture*. It is a system of parallel indexes built for a given collection, with each index reflecting a different text representation strategy. These indexes are called *streams* because they represents different streams of data derived from the underlying text archive. A retrieval process searches all or some of the streams, and the final ranking is obtained by merging individual stream search results. This allows for an effective combination of alternative document representation and retrieval strategies, in particular various NLP and non-NLP methods. The resulting meta-search system can be optimized by maximizing the contribution of each stream. It is also a convenient vehicle for an objective evaluation of streams against one another.

NLP-Based Indexing in Information Retrieval

In information retrieval (IR), a typical task is to fetch *relevant* documents from a large archive 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 fundamental problem remains to be 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 (or surrogates representing combinations of 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.[1] Our goal is to have the documents on the same topic placed close together, while those on different topics placed sufficiently apart. The above should hold for any topics, a daunting task indeed, which is additionally complicated by the fact that 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.

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 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,

[1] In a vector-space model term weights are represented as coordinate values; in a probabilistic model estimates of prior probabilities are used.

where alternative structures are reduced to a "normal form", helps to achieve the 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. In order to obtain the head+modifier pairs of respectable quality, we used a full-scale robust syntactic parsing (TTP) In 1998, in collaboration with the University of Helsinki, we used their Functional Dependency Grammar system to perform all linguistic analysis of TREC data and to derive multiple dependency-based indexing streams.

Stream-based Information Retrieval Model

The stream model was conceived to facilitate a thorough evaluation and optimization of various text content representation methods, including simple quantitative techniques as well as those requiring complex linguistic processing. Our system encompasses a number of statistical and natural language processing techniques that capture different aspects of document content: combining these into a coherent whole was in itself a major challenge. Therefore, we designed a distributed representation model in which alternative methods of document indexing (which we call "streams") are strung together to perform in parallel. Streams 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 streams 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.

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 adapt them in any way.

Advanced Linguistic Streams

Head+Modifier Pairs Stream

Our linguistically most 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. It has to be noted that while the head-modifier relation may suggest semantic dependence, what we obtain here is strictly syntactic, even though the semantic relation is what we are really after. This means in particular that the inferences of the kind where a *head+modifier* is taken as a specialized instance of *head*, are inherently risky, because the head is not necessarily a semantic head, and the *modifier* is not necessarily a semantic modifier, and in fact the opposite may be the case. In the experiments that we describe here, we have generally refrained from semantic interpretation of head-modifier relationship, treating it primarily as an *ordered* relation between otherwise equal elements. Nonetheless, even this simplified relationship has already allowed us to cut through a variety of surface forms, and achieve what we thought was a non-trivial level of normalization. The apparent lack of success of linguistically-motivated indexing in information retrieval may suggest that we haven't still gone far enough.

In our system, the head+modifier pairs stream is derived through a sequence of processing steps that include:

1. Part-of-speech tagging
2. Lexicon-based word normalization (extended "stemming")
3. Syntactic analysis with TTP parser
4. Extraction of head+modifier pairs

5. Corpus-based disambiguation of long noun phrases

Syntactic analysis with TTP Parsing reveals finer syntactic relationships between words and phrases in a sentence, relationships that are hard to determine accurately without a comprehensive grammar. Some of these relationships do convey semantic dependencies, e.g., in *Poland is attacked by Germany* the subject+verb and verb+object relationships uniquely capture the semantic relationship of who attacked whom. The surface word-order alone cannot be relied on to determine which relationship holds. From the onset, we assumed that capturing semantic dependencies may be critical for accurate text indexing. One way to approach this is to exploit the syntactic structures produced by a fairly comprehensive parser.

TTP (Tagged Text Parser) is based on the Linguistic String Grammar developed by Sager (Sager 1981) . The parser currently encompasses some 400 grammar productions, but it is by no means complete. The parser's output is a regularized parse tree representation of each sentence, that is, a representation that reflects the sentence's logical predicate-argument structure. For example, logical subject and logical object are identified in both passive and active sentences, and noun phrases are organized around their head elements. The parser is equipped with a powerful skip-and-fit recovery mechanism that allows it to operate effectively in the face of ill-formed input or under a severe time pressure. TTP has been shown to produce parse structures which are no worse than those generated by full-scale linguistic parsers when compared to hand-coded Treebank parse trees (Strzalkowski and Scheyen 1996).

Extracting head+modifier pairs 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. There are of course exceptions. For example, the three-word phrase "former Soviet president" would be broken into two pairs "former president" and "Soviet president", both of which denote things that are potentially quite different from what the original phrase refers to, and this fact may have potentially a negative effect on retrieval precision. This is one place where a longer phrase appears more appropriate. Below is a small sample of head+modifier pairs extracted (proper names are not included):

original text:

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

head+modifier pairs:

damage+physical, cause+damage, award+assist, award+compensate, compensate+monetary, assist+medical, entitle+veteran

Corpus-based disambiguation of long noun phrases The notorious structural ambiguity of nominal compounds remains a serious difficulty in obtaining quality head-modifier pairs. What it means is that word order information cannot be reliably used to determine relationships between words in complex phrases, which is required to decompose longer phrases into meaningful head+modifier pairs. In order to cope with ambiguity, the pair extractor looks at the distribution statistics of the compound terms to decide whether the association between any two words (nouns and adjectives) in a noun phrase is both syntactically valid and semantically significant. For example, we may accept language+natural and processing+language from "natural language processing" as correct, however, case+trading would make a mediocre term when extracted from "insider trading case". On the other hand, it is important to extract trading+insider to be able to match documents containing phrases "insider trading sanctions act" or "insider trading activity". Phrasal terms are extracted in two phases. In the first phase, only unambiguous head-modifier pairs are generated, while all structurally ambiguous noun phrases are passed to the second phase "as is". In the second phase, the distributional statistics gathered in the first phase are used to predict the strength of alternative modifier-modified links within ambiguous phrases. For details, the reader is referred to (Strzalkowski et al. 1995).

Simple Noun Phrase Stream

In contrast to the elaborate process of generating the head+modifier pairs, unnormalized noun groups are collected from part-of-speech tagged text using a few regular expression patterns. No attempt is made to disambiguate, normalize, or get at the internal structure of these phrases, other than the stemming which has been applied to text prior to the phrase extraction step. The following phrase patterns have been used, with phrase length arbitrarily limited to the maximum 7 words:

1. a sequence of modifiers (adjectives, participles, etc.) followed by at least one noun, such as: "cryonic suspension", "air traffic control system";

2. proper noun sequences modifying a noun, such as: "u.s. citizen", "china trade";

3. proper noun sequences (possibly containing '&'): "warren commission", "national air traffic controller".

The motivation for having a phrase stream is similar to that for head+modifier pairs since both streams attempt to capture significant multi-word indexing terms. The main difference is the lack of normalization, which makes the comparison between these two streams particularly interesting.

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.

Other Streams

Stems Stream The stems stream is the simplest, yet the most effective of all streams, a backbone of the multistream model. It consists of stemmed single-word tokens (plus hyphenated phrases) taken directly from the document text (exclusive of stopwords). The stems stream provides the most comprehensive, though not very accurate, image of the text it represents, and therefore it is able to outperform other streams that we used thus far. We believe however, that this representation model has reached its limits, and that further improvement can only be achieved in combination with other text representation methods. This appears consistent with the results reported at TREC.

Unstemmed-Word Stream In some experiments, notably in routing where incoming documents are assigned to one or more "standing" queries, or profiles, 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. The purpose of having this stream was to see if and when the lexical form of words can help to increase precision, while possibly sacrificing recall in some types of queries. In routing, where queries are extensively tuned through training, having multiple word forms allows, in theory at least, finer-grained adjustments.

Fragments Stream For the routing experiments we also used a stream of fragments. This was the result of splitting the documents of the stems stream into fragments of constant length (1024 characters) and indexing each fragment as if it were a separate document. The queries used with this stream were the same as with the stems stream. Unlike in the regular stream, where the entire documents were retrieved, here each document fragment was scored and ranked independently. The rank of a document was determined by the highest-scoring fragment contained by this document. This stream was motivated by the large body of work into passage-level retrieval (Callan 1994), (Kwok et al. 1993), and its primary purpose was to provide a benchmark for the locality stream described next.

Stream Merging and Weighting

The results obtained from different streams are lists 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

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

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 stronger (i.e., better performing) 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 between the top 10th and 20th 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 (Strzalkowski et al. 1997).

Note that long text queries benefit more from linguistic processing.

TREC-7 participation

In TREC-7, the GE/Rutgers/SICS/Helsinki team has performed runs in the main ad-hoc task. We used two retrieval engines, SMART and InQuery, built into the stream model architecture. The processing of TREC data was performed at Helsinki using the commercial Functional Dependency Grammar (FDG) text processing toolkit. Six linguistic streams have been produced, as described below. Processed text streams were sent via ftp to Rutgers for indexing using their version of Inquery system. Additionally, 4 steams produced by GE NLToolset for TREC-6 were reused in SMART indexing.

Adhoc topics were processed at GE using both automatic and manual topic expansion. We used the interactive Query Expansion Tool to expand topics with automatically generated summaries of top 30 documents retrieved by the original topic. Manual intervention was restricted to accept/reject decisions on summaries. We observed time limit of 10 minutes per topic.

Automatic topics expansion was done by replacing human summary selection by an automatic procedure, which accepted only the summaries that obtained sufficiently high scores.

Two sets of expanded topics (automatic and manual) were sent to Helsinki for NL processing, and then on to Rutgers for retrieval. Rankings were obtained from each stream index and then merged using a combined strategy developed at GE and SICS.

TREC-7 Submissions

1. SUMMARIZATION-BASED MANUALLY-ASSISTED TOPIC EXPANSION. This multi-stream manual Inquery run was produced with manually expanded topics. Summaries used in expansion were derived from top-ranked documents retrieved by SMART using the initial topics (title and description fields only).
2. SUMMARIZATION-BASED AUTOMATIC TOPIC EXPANSION I. This single-stream automatic Inquery run was produced with automatically expanded topics. Plain stems stream and syntactic noun phrase stream were combined and converted into a single Inquery-syntax representation (tokens and quoted strings).
3. SUMMARIZATION-BASED AUTOMATIC TOPIC EXPANSION II. This multi-stream automatic run was produced using SMART rather than Inquery. Automatically expanded queries were NL processed using GE NLToolset.

Helsinki NLP System

We used Helsinki's Functional Dependency Grammar (FDG) includes the EngCG-2 tagger and dependency syntax which links phrase heads to their modifiers and verbs to their complements and adjuncts.

FDG was applied to the whole corpus, with the output passed to the stream extractor. The streams were generated as follows:

- SIMPLE STREAMS

1. STEM: just stemmed words, stopwords removed.
2. NAME: all proper names
3. AAN: simple noun phrases with attributives. Basically adjective-noun sequences minus some exceptions.

- DIRECT DEPENDENCY STREAMS

1. SV: subject-verb pairs where the subject is a noun phrase.
2. VO: verb-complement pairs. The complement includes objects and some object-like adverbial classes.

- INDIRECT DEPENDENCY STREAMS

1. NOFN: *N1 ... of ... N2* pairs, where N1 and N2 are heads of simple noun phrases.
2. SC: subject-complement pairs where the complement modifies the subject, e.g., *flowers grow wild - wild+flower.*

Topic Expansion Experiments

In this section we discuss a semi-interactive approach to information retrieval which consists of two tasks performed in a sequence. First, the system assists the searcher in building a comprehensive statement of information need, using automatically generated topical summaries of sample documents. Second, the detailed statement of information need is automatically processed by a series of natural language processing routines in order to derive an optimal search query for a statistical information retrieval system. We investigate the role of automated document summarization in building effective search statements.

In the opening section of this paper we argued that the quality of the initial search topic, or user's information need statement is the ultimate factor in the performance of an information retrieval system. This means that the query must provide a sufficiently accurate description of what constitutes the relevant information, as well as how to distinguish this from related but not relevant information. We also pointed out that today's NLP techniques are not advanced enough to deal effectively with semantics and meaning, and instead they rely on syntactic and other surface forms to derive representations of content.

In order to overcome these limitations, many IR systems allow varying degrees of user interaction that facilitates query optimization and calibration to closer match user's information seeking goals. A popular technique here is relevance feedback, where the user or the system judges the relevance of a sample of results returned from an initial search, and the query is subsequently rebuilt to reflect this information. Automatic relevance feedback techniques can lead to a very close mapping of *known* relevant documents, however, they also tend to overfit, which in turn reduces their ability of finding *new* documents on the same subject. Therefore, a serious challenge for information retrieval is to devise methods for building better queries, or in assisting user to do so.

Building effective search topics

We have been experimenting with manual and automatic natural language query (or topic, in TREC parlance) building techniques. This differs from most query modification techniques used in IR in that our method is to reformulate the user's statement of information need rather than the search system's internal representation of it, as relevance feedback does. Our goal is to devise a method of full-text expansion that would allow for creating exhaustive search topics such that: (1) the performance of any system using the expanded topics would be significantly better than when the system is run using the original topics, and (2) the method of topic expansion could eventually be automated or semi-automated so as to be useful to a non-expert user. Note that the first of the above requirements effectively calls for a free text, unstructured, but highly precise and exhaustive description of user's search statement. The preliminary results from TREC evaluations show that such an approach is indeed very effective.

One way to view query expansion is to make the user query resemble more closely the documents it is expected to retrieve. This may include 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, query expansion usually is typically 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. An alternative to term-only expansion is a full-text expansion described in (Strzalkowski et al. 1997). In this approach, search topics are expanded by pasting in entire sentences, paragraphs, and other sequences directly from any text document. To make this process efficient, an initial search is performed with the unexpanded queries and the top N (10-30) returned documents are used for query expansion. These documents, irrespective of their overall relevancy to the search topic, are scanned for passages containing concepts referred to in the query. The resulting expanded queries undergo further text processing steps, before the search is run again. We need to note that the expansion material was found in both relevant and non-relevant documents, benefiting 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 feedback.

Summarization-based topic expansion

We used our automatic text summarizer to derive query-specific summaries of documents returned from the first round of retrieval. The summaries were usually 1 or 2 consecutive paragraphs selected from the original document text. The initial purpose was to show to the user, by the way of a quick-read abstract, why a document has been retrieved. If the summary appeared relevant and moreover captured some important aspect of relevant information, then the user had an option to paste it into the query, thus increasing the chances of a more successful subsequent search. Note again that it wasn't important if the summarized documents were themselves relevant, although they usually were.

The topic expansion interaction proceeds as follows:

1. The initial natural language statement of information need is submitted to SMART-based NLIR retrieval engine via a Query Expansion Tool (QET) interface. The statement is converted into an internal search query and run against the TREC database.[2]

2. NLIR returns top N (=30) documents from the database that match the search query.

3. The user determines a topic for the summarizer. By default, it is the title field of the initial search statement (see below).

4. The summarizer is invoked to automatically summarize each of the N documents with respect to the selected topic.

5. The user reviews the summaries (spending approx. 5-15 seconds per summary) and *de-selects* these that are *not* relevant to the search statement.

6. All remaining summaries are automatically attached to the search statement.

7. The expanded search statement is passed through a series of natural language processing steps and then submitted for the final retrieval.

[2]TREC-6 database consisted of approx. 2 GBytes of documents from Associated Press newswire, Wall Street Journal, Financial Times, Federal Register, FBIS and other sources (Harman & Voorhees 1998).

Implementation and evaluation

We have developed an automatic text summarizer as part of our Tipster Phase III contract. This work is described in a separate paper included in this volume.

We have included the summarizer as a helper application within the user interface to the natural language information retrieval system. In this application, the summarizer is used to derive query-related summaries of documents returned from database search. The summarization method used here is the same as for generic summaries described thus far, with the following exceptions:

1. The passage-search "query" is derived from the user's document search query rather than from the document title.

2. The distance of a passage from the beginning of the document is not considered towards its summary-worthiness.

The topical summaries are read by the users to quickly decide their relevance to the search topic and, if desired, to expand the initial information search statement in order to produce a significantly more effective query. The following example shows a topical (query-guided summary) and compares it to the generic summary (we abbreviate SGML for brevity).

INITIAL SEARCH STATEMENT:

< *title* > Evidence of Iranian support for Lebanese hostage takers.

< *desc* > Document will give data linking Iran to groups in Lebanon which seize and hold Western hostages.

FIRST RETRIEVED DOCUMENT (TITLE):

Table 3: Performance improvement for expanded queries

queries:	original	original	expanded	expanded
SYSTEM	SMART	NLIR	SMART	NLIR
PRECISION				
Average	0.1429	0.1837	0.2672	0.2859
%change		+28.5	+87.0	+100.0
At 10 docs	0.3000	0.3840	0.5060	0.5200
%change		+28.0	+68.6	+73.3
At 30 docs	0.2387	0.2747	0.3887	0.3940
%change		+15.0	+62.8	+65.0
At 100 doc	0.1600	0.1736	0.2480	0.2574
%change		+8.5	+55.0	+60.8
Recall	0.57	0.53	0.61	0.62
%change		-7.0	+7.0	+8.7

Arab Hijackers' Demands Similar To Those of Hostage-Takers in Lebanon

SUMMARIZER TOPIC:

Evidence of Iranian support for Lebanese hostage takers

TOPICAL SUMMARY (used for expansion):

Mugniyeh, 36, is a key figure in the security apparatus of Hezbollah, or Party of God, an Iranian-backed Shiite movement believed to be the umbrella for factions holding most of the 22 foreign hostages in Lebanon.

GENERIC SUMMARY (for comparison):

The demand made by hijackers of a Kuwaiti jet is the same as that made by Moslems holding Americans hostage in Lebanon - freedom for 17 pro-Iranian extremists jailed in Kuwait for bombing U.S. and French embassies there in 1983.

PARTIALLY EXPANDED SEARCH STATEMENT:

< *title* > Evidence of Iranian support for Lebanese hostage takers.

< *desc* > Document will give data linking Iran to groups in Lebanon which seize and hold Western hostages.

< *expd* > Mugniyeh, 36, is a key figure in the security apparatus of Hezbollah, or Party of God, an Iranian-backed Shiite movement believed to be the umbrella for factions holding most of the 22 foreign hostages in Lebanon.

TREC Evaluation Results

Table 3 lists selected runs performed with the NLIR system on TREC-6 database using 50 queries (TREC topics) numbered 301 through 350. The expanded query runs are contrasted with runs obtained using TREC original topics using NLIR as well as Cornell's SMART (version 11) which serves here as a benchmark. The first two columns are automatic runs, which means that there was no human intervention in the process at any time. Since query expansion requires human decision on summary selection, these runs (columns 3 and 4) are classified as "manual", although most of the process is automatic. As can be seen, query expansion produces an impressive improvement in precision at all levels. Recall figures are shown at 1000 retrieved documents.

Query expansion appears to produce consistently high gains not only for different sets of queries but also for different systems: we asked other groups participating in TREC to run search using our expanded queries, and they reported similarly large improvements.

Finally, we may note that NLP-based indexing has also a positive effect on overall performance, but the improvements are relatively modest, particularly on the expanded queries. A similar effect of reduced effectiveness of linguistic indexing has been reported also in connection with improved term weighting techniques.

Automatic Topic Expansion

In TREC-7 we started experimenting with completely automated topic expansion. We used the same approach to expansion as outlined below with the following modifications:

1. Top 100 documents retrieved by the initial, unexpanded topic are summarized, rather than 30 used in manual mode. This is because we need to rely on a strict notion of topicality of the summary, and therefore must look at more documents to obtain any expansion. From a user's perspective, this is entirely transparent, however.

2. We replace human selection of expanding summaries by an automatic functions that measure the overlap between the summary and the topic. This overlap, measured over content terms (i.e., with exclusion of common words and certain other words), should be high enough to prevent false matches, while not too high to allow for topic variants to be matched.

3. The summary parameters (i.e., length, spread, etc) is set to normalize its size in such as way as to support effective topicality detection. For example, straight 10short documents (too short!) or for very long documents (too long!).

Preliminary tests conducted using TREC-6 data showed a significant increase in precision over unexpanded queries, although still not as large as in manual expansion. These experiments require continuation.

Conclusions

We have developed a method to derive quick-read summaries from news-like texts using a number of shallow NLP and simple quantitative techniques. The summary is assembled out of passages extracted from the original text, based on a pre-determined DMS template. This approach has produced a very efficient and robust summarizer for news-like texts.

We used the summarizer, via the QET interface, to build effective search queries for an information retrieval system. This has been demonstrated to produce dramatic performance improvements in TREC evaluations. We believe that this query expansion approach will also prove useful in searching very large databases where obtaining a full index may be impractical or impossible, and accurate sampling will become critical.

Information Retrieval has made tremendous advances over the last 30 years in terms of accuracy, efficiency and robustness. It has also been widely commercialized in recent years, particularly on the Internet. In spite of this progress, many challenges remain, and more research is needed to achieve performance levels that would approach human-level accuracy. We believe that this requires a tighter integration of NLP.

Acknowledgements We would like to acknowledge significant contributions to this project from the following people: Jose Perez-Carballo, Louise Guthrie, Fang Lin, Jussi Karlgren, Pasi Tapanainen, Timo Jarvinen, Jim Leistensnider, Troy Straszheim, and Wang Jin. We thank Chris Buckley and Donna Harman for providing their IR engines (SMART and Prise) for this research. This paper is based upon work supported in part by the Defense Advanced Research Projects Agency under Tipster Phase-3 Contract 97-F157200-000.

References

Callan, Jamie. 1994. "Passage-Level Evidence in Document Retrieval." Proceedings of ACM SIGIR'94. pp. 302-310.

Harman, Donna, and Ellen Voorhees (eds). 1998. The Text Retrieval Conference (TREC-6). NIST Special Publication (to appear).

Kwok, K.L., L. Papadopoulos and Kathy Y.Y. Kwan. 1993. "Retrieval Experiments with a Large Collection using PIRCS." Proceedings of TREC-1 conference, NIST special publication 500-207, pp. 153-172.

Sager, Naomi. 1981. *Natural Language Information Processing.* Addison-Wesley.

Strzalkowski, Tomek, Louise Guthrie, Jussi Karlgren, Jim Leistensnider, Fang Lin, Jose Perez-Carballo, Troy Straszheim, Jin Wang, and Jon Wilding. 1997. "Natural Language Information Retrieval: TREC-5 Report." Proceedings of TREC-5 conference.

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

Strzalkowski, Tomek, Jose Perez-Carballo and Mihnea Marinescu. 1995. "Natural Language Information Retrieval: 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 Fourth Text REtrieval Conference (TREC-4), NIST Special Publication 500-236.

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.

Strzalkowski, Tomek, Fang Lin, Jose Perez-Carballo, and Jin Wang. 1997. "Natural Language Information Retrieval: TREC-6 Report." Proceedings of TREC-6 conference.