performed well, especially in routing. As before, there is a tremendous variation in the topic-by-topic results, suggesting that a great deal more research is needed to find how to get the best results in different routing and retrieval scenarios. We are encouraged by the progress of our system, as well as of the overall field, in these experiments, and are hopeful that in the coming years we will learn how to combine our promising results in corpus analysis with the more mature ranking and retrieval models of some of the other systems.

References

[1] Paul S. Jacobs. Using statistical methods to improve knowledge-based news categorization. *IEEE Expert*, 8(2):13–24, April 1993.

[2] George R. Krupka Paul S. Jacobs and Lisa F. Rau. A boolean approximation method for query construction and topic assignment in trec. In *Second Annual Symposium on Document Analysis and Information Retrieval*, May 1993.

This worked surprisingly well in cases where the topic statements were the topic title was a good description (e.g. "Welfare Reform") and absolutely horribly for those with vague titles (e.g. "Find Innovative Companies". We tried to recover from these by including more words from the description and narrative, but then we had to start recognizing the language of these descriptions, filtering out words like "relevant", "mention" and so forth.

The fully automatic ad hoc system certainly didn't do as well as the manual routing system, but it was still at or above median for more than half of the ad hoc topics. Considering that this method could be used within the context of most any legacy retrieval system, the result is worth noting. Furthermore, the generation of Boolean queries from natural language descriptions is an interesting, as well as practical, research problem, because many different retrieval systems can make some use of Boolean queries.

2.3 Ranking

Our document ranking method will be more fully described in the proceedings paper. In both routing and ad hoc, we used a set of word weights, acquired from the relevance judgements in the routing case and from the corpus data in the ad hoc case. We combined the weighted frequency of these terms with an overall count of the number of topic hits per document, normalizing for document length, to produce a score for each document. This was the result of trying many different approaches on the test data, so it was definitely a good method for our system.

However, in comparing our results with those of other systems, our precision curve across various recall points is not nearly as good as a system that does really good ranking. In routing, we are not sure that ranking is important, but it is certainly important in getting good results in TREC. So, we are inclined to try to combine our retrieval method with alternative ranking methods to see, for example, whether more terms are really necessary in order to get better ranking results.

The separation of retrieval and ranking seems to be a valuable tool both for experimental research and for identifying different techniques for applications. It is clearly a problem with both TREC-1 and TREC-2 that the routing task requires a comparison of documents across a large collection, when most routing applications deal with a stream of documents individually or in small groups.

3 Summary

GE's participation in TREC involved the implementation of a number of strategies for creating Boolean queries from the topic descriptions. A statistical corpus analyzer helped to refine queries for both the routing task, and to generate them automatically for the ad hoc task. The simple Boolean retrieval engine

```
TOULOUSE        TOPIC51         228.53395012396337
JETLINERS       TOPIC51         217.8171887668145
LUFTHANSA       TOPIC51         196.6003672315153
MD-80           TOPIC51         196.6003672315153
```

Clearly, these words all have some reason to be associated with this topic, but adding them to the appropriate group in each query (or ignoring them entirely) is a "manual" process. Thus we have tried, as a simple experiment, to generate the Boolean term groupings and expand each term automatically. In the days before the TREC-2 ad hoc test, we tried several different ways of doing this automatic Boolean query generation, and chose the one that worked best on the sample data.

Interestingly, our first attempt was to use the common methods for finding collocations and word associations in sentences, and these worked horribly for term expansion. The problem is that this approach finds more associations like "funeral" and "home" than it does "hostage" and "captive", and the latter, text-level associations are what's required to generate good queries.

The "solution" we tried was, for a sample of about 10 million words in the corpus, to choose the top 20 words based on TF/IDF weights for each document, store the frequency of association among these terms, and then weight each pair using the mutual information statistic. This was much better than using sentence-level information, although it is still a very straightforward approach. For example, the following are the top 10 terms associated with the word "hostage" (in order):

```
hostages
Lebanon
Beirut
Iran
release
Terry
kidnappers
kidnapped
Jihad
Anderson
```

While this is certainly not the optimal set of terms to use in place of "hostage", it is a good start.

The next problem in automatic query construction is when to use a combination of terms and when to use a single term. For example, the term "weather-related fatalities" is a combination of two word groups (weather and fatalities) while "Iran-contra affair" is really only one group (Iran-contra), even though it might appear that "affair" is a significant term.

Again we took the direct approach, choosing to combine terms whenever there was a reasonable percentage of overlap between their associated terms.

tests appear first. In this case, the TERM test AFRICAN appears with a much lower number simply because it is used in many different queries.

The matcher, which can work either on complete documents or paragraphs (but we used paragraph matching only in TREC-2) goes through every word in its input and, using a fast table look-up, sets the TERM tests to true for every word it encounters. At the end of input, either the end of the paragraph or end of each document, it runs through the table of possible tests from low numbers to high numbers and sets tests to true if their conditions are satisfied. A topic test produces a match if it has become true at the end of this process, meaning that the paragraph or document has passed the pre-filter for that query. A single paragraph, of course, can satisfy multiple queries.

This portion of the system was implemented in the space of a few days, and is almost entirely the same as in TREC-1. Our focus since last year has been on query construction and ranking rather than matching or retrieval.

2.2 Query construction

Our approach assumes, in general, that manual query construction is acceptable for routing. In ad hoc retrieval, query time can be of the essence, but in many routing applications, queries are developed and refined over time. The amount of time spent on query construction using a manual method in our system is comparable to the amount of time spent on the topic descriptions used for automatic query generation.

In manual routing, our approach uses a statistical corpus analysis, developed originally for text categorization [1], to pull out terms based on their relative frequency in relevant documents for each topic. The statistic used combines the entropy-based mutual information statistic (testing the independence of each term with each topic) with a correction for low-frequency terms and for ambiguous words. Words with high weights have a high degree of association with a topic. This statistical analysis is also used in ranking.

For example, the following are the top words for Topic 51, "Airbus subsidies":

```
A-330           TOPIC51         1263.3525710477136
AIRBUS          TOPIC51         1183.1683350538406
A-340           TOPIC51         1178.2875994828662
INDUSTRIE       TOPIC51         1071.938154759883
A-320           TOPIC51         1067.5230999824726
MESSERSCHMITT-BOELKOW-BLOHM TOPIC51 851.3637360181779
AERONAUTICAS    TOPIC51         843.3453644472766
CONSTRUCCIONES TOPIC51          807.988536024671
AEROSPATIALE    TOPIC51         762.5184528591329
MBB             TOPIC51         722.6311893197188
WIDE-BODY       TOPIC51         617.7953779264718
MD-11           TOPIC51         613.8643234939403
```

terms or combinations of terms, and because the Boolean matcher must match the simplest expressions first.

For the topic description given above, the output of the rule compiler will include the following tests:

```
52          TERM    AFRICAN
 ...
2029        TERM    MEASURES
 ...
2134        TERM    SANCTION
2135        TERM    SANCTIONS
2136        TERM    DISINVESTMENT
2138        TERM    SULLIVAN
2139        TERM    PRINCIPLES
2141        TERM    PUNITIVE
2144        TERM    BUTHELEZI
2145        TERM    PRETORIA
2146        TERM    ANTI-APARTHEID
2147        TERM    APARTHEID
2149        TERM    DE
2150        TERM    KLERK
2152        TERM    SOUTH
2153        TERM    AFRICA
 ...
2137        OR      2134 2135 2136
2140        AND     2138 2139
2142        AND     2141 2029
2143        OR      2137 2140 2142
2148        OR      2144 2145 2146 2147
2151        AND     2149 2150
2154        OR      2153 52
2155        AND     2152 2154
2156        OR      2148 2151 2155
2157        AND     2143 2156
2158        AND     2156 2143
 ...
TOPIC052    OR       2157 2158
```

Each line in the above data gives a unique number (or topic designator) to the test, a test identifier (either TERM for a simple word test, OR, or AND), and a list of simple terms or previous tests. For example, test 2137 depends on tests 2134, 2135, and 2136, and is true if any of those tests is true, namely, if the text includes any of the words *sanction*, *sanctions*, or *disinvestment*. The tests are automatically ordered so that all tests that are dependent on other tests will have higher numbers than the tests they depend on; thus all TERM

2 Our Approach

The basic approach in our system is to compile queries into Boolean tables that can be matched at high speed against a stream of input text. This approach is meant for routing, and also to be compatible with "downstream" analysis such as what we do in TIPSTER data extraction. In fact, the Boolean compiler we use is designed for handling the much more complex expressions that our system uses in data extraction.

2.1 The Boolean Matcher

The Boolean tables are efficiently organized so that a C program (which we now know as NLGREP can match them against incoming texts at a rate of about 1 million words per minute. This program spends little time on anything other than marking where words in each document match terms in the table. In both routing and retrieval, the total number of terms used for a set of 50 topics is about 2000, or about 40 unique terms per topic. All other words are ignored entirely.

For example, the following is a query for the topic "South African sanctions":

```
sanction == [(member sanction sanctions disinvestment)
     <Sullivan Principles>
     <punitive *2 measures>] ;

safrica == [(member Buthelezi Pretoria anti-apartheid apartheid)
          <De Klerk> <South (member Africa African)> ] ;

;;; rule 1
$sanction * $safrica => (mark-topic 52) ;
```

This description says that any matching text must have *both* an indicator of South Africa ($safrica) and one of sanctions ($sanction), and that the sanction phrase and South Africa phrase must appear in the same paragraph in the document.

A sanction phrase can be any of the simple words *sanction*, *sanctions*, or *disinvestment*, or any phrase including *punitive measures* with no more than two intervening words (like *punitive economic measures*). A South Africa phrase can also be either one of a group of simple words, or a phrase, like *De Klerk*, *South Africa*, or *South African*.

These queries or topic descriptions can be quite complex, and the method has been designed to handle many queries simultaneously, so the rule compiler is designed to produce expressions that can be efficiently applied within a large set of queries. This is important because many queries can share the same simple

relevant texts retrieved has been shown to be lower than 10%, and a high percentage of material that is retrieved can be irrelevant. Also, it can be very difficult for searchers to compose "queries" that combine the words that are effective in locating relevant material without finding large quatities of irrelevant information as well.

GE's involvement in TREC represents a relatively low level of effort aimed at bringing together natural language text processing, data extraction, and statistical corpus analysis methods. Our project aims at innovative approaches to extracting information from text, best exemplified in our participation in the MUC and TIPSTER extraction evaluations and in operational text management systems in GE. In TREC-1, we attempted to show the benefit of natural language interpretation by using Boolean approximation to selection portions of text that could be further interpreted. The main result of this was that natural language seemed to have very little to offer as a precision filtering method, because routing and retrieval problems stem largely from having the wrong terms in the queries [2]. Thus, in TREC-2, we have stuck with the Boolean engine, concentrating on the use of corpus analysis to improve the queries.

Our results in TREC-2, as in TREC-1, were quite good relative to other systems. The manual routing system, which comprised over 99% of our effort, produced at 11 point average of .3308, with an average of 45 relevant documents in the top 100. The residual effort went into a fully automatic ad hoc method, which produced at 11 point average of .2183 and an average of 37 relevant documents in the top 100. As in TREC-1, performance varied dramatically by topic. The routing system showed the best results on 8 of 50 topics. Yet it was below median on 17 topics. This not only suggests areas for further improvement, but also punctuates a variety of query and corpus effects on performance (for example, the Boolean approach still does worst on certain topics where there are very few relevant documents).

While it is very hard to measure progress between TREC-1 and TREC-2 because none of the numbers are directly comparable, the difference between our .2078 routing average in TREC-1 and .3308 in TREC-2 is large enough that we are quite pleased with the rate of progress and confident that we are nowhere near the peak that can be achieved with manual, Boolean routing. In addition, the automatic ad hoc method that we tried, while showing pitiful performance on some of the more convoluted topic descriptions, had a better average than our manual ad hoc system last year.

Thus, there is a great deal to be gained using corpus analysis to automate or assist in query generation, even within the context of straightforward retrieval methods. In addition to having promise for "legacy" operational systems, these results suggest that natural language methods, focused on corpus analysis and query generation, probably can help in improving performance.

GE in TREC-2: Results of A Boolean Approximation Method for Routing and Retrieval*

Paul S. Jacobs
GE Research and Development Center
Schenectady, NY 12301
(518) 387 - 5314

Abstract

This report describes a few experiments aimed at producing high accuracy routing and retrieval with a simple Boolean engine. There are several motivations behind this work, including a tie-in of Boolean term combinations to our extraction methods, the existence and persistence of many "legacy" Boolean systems, and the recognition that many information retrieval problems stem from bad queries rather than bad retrieval. The results show very high accuracy, and significant progress, using a Boolean engine for routing based on queries that are manually generated with the help of corpus data. In addition, the results of a straightforward implementation of a fully automatic ad hoc method show some promise of being able to do good automatic query construction within the context of a Boolean system.

1 Introduction

Full-text search is currently the simplest and most commonly-used method for locating information in large volumes of free text. Because users are accustomed to describing what they are looking for with specific words, and those words are often found in the texts, searching the text for selected words or word combinations is a natural and easy-to-implement method for information retrieval. However, it can be very inaccurate. In some experiments, the percentage of

*This research was sponsored in part by the Advanced Research Project Agency. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Advanced Research Project Agency or the US Government.