Information and Telecommunication Technology Center, Technical Report: FY97-TR-11100-1.

Tuning a Corpus Analysis Approach for Automatic Query Expansion

Susan Gauch and Jianying Wang

sgauch@eecs.ukans.edu
Electrical Engineering and Computer Science
University of Kansas

ABSTRACT

Searching online text collections can be both rewarding and frustrating. While valuable information can be found, typically many irrelevant documents are also retrieved and many relevant ones are missed. Terminology mismatches between the user's query and document contents are a main cause of retrieval failures. Expanding a user's query with related words can improve search performance, but finding and using related words is an open problem.

This research uses corpus analysis techniques to automatically discover similar words directly from the contents of the untagged databases. Using these similarities, user queries are automatically expanded, resulting in conceptual retrieval rather than requiring exact word matches between queries and documents. This work has been extended to multi-database collections where each sub-database has a collection-specific similarity matrix associated with it. If the best matrix is selected, substantial search improvements are possible. However, automatically selecting the appropriate matrix for a particular query remains under investigation.

1. INTRODUCTION

The goal of information retrieval is to identify documents which best match users information needs. At first glance, this seems very simple - merely identify documents (efficiently) which contain the words which are also contained in the user's query. However, the average query submitted by users to World Wide Web search engines is only two words long [Croft et al, 1995], which makes it difficult to identify relevant documents. There are likely to be many relevant documents available which are missed because they do not contain the exact words used in the query.

Automatically adding related words to a query can increase the number of relevant documents identified by increasing the number of words which are used for matching. This is one way to provide *conceptual retrieval*, rather than pure string matching. The user's initial query terms are taken as representatives of the concepts in which they are interested. Then, query expansion adds other terms related to the same concepts, providing a richer representation of the user's query. Our earlier work showed that an expert system which automatically reformulated Boolean queries by including terms from an online thesaurus was, indeed, able to improve search results [Gauch & Smith, 1993]. But, where are the expansion terms to come from? There are three main sources for related words which vary in their level of specificity: 1) query specific; 2) corpus specific; and 3) language specific.

<u>Query specific</u> terms can be identified by locating new terms in a subset of the documents retrieved by a specific query. This is the approach taken by relevance feedback systems, where related terms come from the contents of user-identified relevant documents. This has been shown to be quite effective [Harman, 1992], but it requires the users to indicate which documents are relevant. More recently, search improvements are being achieved [Xu & Croft, 1996] without the need for user relevance judgments. Local analysis of the top N retrieved documents, where N varies from 20 to 100 based on the database being searched, has been found to increase performance over 23% on the TREC3 and TREC4 corpora. The main drawback to these approaches is that there is a reasonable amount of computation that takes place after the user submits his query, which can be a problem for interactive systems.

<u>Corpus specific</u> terms are found by analyzing the contents of a particular full-text database to identify terms used in similar ways. It may be hand-built [Gauch & Smith, 1991], a time-consuming and ad hoc process, or created automatically. Traditional automatic thesaurus construction techniques grouped words together based on their occurrence patterns at a document level [Crouch & Yang, 1992; Qui & Frei, 1993], i.e., words which often occur together in documents are assumed to be similar. Other approaches look inside documents to consider finer-grained word usage patterns. Some incorporate syntactic analysis [Grefenstette, 1992]. Most, however, look for co-occurrence patterns of words [Schütze & Pedersen, 1994] or noun phrases [Jing & Croft, 1994] within windows of a fixed size (measured in terms

of *n* words). Significant search improvements have been achieved with these systems (7.8% and 3.4% on TREC3 and TREC4 respectively [Xu & Croft, 1996]; 5% on TIPSTER Category B [Schütze & Petersen, 1994]; 18-30% on smaller corpora [Qui & Frei, 1993]). These approaches are computationally intensive, but the computations are all done once per database. The only component done on a per query basis is the actual query expansion itself.

<u>Language specific</u> terms may be found from generally available online thesauri which are not tailored for any particular text collection. [Liddy & Myaeng, 1993] use the Longman's Dictionary of Contemporary English, a semantically coded dictionary. [Voorhees, 1994] used WordNet [Miller, 1990], a manually-constructed network of lexical relationships. Because of ambiguity, this type of thesaurus is difficult to use because they include multiple meanings for most words. Selecting the correct meaning for expansion can be difficult. Small improvements (1% [Voorhees, 1994]) are possible with longer queries which provide clues for which word senses are involved, but expanding shorter queries actually degraded performance. In addition, a general thesaurus may not be applicable for more specialized collections which may have their own distinct sublanguages.

We have adopted a corpus-specific approach for locating related terms. We are particularly interested in these techniques because the main calculations are done *a priori*, before the user queries arrive. Also, because the information is built from the specific text collection, the related terms are automatically tuned for the particular database being searched. Finally, since our approach is entirely statistical is should, in principle, be applicable to databases in different languages, although we have not tested this, be . Our approach differs from those presented above in that it does not depend on term co-occurrence, *per se*. This will be further explained in Section 2.

2. CORPUS ANALYSIS TECHNIQUE

We have modified a corpus linguistics approach [Finch & Chater, 1992] that creates a matrix of term-term similarities. For words to be considered similar, they need not actually co-occur, however, they must occur in similar contexts. For example, we could deduce that "color" and "colour" were highly similar words because they are used in similar contexts, even though they are not likely to both appear in the same document. This approach is similar to others in that word usage within a given window is recorded. However, our window size is much smaller because we take into account the position of the words within the window, not just the words themselves. The fact that the word "the" appears immediately prior to a word *w* carries much more information about *w* (i.e., it is a noun or an adjective) than just the fact that the word "the" appears within a 7 word window of *w*. Thus, although we do not do explicit parsing of the text, we do get a quasi-syntactic categorization.

2.1 Similarity Calculation

The first step is to identify a set of words, the *target words*, whose pairwise similarities are to be calculated. Then, for each target word, we construct a *context vector* which summarizes information about word occurrences around the target word. This context vector is a concatenation of sub-vectors, one *position vector* for each position in the window (called the *context positions*). For example, in a window of size 5, there are 4 context positions: -2 (2 positions before the target word), -1(immediately before the target word), +1 (immediately after the target word) and +2 (2 positions after the target word). Each position vector is has one element for each *context words*, the words whose appearance in the window surrounding the target words is recorded. Generally, the context words are the n most frequent words in the database. Thus, each target word has an associated context vector of dimension N_p*N_C , where $N_p$is the number of context positions and N_C is the number of context words. From these context vectors, a N_t x N_t similarity matrix is calculated, where $N_t$is the number of target words.

Initially, the counts from all instances of a word form w_i are summed so that the entry in the corresponding context word position in the vector is the sum of the occurrences of that context word in that position for the corresponding target word form; it is the joint frequency of the context word. Consider an example in which there are only five context words, {"a", "black", "dog", "the, "very"} and two sentences containing the target word "dog" and we only observe the preceding two positions and the following two positions:

(1) The black <u>dog</u> barked very loudly.

(2) A brown <u>dog</u> barked very loudly.

Sentence	**Context Position**	**Observed Word**	**Context Vector**
1	-2	"The"	(0, 0, 0, 1, 0) 4th context word
	-1	"black"	(0, 1, 0, 0, 0) 2nd context word
	+1	"barked"	(0, 0, 0, 0, 0) not a context word
	+2	"very"	(0, 0, 0, 0, 1) 5th context word

Table 1. The context vectors for each of the 4 context positions around the occurrence of the target word "dog" in sentence 1.

The context vector for "dog" in sentence 1 is formed by concatenating the context vectors for each of the 4 context positions:

(0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1)

Similarly, the context vector for "dog" in sentence 2 would be:

(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1) and the combined vector for the word "dog" would be:

(1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2)

After the context vectors for each target word are created, the raw occurrence numbers are replaced by mutual information values [Church & Hanks, 1990] as follows:

$$MI(cw) = \log_2\left(\frac{Nf_{cw}}{f_c f_w} + 1\right)$$

MI(cw) expresses the mutual information value for the context word *c* appearing with the target word *w*. The mutual information is large whenever a context word appears at a much higher frequency, f_{cw}, in the neighborhood of a target word than would be predicted from the overall frequencies in the corpus, f_c and f_w. The formula adds 1 to the frequency ratio, so that a 0 (zero) occurrence corresponds to 0 mutual information. When the mutual information vectors are computed for a number of words, they can be compared to see which words have similar contexts. The comparison we chose is the inner product, or cosine measure, which can vary between -1.0 and +1.0 [Myaeng & Li, 1992].

Finally, to make the identification of the most highly similar terms to a given term more efficient, an auxiliary file is produced from the similarity matrix. It stores, for each target word, a list of words and their similarity values for all words with similarity above a given threshold. This *similarity list* is sorted in decreasing order by similarity value.

3. EVALUATION

To incorporate the results of the corpus analysis into an existing retrieval engine, SMART was modified to allow it to expand queries based on the similarity matrices, search the database with the expanded queries, and return the top 1000 documents for each query. We experimented with different databases, different similarity calculation parameters and different expansion techniques. Finally, we looked at the effects of creating separate similarity matrices for each database in a multi-database collection. The retrieval runs used SMART's *lnc* weights for the documents and the modified *ltc* weights for the queries.

3.1 Collections and Query Sets

Experiments are carried out on 3 collections: 1) TREC4 Category B (0.38 GBs) which is comprised of the Wall Street Journal (WSJ) on Tipster Disk 2 and the San Jose Mercury (SJM) on Tipster Disk 3 with 48 queries (without queries 201 and 206); 2) WSJ (0.25 GBs) only with 45 queries (without queries 201, 213, 214, 232, 236 which have no relevant documents); and 3) TREC5 Category A (2.1 GBs) consisting of the documents on both Tipster Disk 2 and TREC Disk 4.

3.2 Similarity Matrix Calculation Experiments with the TREC 4 Category B corpus

The goal of the first set of experiments was to tune the similarity calculation algorithm. There are four main parameters to evaluate: 1) the number of context words; 2) the number of context positions (i.e., the window size); 3) the amount of the corpus to process (i.e., the sample size); and 4) the number (and location) of the target words. Using a fixed target list, we initially did a series of experiments studying the number of context words, the window size and the sample database size [Gauch & Wang, 1996]. For each combination of the three factors, we evaluated a range of different similarity thresholds for expansion, and Table 2 reports the best performing threshold and the associated 11-pt average. These tests were carried out on the TREC 4 category B collection.

Sample size 10%			
Window Size **Context words**	**5**	**7**	**9**
150	(0.39, 0.1823)	(0.37, 0.1834)	(0.30, 0.1830)
200	(0.39, 0.1825)	(0.34, 0.1831)	(0.27, 0.1815)
250	(0.37, 0.1850)	(0.30, 0.1830)	(0.27, 0.1826)
Sample size 20%			
Context\Window	**5**	**7**	**9**
150	(0.50, 0.1813)	(0.45, 0.1840)	(0.40, 0.1860)
200	(0.45, 0.1834)	(0.38, 0.1887)	(0.34, 0.1873)
250	(0.45, 0.1811)	(0.38, 0.1852)	(0.32, 0.1861)
Sample size 30%			
Context\Window	**5**	**7**	**9**
150	(0.80, 0.1791)	(0.70, 0.1791)	(0.60, 0.1792)
200	(0.70, 0.1791)	(0.60, 0.1796)	(0.40, 0.1804)
250	(0.60, 0.1799)	(0.45, 0.1795)	(0.45, 0.1799)

Table 2. Experimental results for context words, window size and sample size on the TREC 4 category B collection. Note: unexpanded queries yield an 11-pt average of 0.1791.

3.2.1 Window Size

The window size determines the number of positions surrounding the target word occurrences whose contents are considered in forming the context vectors. Since it is directly related to the size of the context vectors (and hence the complexity of all calculations) want to use the smallest window size possible. In Table 3, we averaged all 11-pt averages for different window sizes in table 2 over all tests. We found that a window size of 7 (i.e., the preceding three positions and the following three positions of the target word are observed) produced comparable results to the larger size of 9, and thus was chosen in all subsequent tests. Note that this is much smaller than the window sizes used in other approaches, which usually use windows of 40 or more. Our smaller window is effective because we keep positional information which is normally discarded in other approaches.

Window Size	**5**	**7**	**9**
11-pt average	0.1815 (+1.3)	0.1828 (+2.1)	0.1828 (+2.1)

Table 3. The average 11-pt average for different window sizes, and the resulting improvement over the baseline performance (TREC4 Category B collection).

3.2.2 Context Word Size

Since we want to maximize the observations about each target word, we chose the highest frequency words in the corpus as the context words. Since the number of context words also affect the vector sizes (and hence the computation demands) we want to use the fewest possible that yield acceptable results. Table 4 reports the results of the experiments presented in Table 2, averaged over the number of context words. The best results were obtained with 200 context words, so this was used in all subsequent experiments.

Context Words	**150**	**200**	**250**
11-pt average	0.1819 (+1.6)	0.1828 (+2.1)	0.1825 (+2.0)

Table 4. The average 11-pt averages for different numbers of context words, and the resulting improvement over the baseline performance (TREC4 Category B collection).

3.2.3 Sample Size

Although the sample is read only once, it still affects the speed of the computations. Thus, we examined the effect of sample size on the search results. The 20% (76 MB) sample seemed to do best, and this was confirmed in later experiments.

Sample Size	**10% (38 MB)**	**20% (76 MB)**	**30% (114 MB)**
11-pt average	0.1829 (+2.1)	0.1848 (+3.2)	0.1795 (+0.2)

Table 5. The average 11-pt average for different sample sizes, and the resulting improvement over the baseline performance (TREC4 Category B collection).

3.2.4 Other Factors

We also ran a few small tests to check on the effectiveness of adding stemming and to confirm our belief that the positional information captured in the context vectors contributes to the quality of the results.

Calc. Method	**as described**	**with stemming**	**no position vectors (200)**	**no position vectors (1200)**
11-pt average	0.1887 (+5.4)	0.1880 (+5.0)	0.1382 (-22.8)	0.1727 (-3.6)

Table 6. The average 11-pt average for different matrix calculation techniques (TREC4 Category B collection: 20% sample, window size 7, 200 context words, 4000 target words, expansion threshold of 0.38).

From Table 6, we see that stemming the corpus before calculating the word (or, in this case, stem) similarities has a slight negative effect on the result, so we chose not to stem. If we ignore the postional information and create one 200 element context vector for the target word which records word occurrences anywhere in the window (rather than using the position vectors which record occurrences separately for each context position) we get a marked decrease in performance. To make a stronger case for the worth of the positional information we capture, if we use a context vector of length 1200 (which is the size of the 6 200-element position vectors concatenated) we still see a decrease in the retrieval results, albeit not as dramatic.

3.3 Similarity Matrix Calculation Experiments with the Wall Street Journal corpus

The TREC4 Category B corpus consists of two sub-collections: the Wall Street Journal (WSJ) and San Jose Mercury News (SJM). To avoid possible confusion in the similarity matrix due to differing word usage in the sub-collections, we conducted further experiments on the WSJ sub-collection alone. In particular, we re-examined the effect of sample size and extended our analysis to consider the number and location of the selected target words. Also, we used a somewhat different query expansion method from those in the earlier experiments, using a fixed number of expansion words per target word rather than an expanding with all words above a given threshold. The effect of the expansion technique used is discussed further in Section 3.4. Based on the results in Section 3.2, a window size of 7 and context word size of 200 are used in all of the following experiments.

3.3.1 Sample Selection

In Table 7, the average of 11-pt averages for different size of sample databases is presented. The trend seems that the performance is better as the size of sample database increases. The 11-pt average tends to be stable when the size of sample database is above 20%.

Sample Size	**5% (12.5 MB)**	**10% (25 MB)**	**20% (50 MB)**	**30% (75 MB)**
first run	(1, 0.1938)	(3, 0.2007)	(2, 0.2020)	(2, 0.1989)
second run	(0, 0.1885)	(4, 0.2002)	(3, 0.1982)	(2, 0.2015)
third run	(6, 0.2005)	(2, 0.1944)	(2, 0.2010)	(3, 0.2035)
Average	0.1943 (+3.1)	0.1984 (+5.3)	0.2004 (+6.3)	0.2013 (+6.8)
Std. Dev.	0.0049	0.0029	0.0016	0.0018

Table 7. The 11-pt averages for different sample sizes (WSJ database).

The 20% (50 MB) sample performs almost as well as the 30% (75 MB) sample, so it is used in subsequent tests. In addition, by selecting different samples of the same size, we were able to gauge the sensitivity of the results to the particular sample chosen. The 20% sample size seems to be the least sensitive to the actual sample chosen, as measured by the standard deviation.

3.3.2 Target Word Selection

Having fixed the sample size, the last parameter in the calculation algorithm is the target word lists. This is perhaps the most crucial decision, since if a word is not in the target list it cannot be expanded if it appears in a query. Also, if the word is not in the target list, it cannot be added as a result of expanding a query. Fixing the number of target words at 4,000, we experimented with the location of the target words. The target words were selected based on their frequency. The baseline location were the most frequent words, omitting the context words. Consider a frequency ordered list for the sample. Words 1 through 200 would be the context words and, for offset 0, words 201 through 4,200 would be the target words. Other offsets slid the target words down the frequency list, selecting 4,000 words whose frequency in the sample decreased as the offset increased. In all cases, the 4,000 target words selected from the frequency list were augmented with any missing, non-stopped query words. This was done to ensure that all important query words would be in the target word list. From Table 8, we see that offset 0 provided the best performance.

Offset	**0**	**2000**	**4000**	**6000**	**8000**
11-pt average	0.2003 (+6.3)	0.1885 (+0)	0.1916 (+1.6)	0.1946 (+3.2)	0.1901 (+0.8)

Table 8. 11-pt average for different target word frequency list location for 4000+ target words (WSJ database).

After we found the optimal target word location, we want to know how many target words are enough. Table 9 shows the average of 11-pt average for different numbers of target words (all using offset 0). Surprisingly, 4000+ target words give the best performance, with the added benefit of dramatically lower computation demands since creating the similarity matrix is O($N_t{}^2$) where N_t is the number of target words.

Target Words	**4000+**	**6000+**	**8000+**
first run	0.2049 (+8.7)	0.2006 (+6.4)	0.1993 (+5.7)
second run	0.1952 (+3.6)	0.1957 (+3.8)	0.1935 (+2.7)
third run	0.2015 (+6.9)	0.1988 (+5.5)	0.2017 (+7.0)
Average	0.2005(+6.4)	0.1984(+5.2)	0.1982(+5.1)

Table 9. 11-pt average for different target word size (WSJ database).

3.4 Query Expansion Experiments with the Wall Street Journal corpus

Having tuned the similarity calculation parameters, we then investigated how to best make use of the information in the similarity lists for query expansion.

3.4.1 Methods for Expanding Queries

In early work [Gauch & Chong, 1995], expanding using a similarity threshold alone seems very sensitive to the threshold chosen. Slight changes in the threshold could dramatically affect the number of words used to expand a given query word. We therefore experimented with expansion techniques which capped the number of words used to expand a given word alone and in combination with thresholds. In all, we tested four different query expansion methods:

1) for each query word which appears in the target list, add all words in the similarity list above some threshold. Table 10 shows the 11-pt average for different thresholds.
2) for each query word which appears in the target list, add a fixed number of words from the similarity list. (If there are fewer than that number in the similarity list, add as many as there are). Table 11 shows the 11-pt average for different numbers of expanded words.
3) add a threshold to method 2, i.e., add at most the fixed number of words, but only add those words which are above some threshold. Table 12 shows the 11-pt average for different similarity thresholds, each with a fixed number of 2.
4) add a higher threshold to method 3, i.e., add all words above a high threshold, but at most a fixed number of words above a lower threshold. Tables 13-15 give the average of the 11-pt averages for different combinations of higher and lower thresholds.

To clarify the differences in the expansion techniques, let's consider an example with three similarity lists:

status 1.0000 {role 0.3360} {strategy 0.3292} {system 0.3218} {proposal 0.3206} {problems 0.3206} {position 0.3143} {activities 0.3064} {policies 0.3057} {consequences 0.3024} {reform 0.3017}...

nuclear 1.0000 {military 0.4791} {economic 0.4602} {political 0.4224} {civil 0.3969} {defense 0.3876} {legal 0.3831} {drug 0.3471} {weapons 0.3458} {environmental 0.3413} {natural 0.3407} {air 0.3341} {communist 0.3278} {oil 0.3273} {medical 0.3193} {computer 0.3151} {iraqi 0.3150} {human 0.3141} ...

proliferation 1.000 { persian 0.2517} {decade 0.2005} ...

and the query:

status of nuclear proliferation

Using a threshold of 0.45 for method 1, a fixed number of 2 for method 2, a fixed number of 2 with a threshold of 0.30 for method 3 and a fixed number of 2 with a low threshold of 0.30 and a high threshold of 0.40 for method 4, we would get the following expanded queries:

1) *status 1.000 {} of nuclear 1.000 {military 0.4791} {economic 0.4602} proliferation 1.000 {}*

2) *status 1.0000 {role 0.3360} {strategy 0.3292} of nuclear 1.0000 {military 0.4791} {economic 0.4602} proliferation 1.000 { persian 0.2517} {decade 0.2005}*

3) *status 1.0000 {role 0.3360} {strategy 0.3292} of nuclear 1.0000 {military 0.4791} {economic 0.4602} proliferation 1.000*

4) *status 1.0000 {role 0.3360} {strategy 0.3292} of nuclear 1.0000 {military 0.4791} {economic 0.4602} {political 0.4224} proliferation 1.000*

3.4.2 Evaluating the Methods

threshold	0.40	0.50	0.60	0.70
11-pt average	0.1862 (-1.2)	0.1895 (+0.5)	0.1878 (-0.4)	0.1888 (+0.2)

Table 10. 11-pt average for different thresholds with expansion method 1 (WSJ database). Note: the baseline with no expansion was 0.1885.

max number	1	2	3	4	5
11-pt average	0.1977 (+4.9)	0.2003 (+6.3)	0.1990 (+5.6)	0.1913 (+1.5)	0.1887 (+0.1)

Table 11. 11-pt average for different thresholds with expansion method 2 (WSJ database).

threshold	0.20	0.22	0.24	0.26	0.28	0.30
11-pt avg.	0.2003 (+6.3)	0.2004 (+6.3)	0.2020 (+7.2)	0.2014 (+6.8)	0.2019 (+7.1)	0.1987 (+5.4)

Table 12. 11-pt average for a maximum number of 3 expansion words, different thresholds with expansion method 3 (WSJ database).

lower threshold	0.22	0.24	0.26	0.28	0.30
11-pt avg.	0.1987 (+5.4)	0.2028 (+7.6)	0.2031 (+7.7)	0.2035 (+8.0)	0.2019 (+7.1)

Table 13. 11-pt average for a maximum number of 3 expansion words, higher threshold of 0.42, different lower thresholds with expansion method 4 (WSJ database).

higher threshold	0.42	0.44	0.46	0.48	0.50	0.52
11-pt avg.	0.1987 (+5.4)	0.2022 (+7.3)	0.2031 (+7.7)	0.2020 (+7.2)	0.2006 (+6.4)	0.2002 (+6.2)

Table 14. 11-pt average for a maximum number of 3 expansion words, lower threshold of 0.22, different higher thresholds with expansion method 4 (WSJ database).

Having found good candidates for the lower threshold (0.24) and the higher threshold (0.46), we need to investigate the maximum number of words to add between the thresholds. These results are shown in Table 15.

maximum number	1	2	3	4	5	6
11-pt avg.	0.1955 (+3.7)	0.1950 (+3.4)	0.2049 (+8.7)	0.1994 (+5.8)	0.1990 (+5.6)	0.1971 (+4.6)

Table 15. 11-pt average for a lower threshold of 0.24, a higher threshold of).46, different maximum number of words with expansion method 4 (WSJ database).

Table 16 compares four different methods. Method 4 provides the best performance found to date on the WSJ corpus. It also makes intuitive sense: all words which are clearly similar to the query word (i.e., above the higher threshold) are added. However, at most a small number of words (3, to be exact) which are somewhat similar (i.e., above the lower threshold but not above the higher one) are added.

methods	One	Two	Three	Four
11-pt average (normalized)	0.1895 (+0.5)	0.2003 (+6.3)	0.2020 (+7.2)	0.2049 (+8.7)
11-pt average (not normalized)	0.1877 (-0.4)	0.1905 (+1.1)	0.1917 (+1.7)	0.1828 (-3.0)

Table 16. 11-pt average for different query expansion methods (WSJ database).

3.4.3 Effect of Normalization

The experiments in the previous section dealt with different methods of identifying the expansion words for each query term. However, once the words have been chosen by one of the four methods discussed, the weights on the expansion words (and adjustments, if any, to the weight of the original query term) have yet to be determined. Our initial approach was to treat the original query term as a concept which had a weight of 1.0. Then, when expansion words were added to the concept, they were given a weight equal to their value in the similarity list for the query word. All the weights for that concept (i.e., the original query term plus all expansion words) were then re-normalized to sum to 1.0. This was done so that the query would not be re-balanced to give more weight to a concept merely because it had many synonyms. To verify that normalization was necessary, we expanded queries with each of the four methods and reran the queries without normalizing the weights. Table 16 shows that normalization is extremely important for all of the expansion methods.

Consider Topic 203 in TREC 4. Expansion without normalization, yields

what is the economic 1.000 {political 0.5660} {military 0.4851} impact 1.000 {effect 0.5324} {role 0.3981} of recycling 1.000 {food 0.2403} {machinery 0.2254} tires 1.000 {cars 0.2783} {gas 0.2283}?

whereas with normalization we get

what is the economic 0.4875 {political 0.2759} {military 0.2365} impact 0.5180 {effect 0.2758} {role 0.2062} of recycling 0.6823 {food 0.1639} {machinery 0.1538} tires 0.6637 {cars 0.1847} {gas 0.1515}?

Because of the high similarity values of the words added by expansion, the "economic" and "impact" concepts get much higher weights without normalization relative to the more important concepts of "recycling" and "tires".

4. EXTENDING THE METHOD TO MULTI-DATABASE COLLECTIONS

Our experiments for TREC4 [Gauch & Chong, 1995] showed that analyzing the two databases in Category B separately performed better than treating the corpus as one big database. To extend that work, we created a similarity matrix for each of the seven databases in the TREC 5 Category A collection (AP, CR, FR88, FR94, FT, WSJ, and ZIFF). For comparison, we created a single similarity matrix from a sample taken across all the seven databases. We expanded each query by each of the seven matrices, in turn, and submitted the resulting queries to the entire collection. Table 17 summarizes the results.

matrix selected	**single matrix**	**best**	**worst**	**method A**	**method B**	**method C**
11-pt avg.	0.1079 (-1.9)	0.1353 (+23)	0.0781 (-29)	0.1014 (-7.8)	0.1046 (-4.9)	0.1130 (+2.7)
avg. rank	N/A	1.0	7.0	4.1	3.6	3.4

Table 17. 11-pt average for different similarity matrices (TREC5 Category A collection). Note: the best and worst matrices were manually selected after the queries were run. Baseline performance with no expansion was 0.1100.

Examining the results, we rank ordered the matrices for each query based on the 11-pt average produced for the expanded query it created. When the best matrix is used to expand each query, performance increases dramatically (23%). However, when the worst matrix is used, performance decreases just as dramatically (-29%). Clearly, selecting the correct matrix in a multi-database collection is of crucial importance. However, avoiding the issue by creating one matrix for the entire collection is not viable. The differing word usages cloud the issue, and the resulting matrix causes a slight degradation (-1.9%).

We next turned our attention to the issue of automatically selecting the most appropriate matrix to expand a query, investigating three different methods all based upon the frequency of the query terms in different databases. Method A sums together the relative frequencies of the (non-stopped) query terms in each database (i.e., the frequency of the term in the database sample normalized by the frequency of the term in all database samples). Method B is identical, but normalizes by the most frequent word in the sample rather than the total frequency of the query term in the samples. Method C is a combination of the other two, where both normalizations are done. In general, all the methods select a mid-rank matrix, rather than a high-ranking one. Clearly more work on matrix selection is necessary to approach the performance possible when the correct matrix is used. Perhaps examining the similarity matrices themselves (e.g., the similarity matrix with the most matches or the best matches) or analyzing the results of the unexpanded query (select the matrix for the database which returns the most documents) would be effective.

5. CONFIRMING THE RESULTS ON SPECIALIZED CORPORA

The TIPSTER collections tend to contain diverse papers which are written in general English. To confirm that this approach is particularly well-suited to smaller, more specialized corpora which are by and large written in their own sub-languages, we tested our approach on the Cystic Fibrosis database (Shaw et al, 1991), a collection of all papers (1,239) indexed by the term CYSTIC FIBROSIS in MEDLINE (1974-1979).

Running the similarity calculation as determined in Section 2, and Method 4 with a lower threshold of 0.5 and a higher threshold of 0.7 yielded the following results:

Relevance Score	**>= 1**	**>= 2**	**>= 3**	**>= 4**	**>= 5**	**6**
11-pt avg. (no exp)	0.2905	0.3130	0.3313	0.3373	0.3252	0.2834
11-pt avg. (expand)	0.3732 (+28.5)	0.4000 (+27.8)	0.4161 (+25.6)	0.4205 (+24.7)	0.3784 (+16.4)	0.3023 (+6.7)

Table 18. 11-pt average for different cumulative relevance scores (Cystic Fibrosis collection).

The Cystic Fibrosis file has relevance scores of 0 (not relevant), 1 (somewhat relevant), or 2 (highly relevant) from each of three judges. The relevance scores in Table 18 represent the sums of the scores of all three judges. The first column of the table (>=1) shows the results if we consider documents with a cumulative score of 1 or more to be relevant, etc. We see a monotonically decreasing improvement in retrieval (from a 28.5% gain to a 6.7% gain) as we narrow our interpretation of relevance by requiring documents to have a higher cumulative score. This makes intuitive sense - expanding the queries is likely to find a broad selection of somewhat relevant documents. Still, with an average improvement of 21.6%, this method seems to work extremely well a small, specialized corpus (4.9 MB).

CONCLUSIONS

Our goal is to develop automatic query expansion in order to provide conceptual retrieval. We have implemented an analysis technique which takes word order into account to automatically identify similar words from an untagged corpus. We extensively tested and tuned this technique on databases from the TIPSTER collection. Then, we investigated how to best make use of the similarity information during query expansion in a single database, coming up with a two-tiered approach which add all highly similar words and up to a small, fixed number of somewhat similar words. This work is currently being extended to multi-database collections, and it shows great promise in that context. However, automatic selection of the best matrix is still under investigation. Finally, particularly good results were achieved on a small, specialized database.

BIBLIOGRAPHY

Buckley, C. (1985). Implementation of the SMART Information Retrieval System. *Technical Report 85-686*, Computer Science Department, Cornell University, Ithaca, New York.

Church, K. W., & Hanks, P. (1990). Word Association Norms, Mutual Information and Lexicography. *Computational Linguistics, 16*(1), pp. 22-29.

Croft, W.B., Cook, R. & Wilder, D. (1995). Providing Government Information on the Internet: Experiences with THOMAS. In *Digital Libraries Conference DL '95*, pp. 19-24.

Crouch, C. & Yang, B. (1992). Experiments in Automatic Statistical Thesaurus Construction. In *Proc. 15th Ann. International ACM SIGIR Conf.*, Copenhagen, Denmark, ACM Press, pp. 77-88.

Finch, S., & Chater, N. (1992). Bootstrapping Syntactic Categories Using Statistical Methods. In W. Daelemans & D. Powers (Ed.), In *1st SHOE Workshop*, Tilburg, The Netherlands, pp. 229-235.

Gauch, S. & Chong, M. (1995) Automatic Word Similarity Detection for TREC 4 Query Expansion. In *4th Text Retrieval Conf. (TREC-4)*, NIST #500-236, Gaithersburg, MD, pp. 527-536.

Gauch, S., & Smith, J.B. (1993). An Expert System for Automatic Query Reformulation. *J. of the Amer. Society of Inf. Sci.*, *44* (3), pp. 124-136.

Gauch, S., & Smith, J.B. (1991). Search Improvement via Automatic Query Reformulation. *ACM Trans. on Information Systems*, *9* (3), pp. 249-280.

Gauch, S. & Wang, J. (1996) Automatic Word Similarity Detection for TREC 5 Query Expansion. In *5th Text Retrieval Conf. (TREC-5)*. Gaithersburg, MD, (to appear).

Grefenstette, G. (1992). Use of Syntactic Context to Produce Term Association Lists for Text Retrieval. In *Proc. 15th Ann. International ACM SIGIR Conf.*, Copenhagen, Denmark, ACM Press, pp. 89-97.

Harman, D. (1992). Relevance Feedback Revisited. In *Proc. 15th Ann. International ACM SIGIR Conf.*, Copenhagen, Denmark, ACM Press, pp. 1-10.

Liddy, E.D., & Myaeng, S.H. (1993). DR-LINK's Linguistic-Conceptual Approach to Document Detection, In *1st Text Retrieval Conf. (TREC-1)*, NIST #500-207, pp. 113-129.

Myaeng, S. H., & Li, M. (1992). Building Term Clusters by Acquiring Lexical Semantics from a Corpus. In Y. Yesha (Ed.), *CIKM-92*, Baltimore, MD, ISMM, pp. 130-137.

Qiu, Y. & Frei, H.P. (1993). Concept Based Query Expansion. In *Proc. 16th Ann. International ACM SIGIR Conf.*, Pittsburgh, PA, ACM Press. pp. 160-169.

Schütze, H. & Pedersen, J. (1994). A Cooccurrence-Based Thesaurus and two Applications to Information Retrieval. In *Intelligent Multimedia Information Retrieval Systems RIAO '94*, New York, NY, pp. 266-274.

Shaw, W.M., Jr., Wood, J.B., Wood, R.E. & Tibbo, H.R. (1991). The Cystic Fibrosis Database: Content and Reserach Opportunities. *Library and Information Science Research*, 12, pp. 347-366.

Voorhees, E.M. (1994). Query Expansion Using Lexical-Semantic Relations, In *13th Ann. International ACM SIGIR Conf.*, Dublin, Ireland, ACM Press, pp. 61-69.

Xu, J. & Croft, W.B. (1996). Query Expansion Using Local and Global Document Analysis, In *15th Ann. International ACM SIGIR Conf.*, Zurich, Switzerland, ACM Press, pp. 4-11.