To appear in: ACM Transactions on Information Systems.

A Corpus Analysis Approach for Automatic Query Expansion and its Extension to Multiple Databases[1]

Susan Gauch, Jianying Wang and Satya Mahesh Rachakonda
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 databases which are not tagged with part-of-speech labels. Using these similarities, user queries are automatically expanded, resulting in conceptual retrieval rather than requiring exact word matches between queries and documents. We are able to achieve a 7.6% improvement for TREC5 queries and up to a 28.5% improvement on the narrow-domain Cystic Fibrosis collection. 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. Various techniques to select the appropriate matrix for a particular query are analyzed and a 4.8% improvement in the results is validated.

[1]This is an extended version of a paper presented at the Conference on Information and Knowledge Management, Nov. 1997 (CIKM '97).

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.

Query specific 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.

Corpus specific 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; Qiu & Frei, 1993], i.e., words which often occur together in documents are assumed to be similar. These thesauri can then be used for automatic or manual query expansion. A related approach, Latent Semantic Indexing [Deerwester et al, 1990], does singluar value decomposition on the term-document occurrence patterns to reduce the indexing space into a smaller number of "semantic" factors. Documents, and queries, are then represented and matched based on these factors.

Based on studies that show that the more often words can be substituted into the same context, the more similar they are in meaning [Mill & Charles, 1991], other approaches look at word usage within documents. While not a new idea [Sparck Jones, 1971], modern computers make this approach feasible. While some incorporate syntactic analysis [Grefenstette, 1992], most 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). Varying 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 & Pedersen, 1994]; 18-30% on smaller corpora [Qiu & Frei, 1993]). These approaches are computationally intensive, but the computations are done once per database. The only component done on a per query basis is the actual query expansion itself. Also, because the information is built from the specific text collection, the related terms are automatically tuned for the particular database being searched. On weakness of corpus specific approaches is that they cannot determine term relationships which occur between words which are used in the corpus and those which are used by a different community (e.g., the Congressional Record uniformly uses the term "senior citizen" whereas users might use the term "elderly" in their queries).

<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 it includes 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. Similar to [Schütze & Pedersen, 1994] and [Jing & Croft, 1994], we use fine-grained information about word contexts to create an association thesaurus. In contrast, our windows are an order of magnitude smaller and we consider the order of occurrence of the words within the window (see Section 2). Using this approach, we are able to achieve a 7.6% improvement for TREC5 queries and up to a 28.5% improvement on the narrow-domain Cystic Fibrosis collection (see Section 4). Section 5 considers the extension of this approach to multiple databases.

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. In addition, we use the highest frequency words as context words, which are generally removed as stopwords by other approaches. These words occur most frequently, and thus provide more statistical information in smaller samples. In addition, they provide *ad hoc* part-of-speech information. 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 has one element for each *context word*, the words whose appearance in the window surrounding the target words is recorded. Generally, the context words are the most frequent words in the database.

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**	**Word's Position in Context Vector**	**Context Sub-Vector**
1	-2	"The"	4	(0, 0, 0, 1, 0)
	-1	"black"	2	(0, 1, 0, 0, 0)
	+1	"barked"	N/A	(0, 0, 0, 0, 0)
	+2	"very"	5	(0, 0, 0, 0, 1)

Table 1. The context sub-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 sub-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 formed by adding the context vectors for all occurrences together to form:

(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:

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

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 normalized 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.

2.2 Efficiency Concerns

The time efficiency of building the context vectors is O(N_S) where N_S is the number of word instances in the sample. However, computation time is dominated by building the similarity matrix from the context vectors which is O($N_t{}^2$) where N_t is the number of target words. The space needed to store the context vectors is O(N_t *N_c), which is independent of the sample size.

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. The retrieval runs using normalized term weights for document terms (*lnc*) and the unnormalized weights for the query terms (*ltc*). We experimented with different databases, different similarity calculation parameters and different expansion techniques.

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) and the San Jose Mercury (SJM) with 48 queries; 2) WSJ (0.25 GBs) with 45 short, ad hoc queries from TREC4 and 45 short, ad hoc queries from TREC5; and 3) the Cystic Fibrosis database (4.9 MB), a collection of 1,239 papers cystic fibrosis related papers from MEDLINE with 100 associated queries [Shaw et al, 1991].

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

3.2.1 Tuning the Parameters

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]. We expanded each query with all words above a fixed similarity threshold. For each combination of the three factors, we evaluated a range of different similarity thresholds, 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 using all 48 queries.

Sample size 10% (38 MB)			
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% (76 MB)			
Window Size / Context words	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% (114 MB)			
Window Size / Context words	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. Baseline: unexpanded queries yield an 11-pt average of 0.1791.

As we expected, adjusting the parameters one by one produced only modest improvements, if any. However, analyzing the parameter settings individually was the first step to finding a combination of the parameters that work well. From these experiments, we concluded that a window size of 7, 200 context words, and a sample size of 20% (76 MB) provided the best fixed point in a highly multi-dimensional space with respect to results and efficiency. We were surprised to note that the larger 30% (114 MB) actually degraded performance. We are not sure what causes this effect, however larger samples may be more susceptible to the effects of ambiguity since they are more likely to include rare uses for a particular target word as well as the common uses, degrading the context vectors.

3.2.2 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 3. 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). Baseline: 0.1791.

From Table 3, 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 positional 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. 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. A more complete description of the experimental results appears in [Gauch & Wang, 1997]

3.3.1 Sample Selection

In Table 4, the average of 11-pt averages for different size of sample databases is presented. The trend seems to be 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	0.1938	0.2007	0.2020	0.1989
second run	0.1885	0.2002	0.1982	0.2015
third run	0.2005	(0.1944	0.2010	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 4. The 11-pt averages for different sample sizes (3 samples per size) from the WSJ database. Baseline: 0.1884.

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 selection of the target words based on their frequency. 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 5, we see that offset 0 provided the best performance. Intuitively, the benefit of using relatively frequent non-stopwords as possible expansion terms can be explained because adding a common synonym for a query term is likely to be of more benefit than adding a rare synonym for a query term.

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 5. 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 6 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. We interpret this result to mean that adding lower frequency words to the target list adds more noise than value, possibly because we have insufficient information about the lower frequency words to produce accurate .

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 6. 11-pt average for different sizes of target word lists (WSJ database).

3.3.3 Representative Results

To give a feel for the types of similarity information generated by this approach, we will present some representative results. Some similarities seem intuitively correct, others less so. However, the proof of the validity of this approach is not a qualitative examination of the similarity matrix, but rather the quantitative search improvements presented in the next section.

accord	agreement (0.553) pact (0.509) arrangement (0.424) treaty (0.383) talks (0.348) merger (0.346) settlement (0.333) transaction (0.331) bill (0.322) ...
acquire	sell (0.459) buy (0.435) provide (0.380) eliminate (0.374) convert (0.373) acquired (0.368) build (0.361) purchase (0.357) receive (0.350) ...
acquiring	acquire (0.335) buying (0.2799) joining (0.277) expanding (0.273) issuing (0.2731) making (0.2703 selling (0.2556) using (0.250) sell (0.234) ...
analyst	economist (0.568) trader (0.501) strategist (0.460) consultant (0.428) official (0.391) spokesman (0.354) specialist (0.352) adviser (0.348) ...

The above excerpts illustrate that, due to the influence of positional information in the similarity calculation, words tend to group along parts of speech. For example, the similarity list for *acquire* contains the *buy* and *sell* whereas the similarity list for *acquiring* contains *buying* and *selling.* In addition, various forms of the same word appear together, e.g., *acquire* and *acquiring,* which may explain why stemming provided no improvement in Section 3.2.2. We get a partial stemming effect automatically.

3.4 Query Expansion Experiments with the Wall Street Journal corpus

3.4.1 Query Expansion Technique

Having tuned the similarity calculation parameters, we then investigated how to best make use of the information in the similarity lists for query expansion. 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 [Gauch and Wang, 1997]. 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.
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).
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.
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.

We found that Method 4 (with a lower threshold of 0.24 and a higher threshold of 0.46) provided the best performance. 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.

Method	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 7. 11-pt average for different query expansion methods (WSJ database).

3.4.2 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 7 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. VALIDATING THE RESULTS

We performed two types of validation experiments. In the first, we used the most promising similarity matrix calculations and query expansion techniques for the Wall Street Journal corpus using 45 new queries from TREC5. This resulted in an overall 11 point average of 0.1325 compared to a baseline of 0.1231, an improvement of 7.6%.

The TIPSTER collections tend to contain diverse papers which are written in general English. Our second validation experiment used an entirely different type of corpus to show that the results were applicable across a broad range of collections. We believe that this approach is likely to be of most benefit to corpora within specialized sub-domains. The word usage in such sub-domains tends to have specialized words and specialized word usages which our technique can exploit. 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 for query expansion (higher threshold 0.7; lower threshold 0.5) 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 8. 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 8 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% across all relevance judgment values, this method seems to work extremely well in a small, specialized corpus.

5. APPLYING THE APPROACH TO MULTIPLE DATABASES

5.1 Effect of Multi-databases

In a multi-database environment, each of the databases may have a different domain of interests, resulting in different word usage and different word similarities. This means that it is critical to choose the similarity matrix which provides the most appropriate query expansion words. In this section, we summarize a series of experiments aimed at improving search quality by automatically selecting an appropriate matrix where several different candidates exist.

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 large 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 9 summarizes the results.

matrix selected	**single matrix**	**best**	**average**	**worst**
11-pt avg.	0.1009 (-5.6)	0.1365 (+27.7)	0.1020 (-4.6)	0.0716 (-33.0)
avg. rank	N/A	0.0	3.0	6.0

Table 9. 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: 0.1069.

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%).

5.2 Automatically Selecting the Similarity Matrix

We next turned our attention to the issue of automatically selecting an appropriate matrix to expand a query. There are essentially three types of information available on which to base the similarity matrix selection:

1. Examination of result set returned by running the query on the collection of all databases, paying attention to the contributions of the various component databases in the composite retrieval set.
2. Usage of word frequency information obtained directly from each of the databases.
3. Examination of the contents of the similarity matrices themselves.

Different algorithms based on the above three categories are explored singly and in combination.

5.2.1 Experimental Technique

The following experiments were conducted using TREC queries 251 - 275. The most promising method was then validated using TREC queries 276 - 300. To evaluate the performance of a particular selection technique, we pre-calculated, for each of the 25 test queries, the 11-point average obtained when the query was expanded with each of the seven candidate matrices in turn. Based on those results, we then assigned, for each query, a rank order to each similarity matrix from 0..6 where 0 is the best matrix and 6 is the worst matrix. For each technique evaluated, we examined the rank order of the similarity matrix chosen by that technique and averaged the rank orders over all queries. This value is called the DataBase Selection Average (DBSA). An optimal algorithm would select the best matrix in all cases and produce an average rank over all queries (i.e., DBSA) of 0.0. Randomness would select an average matrix overall, with a rank order of 3.0. The goal of our experiments is to find a technique which produces the lowest DBSA. These experiments are discussed in more detail in [Gauch & Rachakonda, 1997].

5.2.1 Experiments based on Examination of the Retrieval Set

The initial queries are sent directly without any modifications/expansion to the collection of all the seven databases under consideration. The result set contains a rank ordered list of the identifiers of the best-matching documents from which the corresponding sub-database can be determined. In each case, the similarity matrix derived from the database which contributed the most to the retrieval set was selected. Several methods were evaluated, each with a variety of normalization factors. All the four methods were evaluated with a limit on how many documents from the result set were considered. Results from the following methods are summarized in Table 10.

[Method 1.1] Number of documents returned.

[Method 1.4] Number of documents normalized by number of documents in the database.

[Method 1.5] Average rank of documents returned.

[Method 1.8] Average rank of documents normalized by number of documents in the database.

Maximum retrieval set rank considered	***Number documents returned***	***Number normalized by number of doc't in DB***	***Avg. rank of documents returned***	***Avg. rank normalized by number of doc't in DB***
10	2.88	3.02	2.76	2.92
20	2.84	2.76	2.68	2.88
30	2.72	2.76	2.68	2.88
40	2.64	2.60	2.68	2.54
50	2.64	2.62	2.36	2.72
100	2.64	2.44	2.32	2.62
150	2.44	2.48	2.36	2.58
200	2.32	2.54	2.60	2.62
500	2.76	2.68	2.78	2.80
1000	2.80	2.88	2.96	2.88

Table 10. Database Selection Average (DBSA) for methods based on examination of the retrieval set.

It appears that normalization does not help, which is not surprising. We are interested in getting the largest amount of relevant information regardless of the size of the database that contains it.

[Method 1.9] Combination of number of documents with average rank of documents normalized. Finally, we evaluated a method that combined the number of documents with the an average rank as follows:

$$Score = \left(m \times \frac{volume}{1000}\right) \div \left(p \times \frac{1}{rank}\right)$$

The influence of each factor was varied by varying the values for *m* and *p*. However, *m* + *p* always totaled 100%. The number of documents, *n*, is normalized by the total size of the retrieval set to produce a number between 0 and 1. Results are presented in Table 11.

m - p	***n = 50***	***n = 100***	***n = 150***
0% - 100%	2.36	2.32	2.36
20% - 80%	2.32	2.28	2.32
40% - 60%	2.28	2.28	2.36
50% - 50%	2.36	2.32	2.36
60% - 40%	2.44	2.36	2.42
80% - 20%	2.62	2.44	2.42
100% - 0%	2.64	2.64	2.44

Table 11. Database Selection Average (DBSA) for method 1.9 combining number of retrieved documents with average rank.

This produces the best results so far, consistently selecting, on average, a matrix for expansion that is one better than randomness would predict. However, there is a serious drawback to this technique in that queries must be run twice: the original query must be submitted to generate a result set used to identify a promising similarity matrix and then the expanded query is submitted.

5.2.2 Experiments Based on Word Frequency Information

The techniques in this section revolve around using information about the frequency of the query words in the different component databases. The selection methods evaluated are described below. In each case, the similarity matrix derived from the database with the maximum score is selected. The results are summarized in Table 13.

[Method 2.1] Absolute sum of query word frequencies in the database.

[Method 2.2] Sum of query word frequencies normalized by the frequency of the most frequent word in the database.

[Method 2.3] Sum of query word frequencies each normalized to its own total frequency in all databases. This method allows each query word an equal weight in the database selection process. This corrects a problem with methods 2.1 and 2.2 in which rare words contribute only slightly to the database selection process.

[Method 2.5] Absolute sum of query word frequencies multiplied by idf.

Method	**DBSA**
[2.1] Sum of query word frequencies (absolute)	2.52
[2.2] Sum of query word frequencies normalized to maximum word frequency in DB.	3.16
[2.3] Sum of query word frequencies with each word normalized to it's sum of individual frequencies over all DBs.	2.19
[2.5] Sum of query word frequencies weighed by idf	2.32

Table 13. Database Selection Average (DBSA) for methods based on word frequencies.

We see the most promising result so far, Method 2.3, which normalizes the individual query words so that each contributes equally to the database selection algorithm. However, this method is not practical in general since it relies on knowledge of the frequencies of the query words in each of the component databases, information that is usually not available.

5.2.3 Experiments based on Similarity Matrix Contents

The final source of information upon which to base the similarity matrix selection is not information about the database itself (via retrieval sets as in Section 5.2.1 or frequency lists as in Section 5.2.2) but rather through use of the information contained in the similarity matrices themselves. A wide variety of algorithms were evaluated in this category, among them the following:

[Method 3.1] Number of words above a high threshold. For each query word, add the number of all words in the similarity matrix above a given threshold. A fairly high threshold is needed to avoid being overly influenced by one query word with many slightly similar words. Best results (DBSA = 2.06) are found with a Higher Threshold near 0.46 [Gauch and Rachakonda, 1997].

[Method 3.2] Maximum number of words above a lower threshold. By adding a cap on the maximum number of words a single query word can add, a lower threshold may be used successfully. Best results are obtained with a threshold of 0.35 and maximum of 3 - 5 words [Gauch and Rachakonda, 1997].

[Method 3.3] Combination of all words above a higher threshold with a maximum number above a lower threshold. Following the promising results of using the Higher Threshold and Lower Thresholds independently, we combined them. We achieved the first DBSA below 2.0 (1.96) which occurred at a Higher Threshold of 0.50, a Lower Threshold of 0.32 and maximum number of words of 4. These results are presented in Table 14.

	High = 0.46			**High = 0.48**			**High = 0.50**		
Low	**m = 3**	**m = 4**	**m = 5**	**m = 3**	**m = 4**	**m = 5**	**m = 3**	**m = 4**	**m = 5**
0.32	2.12	2.12	2.12	2.32	2.12	2.08	2.08	1.96	2.08
0.35	2.12	2.08	2.08	2.12	2.08	2.08	2.08	2.08	2.08
0.37	2.14	2.14	2.14	2.12	2.08	2.12	2.12	2.24	2.24

Table 14. Database Selection Average (DBSA) for method 3.3 based on adding all words above a Higher Threshold and a maximum number above a Lower Threshold.

This method turns out to be the best that we have been able to find. We analyzed the experimental data more (in order to determine parameter settings for future use) to determine the effect of individual factors. This data is summarized in Table 15.

Effect of Higher Similarity Threshold on DBSA			
Higher Threshold value	0.46	0.48	0.50
Average DBSA	2.1288	2.1244	2.1067
Effect of Lower Similarity Threshold on DBSA			
Lower Threshold value	0.32	0.35	0.37
Average DBSA	2.1111	2.0889	2.1489
Effect of Number of Words added on DBSA			
Number of Word added	3	4	5
Average DBSA	2.1356	2.1000	2.1133

Table 15. Effect of Higher and Lower Thresholds and Number or Words on DBSA.

Thus the best setting of parameters for this method so far are: a Higher Threshold of 0.50, a Lower Threshold of 0.35 and a maximum words of 4.

[Method 3.5] Sum of similarities for all words above a high threshold. Rather than just count the number of words above a given threshold, the similarity values themselves are summed. The best results (2.24) were obtained with a threshold of 0.50 [Gauch and Rachakonda, 1997]..

[Method 3.6] Sum of similarities of for a maximum number of words above a low threshold. The best results (2.24) were obtained with a threshold of 0.30 and a maximum number of words of 3 or 4 [Gauch and Rachakonda, 1997]..

[Method 3.7] Combination of the sums of similarities for all words above a higher threshold and a maximum number of words above a lower threshold. The best results (2.08) occurred at a higher threshold of 0.50, a lower threshold of 0.30 and maximum number of words of 4. These results are presented in Table 16.

	High = 0.48			**High = 0.50**			**High = 0.52**		
Low	**m = 3**	**m = 4**	**m = 5**	**m = 3**	**m = 4**	**m = 5**	**m = 3**	**m = 4**	**m = 5**
0.25	2.24	2.26	2.24	2.28	2.42	2.36	2.28	2.32	2.32
0.30	2.12	2.12	2.14	2.12	2.08	2.10	2.12	2.14	2.14
0.35	2.18	2.18	2.22	2.24	2.24	2.24	2.28	2.42	2.42

Table 16. DBSA for Method 3.7 based on combination of Higher and Lower Similarity Thresholds.

5.2.4 Consolidated Results

Finally, we summarize the results for all methods discussed in this section in Table 17. For methods with multiple parameters, we present the results achieved by the best parameter settings.

Method Used	DBSA	11-point Average	% improvement over baseline
Baseline: Unexpanded	N/A	0.1069	N/A
Single Matrix	N/A	0.1009	-5.61
Worst Matrix	6.00	0.0716	-33.02
Average Matrix	3.00	0.1020	-4.58
Best Matrix	0.00	0.1365	+27.69
Method 1.1	2.32	0.1084	+14.03
Method 1.4	2.44	0.1045	-2.25
Method 1.5	2.32	0.1099	+2.81
Method 1.8	2.54	0.0994	-7.02
Method 1.9	2.28	0.1107	+3.55
Method 2.1	2.52	0.1006	-5.89
Method 2.2	3.16	0.0789	-26.19
Method 2.3	2.19	0.1132	+5.89
Method 2.5	2.32	0.1101	+2.99
Method 3.1	2.06	0.1136	+5.90
Method 3.2	2.08	0.1121	+4.86
Method 3.3	1.96	0.1196	+11.88
Method 3.5	2.32	0.1097	+2.62
Method 3.6	2.24	0.1114	+4.21
Method 3.7	2.08	0.1128	+5.5.2

Table 17. Consolidation of the best results for methods 1.1 - 3.7.

5.2.5 Validation of Results

The best method discovered through our series of experiments was method 3.3 which uses a combination of higher and lower thresholds. It produced a DBSA of 1.96 which resulted in a solid 11.88% improvement over the baseline matrix which is created by sampling all databases at 20% and creating a single matrix. We validated our results by applying this technique to a different collection of queries, TREC queries 276-300 (baseline: 0.1089). A DBSA of 1.92 was achieved which translated into an increase of 4.78% was achieved. While the 11-point average improvement found was not as strong as in the training set, the DBSA was comparable or slightly better. It seems that the similarity selection algorithm was working at least as well, but that the queries in the testing set were perhaps less amenable to improvement by expansion.

6. CONCLUSIONS AND FUTURE WORK

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 approach was able to improve the query results on both the large, broad Wall Street Journal corpus (7.6%) and the small, specialized Cystic Fibrosis corpus (6.7% - 28.5%). This work has been extended to multi-database collections, specifically the seven databases that comprise the TREC5 Category A collection. Our results show that creating a similarity matrix for each of the sub-collections can improve performance (5 - 10%) when the similarity matrix for expanding each query is automatically selected. Techniques which examine the similarity matrices themselves work as well or better than other techniques and have the benefit of not requiring queries to be run twice or having access to information about word frequencies in possibly remote databases.

In future, we wish to investigate the scope of the applicability of the similarity matrices. In particular, we need to investigate when a similarity matrix produced from one corpus be used to expand queries sent to a related corpus. Also, we need to consider the applicability of this approach to dynamic collections. We expect that the amount of data added will not affect performance (we only sample a small percentage of the larger collections as it is), but the ability to efficiently add information about new, important terms and new relationships between terms over time requires study.

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.

Deerwester, S., Dumai, S.T., Furnas, G.W., Landauer, T.K., Harshman, R. (1990). Indexing by Latent Semantic Analysis, *J. of the Amer. Society of Inf. Sci.*, *41* (6), pp. 391-407.

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. & Rachakonda, S. (1997). Experiments in Automatic Similarity Matrix Selection for Query Expansion. *Information and Telecommunication Technology Center Technical Report:* ITTC-FY97-TR-11100-3. University of Kansas.

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).

Gauch, S. & Wang, J. (1997). Tuning a Corpus Analysis Approach for Automatic Query Expansion. *Information and Telecommunication Technology Center Technical Report:* ITTC-FY97-TR-11100-2. University of Kansas.

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.

Jing, Y. & Croft, W.B. (1994). An Association Thesaurus for Information Retrieval, In *Intelligent Multimedia Information Retrieval Systems RIAO '94,* New York, NY, pp. 146-160.

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.

Miller, G. A., Charles, W. G. (1991). Contextual Correlates of Semantic Similarity, In *Language and Cognitive Processes, 6* (1), pp. 1-28.

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.

Sparck Jones, K. (1971). *Automatic Keyword Classification fo Information Retrieval,* Butterworths, London.

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.