A Corpus Analysis Approach for Automatic Query Expansion and Its Extension to Multiple Databases

SUSAN GAUCH, JIANYING WANG, and SATYA MAHESH RACHAKONDA
University of Kansas

Searching online text collections can be both rewarding and frustrating. While valuable information can be found, typically many irrelevant documents are also retrieved, while 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 TREC 5 queries and up to a 28.5% improvement on the narrow-domain Cystic Fibrosis collection. This work has been extended to multidatabase collections where each subdatabase 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.

Categories and Subject Descriptors: H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Linguistic processing*; *Thesauruses*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Query formulation*

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: Query expansion

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

This is an extended version of a paper presented at the Conference on Information and Knowledge Management, Nov. 1997 (CIKM '97).
Authors' address: Electrical Engineering and Computer Science, University of Kansas, 2291 Irving Hill Road, Lawrence, KS 66045–2969; email: sgauch@eecs.ukans.edu.

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 and 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 and 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 TREC 3 and TREC 4 corpora. The main drawback to these approaches is that there is a reasonable amount of computation that takes place after the user submits a 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 and 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 and Yang 1992; Qiu and 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 [Miller and Charles 1991], other approaches look at word usage within documents.

While within-document word usage is not a new idea [Sparck Jones 1971], modern computers make this approach feasible. While some incorporate syntactic analysis [Grefenstette 1992], most look for cooccurrence patterns of words [Schütze and Pedersen 1994] or noun phrases [Jing and 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 TREC 3 and TREC 4 respectively [Xu and Croft 1996]; 5% on TIPSTER Category B [Schütze and Pedersen 1994]; 18–30% on smaller corpora [Qiu and 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. One 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).

Language-specific terms may be found from generally available online thesauri which are not tailored for any particular text collection. Liddy and Myaeng [1993] use the Longman's Dictionary of Contemporary English, a semantically coded dictionary. Voorhees [1994] used WordNet [Miller and Charles 1991], 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 and Pedersen [1994] and Jing and 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 TREC 5 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 and Chater 1992] that creates a matrix of term-term similarities. For words to be considered similar, they need not actually cooccur; 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 seven-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 subvectors, 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 **dog** barked very loudly.

(2) A brown **dog** barked very loudly.

As shown in Table I, the context vector for "dog" in sentence (1) is formed by concatenating the context subvectors for each of the four context positions:

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

Table I. The Context Subvectors for Each of the Four Context Positions Around the Occurrence of the Target Word "dog" in Sentence (1)

Sentence	Context Position	Observed Word	Word's Position in Context Vector	Context Subvector
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)

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 the following 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 and 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 of the context word and the target word, f_c and f_W respectively. 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 and 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 $O(N_t * N_c)$, where N_c is the number of context words, which is independent of the sample size.

3. EVALUATION

To incorporate the results of the corpus analysis into an existing retrieval engine, SMART Buckley 1985 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 used 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 three collections: (1) TREC 4 Category B (0.38GBs) which is comprised of The Wall Street Journal (WSJ) and the San Jose Mercury (SJM) with 48 queries; (2) WSJ (0.25GBs) with 45 short, ad hoc queries from TREC 4 and 45 short, ad hoc queries from TREC 5; and (3) the Cystic Fibrosis database (4.9MB), a collection of 1,239 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 and 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 II reports the best performing threshold and the associated 11-point average. These tests were carried out on the TREC 4 category B collection using all 48 queries.

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 toward 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% (76MB) provided the best fixed point in a highly multidimensional space with respect to results

Table II. Experimental Results for Context Words, Window Size, and Sample Size on the TREC 4 Category B Collection (baseline: unexpanded queries yield an 11-point average of 0.1791)

Sample Size 10% (38MB)			
Window Size	5	7	9
Context Words			
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% (76MB)			
Window Size	5	7	9
Context Words			
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% (114MB)			
Window Size	5	7	9
Context Words			
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)

and efficiency. We were surprised to note that the larger 30% (114MB) 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.

From Table III, 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 six 200-element position vectors concatenated) we still see a decrease in the retrieval results, albeit not as dramatic.

Table III. The Average 11-point average for Different Matrix Calculation Techniques (TREC 4 Category B collection: 20% sample, window size 7, 200 context words, 4000 target words, expansion threshold of 0.38; baseline: 0.1791)

Calculation Method	As Described	With Stemming	No Position Vectors (200)	No Position Vectors (1200)
11-point average	0.1887 (+5.4)	0.1880 (-5.0)	0.1382 (-22.8)	0.1727 (-3.6)

Table IV. The 11-point averages for different sample sizes (3 samples per size) from the WSJ database. Baseline: 0.1884

Sample Size	5% (12.5MB)	10% (25MB)	20% (50MB)	30% (75MB)
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

3.3 Similarity Matrix Calculation Experiments with The Wall Street Journal Corpus

The TREC 4 Category B corpus consists of two subcollections: 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 subcollections, we conducted further experiments on the WSJ subcollection alone. In particular, we reexamined 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 and Wang [1997].

3.3.1 *Sample Selection*. In Table IV, the average of 11-point 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-point average tends to be stable when the size of the sample database is above 20%.

The 20% (50MB) sample performs almost as well as the 30% (75MB) 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

Table V. Eleven-Point Average for Different Target Word Frequency List Location for 4000+ Target Words (WSJ database)

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

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, nonstopped query words. This was done to ensure that all important query words would be in the target word list.

From Table V, we see that offset 0 provided the best performance. Intuitively, the benefit of using relatively frequent nonstopwords 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.

After we found the optimal target word location, we wanted to know how many target words are enough. Table VI shows the average of the 11-point 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 context vectors.

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)...

Table VI. Eleven–Point Average for Different Sizes of Target Word Lists (WSJ database)

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)

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 and 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

Table VII. Eleven-Point Average for Different Query Expansion Methods (WSJ database)

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

higher threshold) are added. However, at most a small number of words (three, to be exact) which are somewhat similar (i.e., above the lower threshold but not above the higher one) are added.

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 renormalized to sum to 1.0. This was done so that the query would not be rebalanced 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 VII shows that normalization is extremely important for all of the expansion methods.

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

> "*what is the economic 1.000 {political 0.5660} {military 0.4851} impact 1.000 {effect 0.52324} {role 0.3981} of recycling 1.000 {food 0.2403} {machinery 0.2254} tires 1.000 {cars 0.2783} {gas 0.2283}?*"

However, with normalization we get something else:

> "*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 TREC

Table VIII. Eleven-Point Average for Different Cumulative Relevance Scores (Cystic Fibrosis collection)

Relevance Score	>= 1	>= 2	>= 3	>= 4	>= 5	6
11-point average (no exp)	0.2905	0.3130	0.3313	0.3373	0.3252	0.2834
11-point average (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)

5. 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 subdomains. The word usage in such subdomains 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 sublanguages, 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:

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 VIII 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 Multidatabases

In a multidatabase 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

Table IX. Eleven-Point Average for Different Similarity Matrices (TREC 5 Category A collection; note that the best and worst matrices were manually selected after the queries were run; baseline: 0.1069)

Matrix Selected	Single Matrix	Best	Average	Worst
11-point average	0.1009 (-5.6)	0.1365 (+27.7)	0.1020 (-4.6)	0.0716 (-33.0)
Average rank	N/A	0.0	3.0	6.0

quality by automatically selecting an appropriate matrix where several different candidates exist.

Our experiments for TREC 4 [Gauch and 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 IX summarizes the results.

Examining the results, we rank ordered the matrices for each query based on the 11-point average produced for the expanded query it created. When the best matrix is used to expand each query, performance increases dramatically (27.7%). However, when the worst matrix is used, performance decreases just as dramatically (−33%). Clearly, selecting the correct matrix in a multidatabase 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 (−5.6%).

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 the 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 precalculated, 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, 6 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 and 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 subdatabase 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 X:

—[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.

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 XI.

Table X. Database Selection Average (DBSA) for Methods Based on Examination of the Retrieval Set

Maximum Retrieval Set Rank Considered	Number of Documents Returned	Number Normalized by Number of Documents in DB	Average Rank of Documents Returned	Average Rank Normalized by Number of Documents 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

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 XII.

—[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.

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.

Table XI. Database Selection Average (DBSA) for Method 1.9 Combining Number of Retrieved Documents with Average Rank

$m - p$	$n = 50$	$n = 100$	$n = 150$
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 XII. Database Selection Average (DBSA) for Methods Based on Word Frequencies

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 its sum of individual frequencies over all DBs.	2.19
[2.5] Sum of query word frequencies weighed by IDF.	2.32

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 was 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 XIII.

Table XIII. 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

	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

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. The data are summarized in Table XIV.

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 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 XV.

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

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.

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

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 Words Added	3	4	5
Average DBSA	2.1356	2.1000	2.1133

Table XV. DBSA for Method 3.7 Based on a Combination of Higher and Lower Similarity Thresholds

	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

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 adds 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 multidatabase collections, specifically the seven databases that comprise the TREC 5 Category A collection. Our results show that creating a similarity matrix for each of the subcollections 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

Table XVI. Consolidation of the Best Results for Methods 1.1–3.7

Method Used	DBSA	11-Point Average	Percentage 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

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.

REFERENCES

BUCKLEY, C. 1985. Implementation of the SMART information retrieval system. Tech. Rep. 85-686. Department of Computer Science, Cornell University, Ithaca, NY.

CHURCH, K. W. AND HANKS, P. 1990. Word association norms, mutual information and lexicography. *Comput. Linguist. 16*, 1 (Mar. 1990), 22–29.

CROFT, W. B., COOK, R., AND WILDER, D. 1995. Providing government information on the Internet: Experiences with THOMAS. In *Proceedings of the Digital Libraries Conference* (DL '95). 19–24.

CROUCH, C. J. AND YANG, B. 1992. Experiments in automatic statistical thesaurus construction. In *Proceedings of the 15th Annual International ACM Conference on Research and Development in Information Retrieval* (SIGIR '92, Copenhagen, Denmark, June 21–24), N. Belkin, P. Ingwersen, A. M. Pejtersen, and E. Fox, Eds. ACM Press, New York, NY, 77–88.

DEERWESTER, S., DUMAI, S. T., FURNAS, G. W., LANDAUER, T. K., AND HARSHMAN, R. 1990. Indexing by latent semantic analysis. *J. Am. Soc. Inf. Sci. 41*, 6, 391–407.

FINCH, S. AND CHATER, N. 1992. Bootstrapping syntactic categories using statistical methods. In *Proceedings of the 1st SHOE Workshop* (The Netherlands), W. Daelemans and D. Powers, Eds. 229–235.

GAUCH, S. AND CHONG, M. 1995. Automatic word similarity detection for TREC 4 query expansion. In *Proceedings of the 4th Text Retrieval Conference* (TREC-4, Washington, D.C.,

Nov.), D. K. Harman, Ed. National Institute of Standards and Technology, Gaithersburg, MD, 527–536.

GAUCH, S. AND RACHAKONDA, S. 1997. Experiments in automatic similarity matrix selection for query expansion. Tech. Rep. ITTC-FY97-TR-11100-3. Information and Telecommunication Technology Center, University of Kansas, Lawrence, KS.

GAUCH, S. AND SMITH, J. B. 1991. Search improvement via automatic query reformulation. *ACM Trans. Inf. Syst. 9*, 3 (July 1991), 249–280.

GAUCH, S. AND SMITH, J. B. 1993. An expert system for automatic query reformulation. *J. Am. Soc. Inf. Sci. 44*, 3, 124–136.

GAUCH, S. AND WANG, J. 1996. Automatic word similarity detection for TREC 5 query expansion. In *Proceedings of the 5th Text Retrieval Conference* (TREC-5, Gaithersburg, MD, Nov.), E. M. Voorhees and D. K. Harman, Eds. National Institute of Standards and Technology, Gaithersburg, MD.

GAUCH, S. AND WANG, J. 1997. Tuning a corpus analysis approach for automatic query expansion. Tech. Rep. ITTC-FY97-TR-11100-2. Information and Telecommunication Technology Center, University of Kansas, Lawrence, KS.

GREFENSTETTE, G. 1992. Use of syntactic context to produce term association lists for text retrieval. In *Proceedings of the 15th Annual International ACM Conference on Research and Development in Information Retrieval* (SIGIR '92, Copenhagen, Denmark, June 21–24), N. Belkin, P. Ingwersen, A. M. Pejtersen, and E. Fox, Eds. ACM Press, New York, NY, 89–97.

HARMAN, D. 1992. Relevance feedback revisited. In *Proceedings of the 15th Annual International ACM Conference on Research and Development in Information Retrieval* (SIGIR '92, Copenhagen, Denmark, June 21–24), N. Belkin, P. Ingwersen, A. M. Pejtersen, and E. Fox, Eds. ACM Press, New York, NY, 1–10.

JING, Y. AND CROFT, W. B. 1994. An association thesaurus for information retrieval. In *Proceedings of the Intelligent Multimedia Information Retrieval Systems* (RIAO '94, New York, NY). 146–160.

LIDDY, E. D. AND MYAENG, S. H. 1993. DR-LINK's linguistic-conceptual approach to document detection. In *Proceedings of the 1st Text Retrieval Conference*. 113–129.

MILLER, G. A. AND CHARLES, W. G. 1991. Contextual correlates of semantic similarity. *Lang. Cogn. Process. 6*, 1, 1–28.

MYAENG, S. H. AND LI, M. 1992. Building term clusters by acquiring lexical semantics from a corpus. In *Proceedings of the 1st International Conference on Information and Knowledge Management* (CIKM-92, Baltimore, MD, Nov.), Y. Yesha, Ed. 130–137.

QIU, Y. AND FREI, H. P. 1993. Concept based query expansion. In *Proceedings of the 16th Annual International ACM SIGIR Conference* (Pittsburgh, PA). ACM Press, New York, NY, 160–169.

SCHUTZE, H. AND PEDERSEN, J. 1994. A cooccurrence-based thesaurus and two applications to information retrieval. In *Proceedings of the Intelligent Multimedia Information Retrieval Systems* (RIAO '94, New York, NY). 266–274.

SHAW, W. M. JR., WOOD, J. B., WOOD, R. E., AND TIBBO, H. R. 1991. The cystic fibrosis database: Content and research opportunities. *Libr. Inf. Sci. Res. 12*, 347–366.

SPARCK JONES, K. 1971. *Automatic Keyword Classification for Information Retrieval*. Butterworths, London, UK.

VOORHEES, E. M. 1994. Query expansion using lexical-semantic relations. In *Proceedings of the 17th Annual International ACM Conference on Research and Development in Information Retrieval* (SIGIR '94, Dublin, Ireland, July 3–6), W. B. Croft and C. J. van Rijsbergen, Eds. Springer-Verlag, New York, NY, 61–69.

XU, J. AND CROFT, W. B. 1996. Query expansion using local and global document analysis. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '96, Zurich, Switzerland, Aug. 18–22), H.-P. Frei, D. Harman, P. Schaübie, and R. Wilkinson, Eds. ACM Press, New York, NY, 4–11.

Received: October 1997; revised: June 1998 and August 1998; accepted: September 1998