A Cooccurrence-Based Thesaurus and Two Applications to Information Retrieval*

Hinrich Schütze and Jan O. Pedersen

Xerox Palo Alto Research Center

3333 Coyote Hill Road, Palo Alto, CA 94304, USA

(415) 812-4734 (phone), (415) 812-4374 (fax)

{schuetze,pedersen}@parc.xerox.com

Running title: A Cooccurrence-Based Thesaurus

July 9, 1996

[1] A shorter version of this paper was presented at the RIAO 94 Conference at Rockefeller University in New York, 11-13 October 1994. We thank Marti Hearst, David Hull, John Tukey and the anonymous reviewers for their helpful comments and Mike Berry for making SVDPACK available to us.

Abstract

This paper presents a new method for computing a thesaurus from a text corpus. Each word is represented as a vector in a multi-dimensional space that captures cooccurrence information. Words are defined to be similar if they have similar cooccurrence patterns. Two different methods for using these thesaurus vectors in information retrieval are shown to significantly improve performance over the Tipster reference corpus as compared to a term vector space baseline.

1 Introduction

Information retrieval systems typically define similarity between queries and documents in terms of a weighted sum of matching words (e.g. the vector model, as in (Salton and McGill, 1983)). If a document is relevant but uses words synonymous to words in the query, it cannot be found. This is a particular problem if the query is short. One solution is to lengthen the query through relevance feedback (Salton and Buckley, 1990). Another approach is to expand the query through synonym relations as found in a thesaurus. Here "synonym" is used loosely to mean "closely related word", as opposed to "syntactically and semantically completely interchangeable word". We define a thesaurus as simply a mapping from words to other closely related words.

For a thesaurus to be useful in information retrieval it must be specific enough to offer synonyms for words as used in the corpus of interest. For example, in a corpus of computer science documents the word "interpreter" would have meanings quite different from everyday language. It must also cover all or most of the words found in queries, including the potentially unbounded set of proper nouns. These two considerations suggest that generic thesauri (such as Roget's) that restrict themselves to common usage are unlikely to be helpful. Instead, it seems more promising to rely on thesauri tuned to the corpus of interest. These might be hand-built for a restricted domain or computed from the text of the corpus itself.

This paper presents a new corpus-based method for constructing a thesaurus based on lexical cooccurrence. The computation proceeds in two phases. First, the lexical cooccurrence pattern of each word is represented as a multi-dimensional vector, the *thesaurus vector*. Second, a similarity measure is induced on words by comparing these vectors. A dimensionality reduction is performed on the multi-dimensional vectors in order to make the similarity computations more efficient. Given a particular word its synonyms are then defined to be its nearest neighbors with respect to this similarity measure.

In the following we discuss previous approaches to thesaurus construction, describe our cooccurrence-based thesaurus, and demonstrate two applications of the thesaurus to information retrieval that improve performance as compared to a standard vector-space similarity search baseline over the ARPA Tipster reference corpus (Harman, 1993b). The first application defines the *context vector* of a document to be the weighted sum of the thesaurus vectors of the words occurring in the context. These context vectors then induce a similarity measure on documents and queries which can be directly compared to standard vector-space methods. The second application analyzes a query into subtopics. Documents are then scored and ranked by the degree to which they simultaneously match the subtopics of a query.

2 Related Work

A thesaurus is a data structure that defines semantic relatedness between words. It is typically used in information retrieval to expand search terms with other closely related words. Even if a thesaurus is not explicitly computed, the mapping performed by query expansion implicitly defines a thesaurus. Therefore, we will first discuss previous approaches to thesaurus construction and then comment on query expansion work.

The simplest, and perhaps most conventional, approach to thesaurus construction is to manually build an explicit semantic mapping table. This is labor-intensive, and hence only possible in specialized domains where repeated use may justify the cost. For example, the RUBRIC and TOPIC text retrieval systems (McCune et al., 1985) require a domain expert to prepare a hierarchical structure of "topics" (each topic is a boolean combination of other topics and search terms) germane to a particular subject area. Searchers then employ terms from this hierarchy to form queries that automatically expand to complex boolean expressions.

Another approach is to reuse existing online lexicographic databases, such as WordNet (Voorhees and Hou, 1992) or Longman's subject codes (Liddy and Paik, 1992). However, generic thesauri of this sort will often not be specific enough for the text collection at hand. For example, in (Voorhees and Hou, 1992), "acts" is expanded with the meaning "acts of the apostles" in a corpus of legal documents. In addition, generic thesauri frequently do not record information about proper nouns, yet proper nouns are often excellent retrieval cues.

Corpus-based methods perform a computation on the text of the documents in the corpus to induce a thesaurus. For example, Evans et al. (Evans et al., 1991) construct a hierarchical thesaurus from a computed list of complex noun phrases where subsumption roughly corresponds to the subset relation defined on terms (e.g. "intelligence" subsumes "artificial intelligence"). While this method is superior to approaches that treat phrase terms as unanalyzed atoms, there is no notion of semantic similarity of basic terms. For example, the semantic similarity of “astronaut” and “cosmonaut” is not represented in the hierarchy.

Several researchers have used head-modifier relationships to determine semantic closeness (Grefenstette, 1992; Ruge, 1992; Strzalkowski, 1995). Cooccurrence statistics and head-modifier relationships get at different kinds of information. Two terms that modify the same words (or are modified by the same words) often belong to the same semantic category. Associatively related words can belong to different semantic categories (for example “doctor” and “disease”). However, words with similar heads or modifiers are not always good candidates for expansion. For example, adjectives referring to countries have similar heads (“the Japanese/Chilean capital”, “the Japanese/Chilean government”), but adding “Japanese” to a query that contains “Chilean” will rarely produce good results. Note that there are many words that distinguish “Japanese” and “Chilean” in terms of cooccurrence in a sentence: “Tokyo”, “Andes”, “Samurai”, etc. Grefenstette (Grefenstette, 1992) and Strzalkowski (Strzalkowski, 1995) demonstrate that head-modifier-based term expansion can improve retrieval performance. Our goal in this paper is to show that cooccurrence-based similarity, which is conceptually simpler than similarity with respect to heads or modifiers, is an equally powerful source of information for text retrieval.

Crouch (Crouch, 1990) approaches semantic relatedness by considering the occurrence of terms in documents. Documents are clustered into small groups based on a similarity measure that considers two documents similar if they share a significant number of terms, with medium frequency terms preferentially weighted. Terms are then grouped by their occurrence in these document clusters. Since a complete-link document clustering is performed the procedure is compute intensive; it would not scale to the Tipster reference collection. Further, the central assumption that terms are related if they often occur in the same documents seems problematic for corpora with long documents. It also does not capture the intuitive notion that synonyms do not cooccur, but rather have similar cooccurrence patterns. In contrast the procedure proposed in this paper makes use of lexical cooccurrence, which is more informative both qualitatively and quantitatively (cf. (Schütze, 1992)).

Two terms lexically cooccur if they appear in text within some distance of each other (typically a window of k words). Qualitatively, the fact that two words often occur close to each other is more likely to be significant than the fact that they occur in the same documents, especially if documents are long.

Quantitatively, there are on an order of magnitude more cooccurrence events than occurrence-in-document events in a given document collection. For a word occurring n times in the document collection and for a definition of cooccurrence as occurring in a window of k words, there are nk cooccurrence events, but only n occurrence-in-document events. If the goal is to capture information about specific words, we believe that lexical cooccurrence is the preferred basis for statistical thesaurus construction.

Another difference from the approach in (Crouch, 1990) and (Crouch and Yang, 1992) is that there thesaurus classes are constructed by way of complete-link clustering; words are binned into groups of related words. This is problematic for parts of the semantic space in which there are no clear boundaries. If classes are made too small, some words will be cut off from part of their topical neighborhood. If they are too large, words will be forced into classes with words from different topics. Any particular class size will either separate some words from close neighbors or lump together some words with distant terms.

In contrast, we propose to construct a multi-dimensional continuous space in which each word's thesaurus vector represents its individual position. A continuous space does not force a classification choice, and hence avoids some of the ensuing problems.

Salton and McGill (Salton and McGill, 1983) and Qiu and Frei (Qiu and Frei, 1993) construct thesauri by transposing the standard term-by-document matrix in order to define a similarity measure on terms rather than documents. Terms are represented as high-dimensional vectors with a component for each document in the corpus. The value of each component is a function of the frequency the term has in that document. Qiu and Frei (Qiu and Frei, 1993) show that query expansion using the cosine similarity measure on these vectors improves retrieval performance. However, because the term vectors are high-dimensional, the time complexity for computing the similarity between terms is related to the size of the corpus (in the same way that the cost of document similarity search is related to the size of the corpus). This prevents its use on a large scale. Further, as argued above lexical cooccurrence offers a richer basis for determining word similarity than document occurrence.

Jing and Croft (Jing and Croft, 1994) also exploit lexical cooccurrence information in their PhraseFinder system to build a thesaurus. They use the lexical cooccurrence statistics directly instead of performing a dimensionality reduction as we do. Both approaches have advantages. The dimensionality reduction creates more compact representations. It also generalizes the similarity measure: even terms that do not have any neighbors in common can be close according to the generalized similarity measure if justified by the overall pattern of cooccurrence. However, some precision is lost in the dimensionality reduction. Jing and Croft's approach is thus more conservative and therefore less likely to cause expansion errors. PhraseFinder is also more conservative in that only phrases are used in expansion. Since phrases are rarely ambiguous, this further reduces the risk of expansion errors.

Peat and Willet (Peat and Willett, 1991) argue against the utility of cooccurrence-based expansion when term cooccurrence is used directly for determining similarity, for example using the measure:

$$\frac{f_{xy}}{\sqrt{f_x f_y}}$$

where f_x and f_y are the number of documents in which x and y occur, respectively, and f_{xy} is the number of documents in which both x and y occur. It is argued that according to this measure only terms occurring with similar frequency are judged similar. However, this measure and others discussed in the article are based on first-order cooccurrence, i.e., terms can only be similar if they cooccur with each other directly. Therefore, Peat and Willet's criticism does not apply to the cooccurrence information we use here: second-order cooccurrence, i.e. words sharing the same neighbors. An example of similar terms with different frequency in our experiments are "accident" (frequency 2590) and "mishaps" (frequency 129) (see Table 1).

2.1 Latent Semantic Indexing

Our work on cooccurrence-based thesauri is closely related to Latent Semantic Indexing (LSI) (Deerwester et al., 1990). In LSI, document-by-word matrices are processed by Singular Value Decomposition (SVD) instead of word-by-word matrices. This technique addresses the synonymy problem in document retrieval. A query about "cosmonauts" will not retrieve any documents about "astronauts" in a system based on word-matching. Loosely speaking, the dimensionality reduction computed by a SVD on the document-by-word matrix collapses synonyms onto the same underlying dimension. Therefore, SVD-based representations of the query about cosmonauts will successfully retrieve documents about astronauts.

There are two main differences between our work and LSI, a technical and a conceptual one. The technical difference is that we introduce a more efficient way to compute an LSI-like dimensionality reduction by way of class-based generalization (see below). The setup described in (Deerwester et al., 1990) requires a decomposition of a full term-by-document matrix, which for a large collection such as TIPSTER is an extremely time-consuming operation, since its complexity is quadratic in the number of documents (but see (Dumais, 1993) for sampling strategies to reduce time complexity). Although our method requires two additional passes through the corpus (for collecting the matrices A and B described below), it took only about 30 minutes to compute the Singular Value Decomposition used in this paper. This time is independent of the size of the corpus.

The conceptual difference between LSI and the cooccurrence-based thesaurus introduced here is that we use term representations independently whereas in LSI, term representations are only used to compute document representations. For example, our factorization algorithm groups the terms of a query into topic-coherent groups based on the cooccurrence vectors. This approach exploits more of the information in the term vectors than LSI.

Another difference between LSI and the cooccurrence-based thesaurus introduced here appears to be that LSI works with term-by-document matrices whereas we work with term-by-term matrices. However, this difference turns out to be smaller than one would expect. Suppose that A is a word-by-document matrix with the following SVD decomposition:

$$A = USV^T$$

where U and V are orthonormal matrices and S is a diagonal matrix containing the singular values, which measure the strengths of the dimensions of the decomposition (Golub and van Loan, 1989).

Then $C = AA^T$ is a word-by-word matrix that records in entry c_{ij} how often words i and j cooccur with each other in the same documents, as is immediately apparent from the definition of matrix multiplication.

Some basic matrix algebraic manipulations show that C has the following decomposition:

$$\begin{aligned} C = AA^T &= USV^T(USV^T)^T && (1) \\ &= USV^TVS^TU^T && (2) \\ &= US^2U^T && (3) \end{aligned}$$

So SVD produces similar reduced word vectors (the matrix U) from word-by-word matrices and word-by-document matrices, the only difference being that dimensions with large singular values are more strongly weighted for the word-by-word matrices (squared values rather than simple values from S)[1].

In our comparison of the two SVD methods, we have so far neglected one important aspect of the induction process: The region over which cooccurrence is defined. In LSI, this region is the document. In our approach, the region is a window of k words. There is no obvious generalization of the word-by-document matrix used in LSI to a word-by-context matrix since the number of contexts is very large if a sliding window is used. Local contexts give rise to cooccurrence counts that are of better quality and quantity than document-based counts as we have argued above. We therefore think that this feature is an advantage of our induction algorithm.

In summary, there are both technical and conceptual differences between LSI and our cooccurrence-based thesaurus. The thesaurus induction algorithm introduced here is more efficient and employs a different weighting scheme due to the shortcut we take in computing the dimensionality reduction. Conceptually, we focus on the term representations that are computed by LSI rather than the document representations and we define cooccurrence with respect to local contexts rather than documents.

LSI has also been applied to concept-by-term matrices (Chute et al., 1991) derived from a manually constructed thesaurus. As discussed above, our main interested here is in corpus-derived thesauri.

[1] The above derivation assumes that the matrix A contains raw cooccurrence counts. This is true neither in LSI nor in the matrices used in this paper. Instead, counts are transformed by some form of a term-frequency and inverse document-frequency weighting function ϕ (see below for the particular form we use). If ϕ is applied to the original matrix A, then each entry of term j's vector will be weighted by idf_j, the inverse term frequency of term j.

$$c_{ij} = \sum_d \phi(tf_{id}, idf_i)\phi(tf_{jd}, idf_j)$$

where tf_{id} is the frequency of term i in document d. This results in larger values for rare and highly discriminatory words and smaller values for frequent and less discriminatory words. In the experiments described below we apply the weighting function directly to C so that the entries of C are as follows:

$$c'_{ij} = \phi(\sum_d (tf_{id} tf_{jd}), idf_i)$$

However, we normalize the columns of the matrix which correspond to term vectors. This has an effect similar to the weighting by idf_j in LSI: downweighting of the entries of high-frequency terms and upweighting of the entries of low-frequency terms.

3 Cooccurrence Thesaurus

The goal of the lexical-cooccurrence-based thesaurus is to associate with each term a vector that represents its pattern of local cooccurrences. This vector can then be compared with others to measure the cooccurrence similarity, and hence semantic similarity of terms.

The starting point of the computation is to collect a (symmetric) term-by-term matrix C, such that element c_{ij} records the number of times that words i and j cooccur in a window of size k (k is forty words in our experiments). Topical or semantic similarity between two words can then be defined as the cosine between the corresponding columns of C. The assumption is that words with similar meanings will occur with similar neighbors if enough text material is available.

However, simple resource calculations suggest that this direct approach is not workable. The matrix C has $v^2/2$ distinct entries where v is the size of the vocabulary. Although this matrix is sparse, we can expect v to be very large, and hence the overall storage requirement to be unworkable. For example, the Tipster category B corpus (Harman, 1993b), which is the subject of our experiments, has over 450,000 unique terms.

Even if enough memory were found to represent C directly, the thesaurus vectors associated with each word (columns of C) would be v-dimensional. Although these vectors are somewhat sparse, this implies that word comparisons are an order v operation, which is prohibitively expensive for large scale application.

We address these issues by reducing the dimensionality of the problem to a workable size. The key dimensionality reduction tool is a Singular Value Decomposition (Golub and van Loan, 1989) of a matrix of cooccurrence counts. However, this matrix must be constructed in a series of steps to keep the computations tractable at each stage.

3.1 Practical Implementation

The thesaurus is constructed in three steps as shown in Figure 1. The goal is to apply a Singular Value Decomposition to reduce the dimensionality of the problem in a disciplined fashion and in the process produce more compact representations. However, since SVD is expensive, (time proportional to n^2, where n is the dimensionality of the matrix), the dimensionality of the matrix fed into SVD cannot be too high. In particular, we cannot use the original matrix C. Instead we approximate it in a two stage computation that

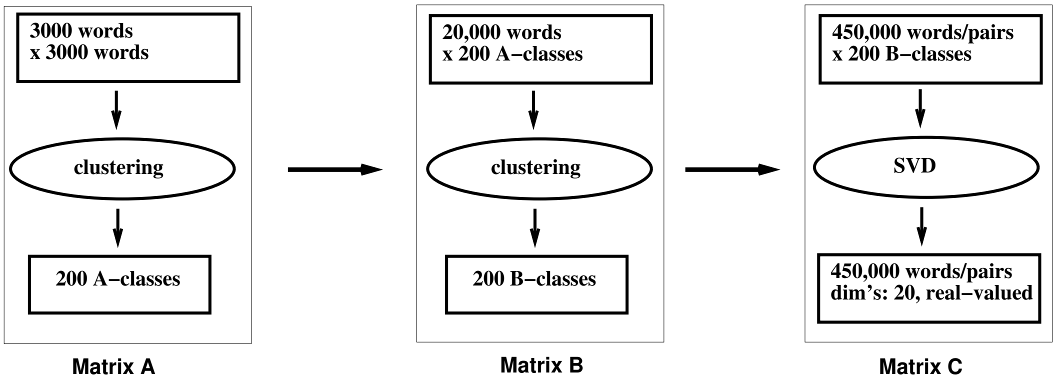

Figure 1: Iterative thesaurus construction.

derives two sets of topical word classes from the corpus (the A-classes and B-classes in the figure) which we use to contain the dimensionality of the problem without sacrificing too much information.

The topical word classes allow us to accumulate information exploiting the topical similarity of the words in a class. We begin by constructing the full cooccurrence matrix for a subset of terms in the corpus (see "Matrix A" in Figure 1). In our experiment we chose 3,000 medium frequency words (frequency ranks 2,000 through 5,000) for this subset. Element $a_{i,j}$ of the matrix records the number of times that words w_i and w_j cooccurred in a window of 40 words in the text collection. We then form the first set of topical word classes by clustering this A-subset into a groups based on the cosine similarity between the columns of matrix A. In our experiment we found 200 A-classes $g_{A1}, g_{A2}, \ldots, g_{A200}$ using group average agglomerative clustering (Gnanadesikan, 1977).

We now consider a larger vocabulary subset and collect a second matrix B which records for each word in this larger B-subset the number of times words in each A-class occur in neighborhoods around that word (see "Matrix B" in Figure 1). Each element $b_{i,j}$ records the number of times that word w_j cooccurs with any of the medium-frequency words from class g_{Ai}. This is similar to the previous cooccurrence matrix construction except that the matrix is no longer symmetric: rows correspond to A-classes, columns to words. In our experiment the B-subset contained the 20,000 most frequent words, excluding stop words. This B-subset is again partitioned into b word classes by clustering the columns of matrix B. The purpose of this second iteration is to ensure that each word in the corpus has sufficiently many neighbors from at least one word class. If we use only A-classes then many words would have no cooccurrence events. In contrast every word cooccurs with several words in the B-subset and hence will have many cooccurrence events with respect to B-classes. However, we could not compute the b-classes directly because a $b \times b$ matrix is computationally intractable. In our experiment the 20,000 columns of matrix B were clustered into 200 B-classes $g_{B1}, g_{B2}, \ldots, g_{B200}$ using the Buckshot fast clustering algorithm (Cutting et al., 1992). Buckshot clusters a random sample first (2000 of the 20,000 words in this case), and then extends this clustering to the complete set by assigning every word to the closest cluster centroid. Experience suggests that this algorithm gives good clustering results (Cutting et al., 1992).

Finally, a third cooccurrence matrix C' is collected for the full corpus vocabulary vs. the B-classes (see "Matrix C" in Figure 1). Element $c_{i,j}$ contains the number of times that term j cooccurs in a window of k words with any word in class g_{Bi}. Matrix C' has b rows and v columns. In our experiment we used all 176,116 words that occurred at least twice in the collection and all 272,914 pairs of adjacent words that occurred at least 5 times, for a total of 449,030 unique terms. At this stage an SVD dimensionality reduction to p ($p < b$) is performed so that each of the v terms can be represented as a compact p dimensional vector and also to improve generalization ((Deerwester et al., 1990), (Berry, 1992)). In our experiment to reduce compute time only a subset of the matrix, corresponding to the 1000th through 6000th most frequent word, was decomposed. This decomposition defined a mapping from the 200-dimensional B-class space to a 20-dimensional reduced space. By applying the mapping to each of the 449,030 200-component B-class vectors, a smaller 20-dimensional vector was computed for each word and pair.

One might ask why we do not employ clustering for the final reduction in dimensionality. The answer lies in the smoothing and improved generality resulting from an SVD reduction. Similarity between b-component vectors can be an errorful measure of semantic similarity since there may be several word classes with similar topics. In our experiment, for example, class g_{B4} contains words like "navy", "radar", and "missile", while some of the members of class g_{B47} are "tanks", "missiles", and "helicopters". If one of two words has many neighbors in g_{B4} and the other has many in g_{B47} then they would not be similar in the 200-dimensional space, but they are similar in the reduced space. This is because the SVD algorithm recognizes and eliminates such redundancies.

As described above this computation requires four passes through the corpus. The first pass computes

accident	repair faulty personnel accidents exhaust equipped MISHAPS injuries sites
advocates	passage PROPONENTS argument address favoring/-s compromise congress urge
litigation	LAWSUITS audit lawsuit file auditors auditor suit sued proceedings
tax	taxes income;tax/-es new;tax taxpayer/-s incentives LEVIES corporate;taxes
treatment	drugs syndrome administer/-ed study administering PROCEDURE undergo aids

Table 1: Five terms and their nearest neighbors in the thesaurus.

word and word pair frequencies. The second pass computes Matrix A and the A-classes, the third pass Matrix B and the B-classes, the fourth pass Matrix C. In addition, Matrix C is SVD decomposed and thesaurus vectors computed. In our experiment, each pass through the Tipster Category B corpus took roughly six hours on a Sun SparcCenter 1000 (includes CPU and IO time). Note that these computations could have been accelerated by using loosely coupled coarse-grained parallelism to effect a linear reduction in compute time. The SVD decomposition required roughly 30 minutes to compute (using SVDPACK, (Berry, 1992)).

3.2 Sample Synonyms

The net effect of this computation is to produce for each unique term a dense p-dimensional vector that characterizes its cooccurrence neighborhoods (where $p = 20$). These vectors then define a thesaurus by associating each word with its nearest neighbors with respect to a similarity measure on vectors. In our experiments we use the cosine measure. The following table displays some of the associations found in our experiment over the Tipster category B corpus. Each row displays a word and its nine nearest neighbors. For example, "repair" is the nearest neighbor of "accident". Word pairs used as terms are displayed as couples separated by semicolon. Words in upper case are hand-selected synonyms as might be found in a manually constructed thesaurus. They are particularly interesting because they are unlikely to cooccur with their mates and hence illustrate that this thesaurus construction effectively uses second-order cooccurrence (sharing neighbors in the corpus) rather than simple first-order cooccurrence (occurring next to each other) to find synonyms.

4 Context Vectors

The thesaurus vectors computed above represent for each word its cooccurrence signature. To use this information directly in search, one needs a similar representation for documents. The simplest approach is to represent each document by the vector which is the sum of the thesaurus vectors for the words in its text. Formally,

$$\vec{d_j} = \sum_i w_{ij} \vec{v_i}$$

where $\vec{d_j}$ is the vector for document j, w_{ij} is the weight for word i in document j, and $\vec{v_i}$ is the thesaurus vector for word i. Queries may be represented as vectors in the same way.

In our experiments, we use augmented tf.idf weighting when summing thesaurus vectors: (Salton and Buckley, 1990)

$$w_{ij} = (0.5 + 0.5 * \frac{tf_{ij}}{max_i(tf_{ij})}) * log(\frac{N}{n_i})$$

where tf_{ij} is the frequency of word i in document j, N is the total number of documents, n_i is the document frequency of word i.

Recall that document vectors that are computed according to this scheme are called "context vectors". Context vectors were first used by Wilks et al. (Wilks et al., 1990) and Gallant (Gallant, 1991). Wilks et al. derive context vectors from dictionary entries. Gallant's original context vectors were based on hand-assigned microfeatures. In more recent work, context vectors are constructed from randomly initialized word vectors (Gallant et al., 1992). In contrast, our approach is corpus-based and completely automatic since it depends only on the underlying thesaurus vectors. General dictionaries and hand-encoded features vary in how appropriate they are for different text collections. A derivation from the corpus of the application increases the chances that the representations are tuned to the relevant topics.

4.1 Application to TIPSTER

Context vectors were computed for the 25 Category B topics of the Tipster collection (Harman, 1993b) and for each of the about 173,000 Wall Street Journal articles in the collection. For each query, documents were ranked according to vector similarity as computed by the cosine measure and precision/recall statistics collected. We compared our results against a baseline standard vector space similarity search (word-based retrieval) with augmented tf.idf term weighting ((Salton and McGill, 1983),(Salton and Buckley, 1990)).

Our direct application of context vectors achieved a small performance improvement in average precision. The baseline word-based average precision for this collection is 0.271 while context vectors achieved 0.274.

To achieve better results we considered schemes that combined the scores from the word-based baseline and context vectors. Formally, we considered document ranks of the form

$$r' = \alpha * r_{words} + (1 - \alpha) * r_{cv}$$

where r_{cv} is the context vector rank and r_{words} is the word-based rank and α is a free parameter between 0 and 1. See Callan (Callan, 1994) for a similar approach to merging results from different retrieval methods (the "data fusion" problem, see Belkin et al. (Belkin et al., 1995) for a general discussion).

Table 2 shows precision at eleven points of recall for word-based retrieval (left column) and for a linear combination for the optimal choice of α, found by exhaustive search ($\alpha = 0.7$, middle column in Table 2). The average precision for word-based retrieval is 0.271 and the average precision for the linear combination of word-based retrieval and context vectors is 0.300.

5 Word Factorization

Another application of thesaurus vectors is to analyze the query into topic-coherent word groups, which we call *word factors*. The goal is to ensure that documents are relevant to the entire query by arranging that their score with respect to each factor be high. In addition, word factors may be manually screened for relevance. Word factors containing nuisance or non-topical terms can be deleted from the query.

We used group average agglomerative clustering to group query terms into factors based on their thesaurus vectors. In this experiment, we arbitrarily decided to cluster each topic into three word factors. For example, the following three factors were found for topic description 51, which is about trade conflicts between the United States and European countries on subsidies to the aircraft industry (see appendix for full text of query). The words and pairs of the topic description were clustered into three groups as shown below. The pairs of words that were used as terms are printed as two consecutive words separated by a semicolon. All

	word-based	word-based + CV's		CV's and factors	
at 0.00	0.658	0.787	+ 19.6	0.832	+ 26.4
at 0.10	0.501	0.551	+ 10.0	0.617	+ 23.2
at 0.20	0.414	0.457	+ 10.4	0.487	+ 17.6
at 0.30	0.338	0.375	+ 10.9	0.388	+ 14.8
at 0.40	0.292	0.314	+ 7.5	0.322	+ 10.3
at 0.50	0.230	0.245	+ 6.5	0.249	+ 8.3
at 0.60	0.174	0.186	+ 6.9	0.207	+ 19.0
at 0.70	0.135	0.143	+ 5.9	0.170	+ 25.9
at 0.80	0.105	0.107	+ 1.9	0.127	+ 21.0
at 0.90	0.077	0.080	+ 3.9	0.085	+ 10.4
at 1.00	0.057	0.057	+ 0.0	0.056	- 1.8
average precision					
	0.271	0.300	+ 10.7	0.322	+ 18.8

Table 2: Precision at 11 points of recall for term matching, for a combination of context vectors and term matching and for factor matching.

directly juxtaposed words occurring at least five times in the corpus were used as terms.

- aid assistance british code complaint

  consortium controversy douglas economics

  european;governments financing french

  german government;assistance governments

  international;economics loan objection petition policy;review producer retaliation review sanctions

  spanish spanish;government tension

- aeronauticas aeronauticas;s.a aerospace

  aerospace;plc aerospatiale airbus airbus;industrie aircraft aircraft;consortium blohm boeing boelkow

  boelkow;blohm british;aerospace construcciones construcciones;aeronauticas

  douglas;corp european;aircraft gmbh mcdonnell mcdonnell;douglas messerschmitt messerschmitt;boelkow

  plc s.a

- airbus;subsidies anti;dumping countervailing

  countervailing;duty dumping dumping;duty

  federal;subsidies gatt general;agreement

  review;group subsidies tariffs trade;dispute

  trade;policy trade;tension

The three factors correspond to the main topics of the query: international politics, the aircraft industry and trade conflicts.

We can use these word factors to ameliorate one outstanding problem of similarity search: it treats search terms as if they were in one large disjunction. By scoring each factor separately and recombining them appropriately, we force documents to score highly on all factors, and thus introduce a conjunctive constraint.

For example, a document may score high for a query as a whole although it deals with only one of the subtopics of the query. A case in point is topic 51. Many high-scoring documents are about the aircraft industry without mentioning trade conflicts or international politics. Instead of evaluating the query as a whole, each subtopic should be evaluated individually and the results combined. If a document is irrelevant to one of the important subtopics of the query, then it often is irrelevant as a whole (for example a document on the aircraft industry without any mention of trade). Therefore, we employed the following algorithm for ranking documents:

- rank documents for each of the subqueries q_1, q_2, q_3, resulting in ranks r_{1i}, r_{2i}, r_{3i} for document i
- assign the maximum of the ranks to document i:

$$r'(i) = max_j(r_{ji})$$

- rank the r' to get the final ranking

This algorithm corresponds to imposing a boolean constraint on the subtopics of a query, namely a rank-based "AND". The highest rank, i.e., the rank of the lowest scoring subtopic, becomes the final rank of the document just as a single false conjunct makes a boolean conjunction false. Two other approaches that combine Boolean and vector similarity retrieval are Extended Boolean Retrieval (Salton et al., 1983) and Bayesian Networks (Turtle and Croft, 1991). In both approaches, Boolean queries are constructed manually whereas we attempt to induce the Boolean structure of a query from the thesaurus vectors of its words. However, we assume that all statements of information need can be expressed as a conjunction of subtopics. In some cases, the structure of a query is disjunctive rather than conjunctive. Further work is necessary to determine the correct logical operator for a query automatically.

Word factorization has a second goal apart from imposing conjunctive constraints: the semi-automatic elimination of irrelevant words. Many words in the Tipster topic descriptions are not relevant for the query in question, but they were not stop words since they could be relevant for other queries. For example, topic description 75 is about failed or successful automation. The topic is identified as belonging to the general area of "Science and Technology". Therefore, "science" is one of the terms of the query, but it is not relevant for the query. One of the word factors of topic 75 is the following:

- failed instance force conversely science

This word factor doesn't contain good search terms and was therefore not used in retrieval. The decision whether a word factor was relevant or not was made manually. The word factors that were judged relevant were then combined according to the algorithm described above.

As in the previous experiment, a linear combination of word-based retrieval and context vectors to evaluate document rank with respect to each factor proved superior to using either method on its own (average precision 0.308 for a ranking based only on words, 0.287 for a ranking based only on context vectors). The results in Table 2 were obtained by combining the ranks of word-based retrieval and context vector searches for each factor separately (the combination factor was $\alpha = 0.86$) and then selecting the largest of the ranks. Table 2 shows precision for 11 recall points for $\alpha = 0.86$. Average precision is 0.322. This is a relative improvement of 18.8% over the word-based result of 0.271 reported above.

This result compares well with the best systems described in (Harman, 1993a) for manual and automatic adhoc retrieval in Category B (for example, 0.3219 in (Kwok et al., 1993)). At this point our system lacks stemming, and the implementation of statistical phrases is restricted (only adjacent words were considered). We hope to improve our results further with stemming and more sophisticated statistical phrases.

6 Conclusion

We have proposed a new algorithm for inducing a thesaurus from a text corpus and have demonstrated two applications of it to information retrieval. First, we have shown that representing documents with context vectors derived from the thesaurus improves recall/precision performance in the Tipster collection. Second, we have introduced a novel application of thesauri to cluster query terms, and have shown that this procedure performs substantially better than a word-match baseline.

Our experiments suggest the quality of the context vector of a document is sensitive to its length. Long documents produce poor context vectors since adding up thesaurus vectors from many topics leads to the averaging out of fine distinctions that we need for effective retrieval. We are therefore planning to experiment with segmented documents, since we hope that our method will improve results even further when applied to document fragments of similar length.

7 Appendix: Tipster Topic 51

<dom> Domain: International Economics

<title> Topic: Airbus Subsidies

<desc> Description:

Document will discuss government assistance to Airbus Industrie, or mention a trade dispute between Airbus and a U.S. aircraft producer over the issue of subsidies.

<smry> Summary:

Document will discuss government assistance to Airbus Industrie, or mention a trade dispute between Airbus and a U.S. aircraft producer over the issue of subsidies.

<narr> Narrative:

A relevant document will cite or discuss assistance to Airbus Industrie by the French, German, British or Spanish government(s), or will discuss a trade dispute between Airbus or the European governments and a U.S. aircraft producer, most likely Boeing Co. or McDonnell Douglas Corp., or the U.S. government, over federal subsidies to Airbus.

<con> Concept(s):

1. Airbus Industrie
2. European aircraft consortium, Messerschmitt-Boelkow-Blohm GmbH, British Aerospace PLC, Aerospatiale, Construcciones Aeronauticas S.A.
3. federal subsidies, government assistance, aid, loan, financing
4. trade dispute, trade controversy, trade tension
5. General Agreement on Tariffs and Trade (GATT) aircraft code
6. Trade Policy Review Group (TPRG)
7. complaint, objection
8. retaliation, anti-dumping duty petition, countervailing duty petition,

`sanctions`

References

Belkin, N., Kantor, P., Fox, E., and Shaw, J. (1995). Combining the evidence of multiple query representations for information retrieval. *Information Processing & Management*, 31(3):431–448.

Berry, M. W. (1992). Large-scale sparse singular value computations. *The International Journal of Supercomputer Applications*, 6(1):13–49.

Callan, J. P. (1994). Passage-level evidence in document retrieval. In *Proceedings of SIGIR '94*, pages 302–310.

Chute, C. G., Yang, Y., and Evans, D. A. (1991). Latent semantic indexing of medical diagnoses using umls semantic structures. In Clayton, P. D., editor, *Fifteenth Annual Symposium on Computer Applications in Medical Care (SCAMC)*, pages 185–189, Washington DC. IEEE Computer Society.

Crouch, C. J. (1990). An approach to the automatic construction of global thesauri. *Information Processing & Management*, 26(5):629–640.

Crouch, C. J. and Yang, B. (1992). Experiments in automatic statistical thesaurus construction. In *Proceedings of SIGIR*, pages 77–88.

Cutting, D. R., Pedersen, J. O., Karger, D., and Tukey, J. W. (1992). Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of SIGIR '92*, pages 318–329.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Dumais, S. T. (1993). Latent semantic indexing (lsi) and trec-2. In *The Second Text REtrieval Conference (TREC-2)*, pages 105–115.

Evans, D. A., Ginther-Webster, K., Hart, M., Lefferts, R. G., and Monarch, I. A. (1991). Automatic indexing using selective nlp and first-order thesauri. In *Proceedings of the RIAO*, volume 2, pages 624–643.

Gallant, S. I. (1991). A practical approach for representing context and for performing word sense disambiguation using neural networks. *Neural Computation*, 3(3):293–309.

Gallant, S. I., Caid, W. R., Carleton, J., Hecht-Nielsen, R., Qing, K. P., and Sudbeck, D. (1992). HNC's matchplus system. In *Proceedings of TREC*.

Gnanadesikan, R. (1977). *Methods for Statistical Data Analysis of Multivariate Observations*. John Wiley & Sons, New York.

Golub, G. H. and van Loan, C. F. (1989). *Matrix Computations*. The Johns Hopkins University Press, Baltimore and London.

Grefenstette, G. (1992). Use of syntactic context to produce term association lists for text retrieval. In *Proceedings of SIGIR '92*, pages 89–97.

Harman, D., editor (1993a). *The First Text REtrieval Conference (TREC-1)*. U.S. Department of Commerce, Washington DC. NIST Special Publication 500-207.

Harman, D. (1993b). Overview of the first trec conference. In *Proceedings of SIGIR '93*.

Jing, Y. and Croft, W. B. (1994). An association thesaurus for information retrieval. In *Proceedings of RIAO*, pages 146–160, Rockefeller University, New York.

Kwok, K. L., Papadopoulos, L., and Kwan, K. Y. Y. (1993). Retrieval experiments with a large collection using pircs. pages 153–172. In (Harman, 1993a).

Liddy, E. D. and Paik, W. (1992). Statistically-guided word sense disambiguation. In Goldman, R., Norvig, P., Charniak, E., and Gale, B., editors, *Working Notes of the AAAI Fall Symposium on Probabilistic Approaches to Natural Language*. AAAI Press.

McCune, B. P., Tong, R., and Dean, J. (1985). Rubric, a system for rule-based information retrieval. *IEEE Transactions on Software Engineering*, 9:939–944.

Peat, H. J. and Willett, P. (1991). The limitations of term co-occurrence data for query expansion in document retrieval systems. *Journal of the American Society for Information Science*, 42(5):378–383.

Qiu, Y. and Frei, H. (1993). Concept based query expansion. In *Proceedings of SIGIR '93*.

Ruge, G. (1992). Experiments on linguistically-based term associations. *Information Processing & Management*, 28(3):317–332.

Salton, G. and Buckley, C. (1990). Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288–297.

Salton, G., Fox, E. A., and Wu, H. (1983). Extended boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036.

Salton, G. and McGill, M. J. (1983). *Introduction to modern information retrieval*. McGraw-Hill, New York.

Schütze, H. (1992). Dimensions of meaning. In *Proceedings of Supercomputing '92*, pages 787–796, Minneapolis MN.

Strzalkowski, T. (1995). Natural language information retrieval. *Information Processing & Management*, 31(3):397–417.

Turtle, H. and Croft, W. B. (1991). Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems*, 9(3):187–222.

Voorhees, E. M. and Hou, Y.-W. (1992). Vector expansion in a large collection. In *Proceedings of TREC*.

Wilks, Y. A., Fass, D. C., ming Guo, C., McDonald, J. E., Plate, T., and Slator, B. M. (1990). Providing machine tractable dictionary tools. *Journal of Computers and Translation*, 2.