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
{schuetze,pedersen}@parc.xerox.com

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 ARPA Tipster evaluation corpus as compared to a tf.idf 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 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 one must 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. 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 the new 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 text retrieval evaluation corpus (Harman 1993b). The first application defines the *context vector* of a document to be the weighted sum of the thesaurus vectors of its contained words. 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 proper.

The simplest, and perhaps most conventional, approach to thesaurus construction is to manually build an explicit semantic mapping table. This is clearly 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, they 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. (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.

Grefenstette (1992) and Ruge (1992) use head-modifier relationships to determine semantic closeness. This solution is costly since parsing technology is required to determine head-modifier relations in sentences. It is also unclear to what extent words with similar heads or modifiers are 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 coocurrence in a sentence: "Tokyo", "Andes", "Samurai", etc. Grefenstette (1992) demonstrates 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 information retrieval.

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 very 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. 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.

Crouch (1990) constructs thesaurus classes; words are binned into groups of related words. This is problematic since the boundaries between classes

will be inevitably somewhat artificial. 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.

Qiu and Frei (1993) present an elegant and effective thesaurus construction by inverting the standard vector-space document similarity function to define a similarity measure on terms. 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. They 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, as will be proposed in the discussion on context vectors below. Further, as argued above lexical cooccurrence offers a richer basis for determining word similarity than document occurrence.

Peat and Willett (1991) argue against the utility of cooccurrence-based expansion in principle. They observe that because synonyms often do not occur together a cooccurrence-based approach may have difficulty identifying synonymy relations. However, although synonyms frequently don't cooccur, they tend to share neighbors that occur with both. For example, "litigation" and "lawsuit" share neighbors such as "court", "judge", and "proceedings". Our thesaurus represents lexical cooccurrence patterns and hence defines semantic closeness in terms of common neighbors. This implies we do not require synonyms to cooccur, but rather require them to have similar cooccurrence patterns.

A second problem pointed out by Peat and Willett is that many researchers use measures for defining closeness that will group words according to frequency: it is impossible for a frequent word to have an infrequent neighbor according to these measures. We avoid this difficulty by reducing the dimensionality of the thesaurus space using a singular value decomposition (cf. Deerwester et al. 1990). The reason for the closeness of terms with equal frequency is, roughly, that they have about the same number of zero entries in their term vectors. For a given term, SVD assigns values to all dimensions of the space, so that frequent and infrequent terms can be close in the reduced space if they occur with similar terms. For example, "accident" (frequency 2590 in our corpus) and "mishaps" (frequency 129) will come out as similar in the experiment described below despite the frequency difference between them.

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. Qiu and Frei (1993) use a similar scheme, although the matrix in their case is documents vs. terms.

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 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 agglomerate information over similar words. 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 usual 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).[1]

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 word and word pair frequencies. The sec-

[1] A random sample of 2,000 of the 20,000 words was clustered first, and this clustering was then extended to all 20,000 words by assigning every word to the cluster whose centroid it was closest to.

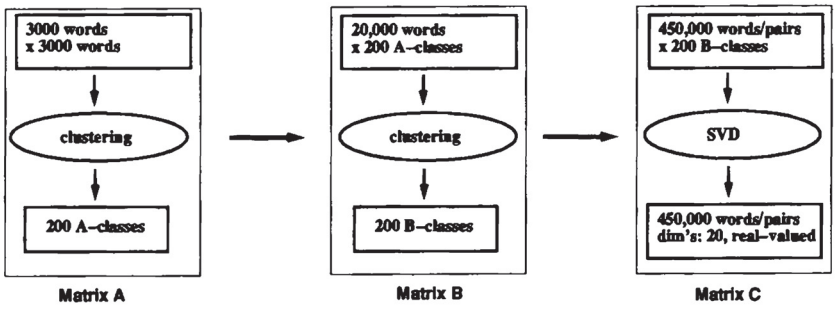

Figure 1: Iterative thesaurus construction.

ond 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 (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. 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 the cosine. 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 coocurrence (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 proposed by Gallant (1991) as an encoding based on hand-assigned microfeatures. In contrast, our approach is completely automatic since it depends only on the underlying thesaurus vectors. Hand-encoded features may also 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.

The work on Latent Semantic Indexing (Deerwester et al. 1990) also bears close resemblance to

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.