Indexing by Latent Semantic Analysis

Scott Deerwester
Graduate Library School
University of Chicago
Chicago, IL 60637

Susan T. Dumais
George W. Furnas
Thomas K. Landauer
Bell Communications Research
435 South St.
Morristown, NJ 07960

Richard Harshman
University of Western Ontario
London, Ontario Canada

ABSTRACT

A new method for automatic indexing and retrieval is described. The approach is to take advantage of implicit higher-order structure in the association of terms with documents ("semantic structure") in order to improve the detection of relevant documents on the basis of terms found in queries. The particular technique used is singular-value decomposition, in which a large term by document matrix is decomposed into a set of ca 100 orthogonal factors from which the original matrix can be approximated by linear combination. Documents are represented by ca 100 item vectors of factor weights. Queries are represented as pseudo-document vectors formed from weighted combinations of terms, and documents with supra-threshold cosine values are returned. Initial tests find this completely automatic method for retrieval to be promising.

1. Introduction

We describe here a new approach to automatic indexing and retrieval. It is designed to overcome a fundamental problem that plagues existing retrieval techniques that try to match words of queries with words of documents. The problem is that users want to retrieve on the basis of conceptual content, and individual words provide unreliable evidence about the conceptual topic or meaning of a document. There are usually many ways to express a given concept, so the literal terms in a user's query may not match those of a relevant document. In addition, most words have multiple meanings, so terms in a user's query will literally match terms in documents that are not of interest to the user.

The proposed approach tries to overcome the deficiencies of term-matching retrieval by treating the unreliability of observed term-document association data as a statistical problem. We assume there is some underlying latent semantic structure in the data that is partially obscured by the randomness of word choice with respect to retrieval. We use statistical techniques to estimate this latent structure, and get rid of the obscuring "noise". A description of terms and documents based on the latent semantic structure is used for indexing and retrieval.[1]

The particular "latent semantic indexing" (LSI) analysis that we have tried uses singular-value decomposition. We take a large matrix of term-document association data and construct a "semantic" space wherein terms and documents that are closely associated are placed near one another. Singular-value decomposition allows the arrangement of the space to reflect the major associative patterns in the data, and ignore the smaller, less important influences. As a result, terms that did not actually appear in a document may still end up close to the document, if that is consistent with the major patterns of association in the data. Position in the space then serves as the new kind of semantic indexing, and retrieval proceeds by using the terms in a query to identify a point in the space, and documents in its neighborhood are returned to the user.

2. Deficiencies of current automatic indexing and retrieval methods

A fundamental deficiency of current information retrieval methods is that the words searchers use often are not the same as those by which the information they seek has been indexed. There are actually two sides to the issue; we will call them broadly *synonymy* and *polysemy*. We use *synonymy* in a very general sense to describe the fact that there are many ways to refer to the same object. Users in different contexts, or with different needs, knowledge, or linguistic habits will describe the same information using different terms. Indeed, we have found that the degree of variability in descriptive term usage is much greater than is commonly suspected. For example, two people choose the same main key word for a single well-known object less than 20% of the time [1] . Comparably poor agreement has been reported in studies of inter-indexer consistency [2] and in the generation of search terms by either expert intermediaries [3] or less experienced searchers [4] [5] . The prevalence of synonyms tends to decrease the "recall" performance of retrieval systems. By *polysemy* we refer to the general fact that most words have more than one distinct meaning

1. By "semantic structure" we mean here only the correlation structure in the way in which individual words appear in documents; "semantic" implies only the fact that terms in a document may be taken as referents to the document itself or to its topic.

(homography). In different contexts or when used by different people the same term (e.g. "chip") takes on varying referential significance. Thus the use of a term in a search query does not necessarily mean that a document containing or labeled by the same term is of interest. Polysemy is one factor underlying poor "precision".

The failure of current automatic indexing to overcome these problems can be largely traced to three factors. The first factor is that the way index terms are identified is incomplete. The terms used to describe or index a document typically contain only a fraction of the terms that users as a group will try to look it up under. This is partly because the documents themselves do not contain all the terms users will apply, and sometimes because term selection procedures intentionally omit many of the terms in a document.

Attempts to deal with the synonymy problem have relied on intellectual or automatic term expansion, or the construction of a thesaurus. These are presumably advantageous for conscientious and knowledgeable searchers who can use such tools to suggest additional search terms. The drawback for fully automatic methods is that some added terms may have different meaning from that intended (the polysemy effect) leading to rapid degradation of precision [6] .

It is worth noting in passing that experiments with small interactive data bases have shown monotonic improvements in recall rate without overall loss of precision as more indexing terms, either taken from the documents or from large samples of actual users' words are added [7] [8] . Whether this "unlimited aliasing" method, which we have described elsewhere, will be effective in very large data bases remains to be determined. Not only is there a potential issue of ambiguity and lack of precision, but the problem of identifying index terms that are not in the text of documents grows cumbersome. This was one of the motives for the approach to be described here.

The second factor is the lack of an adequate automatic method for dealing with polysemy. One common approach is the use of controlled vocabularies and human intermediaries to act as translators. Not only is this solution extremely expensive, but it is not necessarily effective. Another approach is to allow Boolean intersection or coordination with other terms to disambiguate meaning. Success is severely hampered by users' inability to think of appropriate limiting terms if they do exist, and by the fact that such terms may not occur in the documents or may not have been included in the indexing.

The third factor is somewhat more technical, having to do with the way in which current automatic indexing and and retrieval systems actually work. In such systems each word type is treated as independent of any other (see, for example, van Rijsbergen [9]). Thus matching (or not) both of two terms that almost always occur together is counted as heavily as matching two that are rarely found in the same document. Thus the scoring of success, in either straight Boolean or coordination level searches, fails to take redundancy into account, and as a result may distort results to an unknown degree. This problem exacerbates a user's difficulty in using compound-term queries effectively to expand or limit a search.

3. Rationale of the Latent Semantic Indexing (LSI) method

3.1 Illustration of retrieval problems

We illustrate some of the problems with term-based information retrieval systems by means of a fictional matrix of terms by documents (Table 1).

Sample Term by Document matrix

	access	*document*	*retrieval*	*information*	*theory*	*database*	*indexing*	*computer*	REL	MATCH
Doc 1	x	x	x			x	x		R	
Doc 2				x*	x			x*		M
Doc 3			x	x*				x*	R	M

Query: "IDF in *computer*-based *information* look-up"

Table 1

Below the table we give a fictional query that might have been passed against this database. An "R" in the column labeled REL (relevant) indicates that the user would have judged the document relevant to the query (here documents 1 and 3 are relevant). Terms occurring in both the query and a document (*computer* and *information*) are indicated by an asterisk in the appropriate cell; an "M" in the MATCH column indicates that the document matches the query and would have been returned to the user. Documents 1 and 2 illustrate common classes of problems with which the proposed method deals. Document 1 is a relevant document, which, however, contains none of the words in the query. It would, therefore, not be returned by a straightforward term overlap retrieval scheme. Document 2 is a non-relevant document which does contain terms in the query, and therefore would be returned, despite the fact that the query context makes it clear enough to a human observer that a different sense of at least one of the words is intended. Note that in this example none of the meaning conditioning terms in the query is found in the index. Thus intersecting them with the query terms would not have been a plausible strategy for omitting document 2.

Start by considering the synonymy problem. One way of looking at the problem is that document 1 should have contained the term "look-up" from the user's perspective, or conversely that the query should have contained the term "access" or "retrieval" from the system's. To flesh out the analogy, we might consider any document (or title or abstract) to consist of a small selection from the complete discourse that might have been written on its topic. Thus the text from which we extract index terms is a fallible observation from which to infer what terms actually apply to its topic. The same can be said about the query; it is only one sample description of the intended documents, and in principle could have contained many different terms from the ones it does.

Our job then, in building a retrieval system, is to find some way to predict what terms "really" are implied by a query or apply to a document (i.e. the "latent semantics") on the basis of the fallible sample actually found there. If there were no correlation between the occurrence of one term and another, then there would be no way for us to use the data in a term by document matrix to estimate the "true" association of terms and documents where data are in error. On the other hand, if there is a great deal of structure, i.e. the occurrence of some patterns of words gives us a strong clue as to the likely occurrence of others, then data from one part (or all) of the table can be used to correct other portions. For example suppose that in our total collection the words "access" and "retrieval" each occurred in 100 documents, and that 95 of these documents containing "access" also contained "retrieval". We might reasonably guess that the absence of "retrieval" from a document containing "access" might be erroneous, and consequently wish to retrieve the document in response to a query containing only "retrieval". The kind of structure on which such inferences can be based is not

limited to simple pairwise correlation.

In document 2 we would like our analysis to tell us that the term "information" is in fact something of an imposter. Given the other terms in the query and in that document we would predict no occurrence of a term with the meaning here intended for "information", i.e. knowledge desired by a searcher. A correlational structure analysis may allow us to down-weight polysemous terms by taking advantage of such observations.

Our overall research program has been to find effective models for overcoming these problems. We would like a representation in which a set of terms, which by itself is incomplete and unreliable evidence of the relevance of a given document, is replaced by some other set of entities which are more reliable indicants. We take advantage of implicit higher-order (or latent) structure in the association of terms and documents to reveal such relationships.

3.2 The choice of method for uncovering latent semantic structure

The goal is to find and fit a useful model of the relationships between terms and documents. We want to use the matrix of observed occurrences of terms applied to documents to estimate parameters of that underlying model. With the resulting model we can then estimate what the observed occurrences really should have been. In this way, for example, we might predict that a given term should be associated with a document, even though, because of variability in word use, no such association was observed.

The first question is what sort of model to choose. A notion of semantic similarity, between documents and between terms, seems central to modeling the patterns of term usage across documents. This led us to restrict consideration to proximity models, i.e., models that try to put similar items near each other in some space or structure. Such models include: hierarchical, partition and overlapping clusterings; ultrametric and additive trees; and factor-analytic and multidimensional distance models (see Carroll & Arabie [10] for a survey).

Aiding information retrieval by discovering latent proximity structure has at least two lines of precedence in the literature. Hierarchical classification analyses are frequently used for term and document clustering [11] [12] [13] . Latent class analysis [14] and factor analysis [15] [16] [17] have also been explored before for automatic document indexing and retrieval.

In document clustering, for example, a notion of distance is defined such that two documents are considered close to the extent that they contain the same terms. The matrix of document-to-document distances is then subjected to a clustering analysis to find a hierarchical classification for the documents. Retrieval is based on exploring neighborhoods of this structure. Similar efforts have analyzed word usage in a corpus and built clusters of related terms, in effect making a statistically-based thesaurus. We believe an important weakness of the clustering approach is that hierarchies are far too limited to capture the rich semantics of most document sets. Hierarchical clusterings permit no cross classifications, for example, and in general have very few free parameters (essentially only n parameters for n objects). Empirically, clustering improves the computational efficiency of search; whether or not it improves retrieval success is unclear [13] [18] [19] .

Previously tried factor analytic approaches have taken a square symmetric matrix of similarities between pairs of documents (based on statistical term overlap or human judgments), and used linear algebra to construct a low dimensional spatial model wherein similar documents are placed near one

another. The factor analytic model has the potential of much greater richness than the clustering model (a k dimensional model for n points has nk parameters). However previous attempts along these lines, too, had shortcomings. First, factor analysis is computationally expensive, and since most previous attempts were made 15-20 years ago, they were limited by processing constraints [16]. Second, most past attempts considered restricted versions of the factor analytic model, either by using very low dimensionality, or by converting the factor analysis results to a simple binary clustering [16]. Third, some attempts have relied on excessively tedious data gathering techniques, requiring the collection of thousands of similarity judgments from humans [17].

Previously reported clustering and factor analytic approaches have also struggled with a certain representational awkwardness. Typically the original data explicitly relate two types of entities, terms and documents, and most conceptions of the retrieval problem mention both types (e.g., given terms that describe a searchers' interests, relevant documents are returned). However, representations chosen so far handle only one at a time (e.g., either term clustering or document clustering). Any attempts to put the ignored entity back in the representation have been arbitrary and after the fact. An exception to this is a proposal by Koll [20] in which both terms and documents are represented in the same space of concepts (see also Raghavan & Wong [21]). While Koll's approach is quite close in spirit to the one we propose, his concept space was of very low dimensionality (only seven underlying dimensions), and the dimensions were hand-chosen and not truly orthogonal as are the underlying axes in factor analytic approaches. [2]

Our approach differs from previous attempts in a number of ways that will become clearer as the model is described in more detail. To foreshadow some of these differences, we: (1) examine problems of reasonable size (1000-2000 document abstracts; and 5000-7000 index terms); (2) use a rich, high-dimensional representation (about 100 dimensions) to capture term-document relations (and this appears necessary for success); (3) use a mathematical technique which explicitly represents both terms and documents in the same space; and (4) retrieve documents from query terms directly, without rotation or interpretation of the underlying axes and without using intermediate document clusters.

We considered alternative models using the following three criteria:

1. *Adjustable representational richness.* To represent the underlying semantic structure, we need a model with sufficient power. We believe hierarchical clusterings to be too restrictive, since they allow no multiple or crossed classifications and have essentially only as many parameters as objects. Since the right kind of alternative is unknown, we looked for models whose power could be varied, as some compensation for choosing a perhaps inappropriate structure. The most obvious class is dimensional models, like multidimensional scaling and factor analysis, where representational power can be controlled by choosing the number, k, of dimensions (i.e., k parameters per object).

---

2. Koll begins with a set of 7 non-overlapping but almost spanning documents which form the axes of the space. Terms are located on the axis of the document in which they occur; the remainder of the documents are processed sequentially and placed at the average of their terms. This approach has been evaluated on only a small dataset where it was moderately successful.

2. *Explicit representation of both terms and documents.* The desire to represent both terms and documents simultaneously is more than esthetic. In our proximity-based latent structure paradigm, retrieval proceeds by appropriately placing a new object corresponding to the query in the semantic structure and finding those documents that are close by. One simple way to achieve appropriate placement is if terms, as well as documents, have positions in the structure. Then a query can be placed at the centroid of its term points. Thus for both elegance and retrieval mechanisms, we needed what are called two-mode proximity methods (Carroll and Arabie [10]), that start with a rectangular matrix and construct explicit representations of both row and column objects. One such method is multidimensional unfolding [22] [23] [24] , in which both terms and documents would appear as points in a single space with similarity related monotonically to Euclidean distance. Another is two-mode factor analysis [25] [26] [27] [28] , in which terms and documents would again be represented as points in a space, but similarity is given by the inner product between points. A final candidate is unfolding in trees [29] , in which both terms and documents would appear as leaves on a tree, and path length distance through the tree would give the similarity (one version of this is equivalent to simultaneous hierarchical clustering of both terms and objects). The explicit representation of both terms and documents also leads to a straightforward way in which to add or "fold-in" new terms or documents that were not in the original matrix. New terms can be placed at the centroid of the documents in which they appear; similarly, new documents can be placed at the centroid of their constituent terms.[3]
3. *Computational tractability for large datasets.* Many of the existing models require computation that goes up with N^4 or N^5 (where N is the number of terms plus documents). Since we hoped to work with document sets that were at least in the thousands, models with efficient fitting techniques were needed.

The only model which satisfied all three criteria was two-mode factor analysis. The tree unfolding model was considered too representationally restrictive, and along with non-metric multidimensional unfolding, too computationally expensive. Two-mode factor analysis is a generalization of the familiar factor analytic model based on singular value decomposition (SVD). (See Forsythe, Malcolm & Moler [30] , Chapter 9, for an introduction to SVD and its applications.) SVD represents both terms and documents as vectors in a space of choosable dimensionality, and the dot product or cosine between points in the space gives their similarity. In addition, a program was available [31] that fit the model in time of order $N^2 \times k^3$.

---

3. There are several important and interesting issues raised by considering the addition of new terms and documents into the space. First, the addition of new objects introduces some temporal dependencies in the representation. That is, where a new term or document gets placed depends on what other terms and documents are already in the space. Second, in general, simply folding-in new terms or documents will result in a somewhat different space than would have been obtained had these objects been included in the original analysis. Since the initial analysis is time consuming, it is clearly advantageous to be able to add new objects by folding-in. How much of this can be done without rescaling is an open research issue, and is likely to depend on the variability of the database over time, the representativeness of the original of documents and terms, etc.

4. SVD or two-mode factor analysis

4.1 Overview

The latent semantic structure analysis starts with a matrix of terms by documents. This matrix is then analyzed by singular value decomposition (SVD) to derive our particular latent semantic structure model. Singular value decomposition is closely related to a number of mathematical and statistical techniques in a wide variety of other fields, including eigenvector decomposition, spectral analysis, and factor analysis. We will use the terminology of factor analysis, since that approach has some precedence in the information retrieval literature.

The traditional, one-mode factor analysis begins with a matrix of associations between all pairs of one type of object, e.g., documents [16]. This might be a matrix of human judgments of document to document similarity, or a measure of term overlap computed for each pair of documents from an original term by document matrix. This square symmetric matrix is decomposed by a process called "eigen-analysis", into the product of two matrices of a very special form (containing "eigenvectors" and "eigenvalues"). These special matrices show a breakdown of the original data into linearly independent components or "factors". In general many of these components are very small, and may be ignored, leading to an approximate model that contains many fewer factors. Each of the original documents' similarity behavior is now approximated by its values on this smaller number of factors. The result can be represented geometrically by a spatial configuration in which the dot product or cosine between vectors representing two documents corresponds to their estimated similarity.

In two-mode factor analysis one begins not with a square symmetric matrix relating pairs of only one type of entity, but with an arbitrary rectangular matrix with different entities on the rows and columns, e.g., a matrix of terms and documents. This rectangular matrix is again decomposed into three other matrices of a very special form, this time by a process called "singular-value-decomposition" (SVD). (The resulting matrices contain "singular vectors" and "singular values".) As in the one-mode case these special matrices show a breakdown of the original relationships into linearly independent components or factors. Again, many of these components are very small, and may be ignored, leading to an approximate model that contains many fewer dimensions. In this reduced model all the term-term, document-document and term-document similarity is now approximated by values on this smaller number of dimensions. The result can still be represented geometrically by a spatial configuration in which the dot product or cosine between vectors representing two objects corresponds to their estimated similarity.

Thus, for information retrieval purposes, SVD can be viewed as a technique for deriving a set of uncorrelated indexing variables or factors; each term and document is represented by its vector of factor values. Note that by virtue of the dimension reduction, it is possible for documents with somewhat different profiles of term usage to be mapped into the same vector of factor values. This is just the property we need to accomplish the improvement of unreliable data proposed earlier. Indeed, the SVD representation, by replacing individual terms with derived orthogonal factor values, can help to solve all three of the fundamental problems we have described.

In various problems, we have approximated the original term-document matrix using 50-100 orthogonal factors or derived dimensions. Roughly speaking, these factors may be thought of as artificial concepts; they represent extracted common meaning components of many different words and documents. Each term or document is then characterized by a vector of weights indicating its strength of association with each of these underlying concepts. That is, the "meaning" of a

particular term, query, or document can be expressed by k factor values, or equivalently, by the location of a vector in the k-space defined by the factors. The meaning representation is economical, in the sense that N original index terms have been replaced by the $k < N$ best surrogates by which they can be approximated. We make no attempt to interpret the underlying factors, nor to "rotate" them to some meaningful orientation. Our aim is not to be able to describe the factors verbally but merely to be able to represent terms, documents and queries in a way that escapes the unreliability, ambiguity and redundancy of individual terms as descriptors.

It is possible to reconstruct the original term by document matrix from its factor weights with reasonable but not perfect accuracy. It is important for the method that the derived k-dimensional factor space *not* reconstruct the original term space perfectly, because we believe the original term space to be unreliable. Rather we want a derived structure that expresses what is reliable and important in the underlying use of terms as document referents.

Unlike many typical uses of factor analysis, we are not necessarily interested in reducing the representation to a very low dimensionality, say two or three factors, because we are not interested in being able to visualize the space or understand it. But we do wish both to achieve sufficient power and to minimize the degree to which the space is distorted. We believe that the representation of conceptual space for any large document collection will require more than a handful of underlying independent "concepts", and thus that the number of orthogonal factors that will be needed is likely to be fairly large. Moreover, we believe that the model of a Euclidean space is at best a useful approximation. In reality, conceptual relations among terms and documents certainly involves more complex structures, including, for example, local hierarchies and non-linear interactions between meanings. More complex relations can often be made to approximately fit a dimensional representation by increasing the number of dimensions. In effect, different parts of the space will be used for different parts of the language or object domain. Thus we have reason to avoid both very low and extremely high numbers of dimensions. In between we are guided only by what appears to work best. What we mean by "works best" is not (as is customary in some other fields) what reproduces the greatest amount of variance in the original matrix, but what will give the best retrieval effectiveness.

How do we process a query in this representation? Recall that each term and document is represented as a vector in k-dimensional factor space. A query, just as a document, initially appears as a set of words. We can represent a query (or "pseudo-document") as the weighted sum of its component term vectors. (Note that the location of each document can be similarly described; it is a weighted sum of its constituent term vectors.) To return a set of potential candidate documents, the pseudo-document formed from a query is compared against all documents, and those with the highest cosines, that is the nearest vectors, are returned. Generally either a threshold is set for closeness of documents and all those above it returned, or the n closest are returned. (We are concerned with the issue of whether the cosine measure is the best indication of similarity to predict human relevance judgments, but we have not yet systematically explored any alternatives, cf. Jones & Furnas [32] .)

A concrete example may make the procedure and its putative advantages clearer. Table 2 gives a sample dataset. In this case, the document set consisted of the titles of 9 Bellcore technical memoranda. Words occurring in more than one title were selected for indexing; they are italicized. Note that there are two classes of titles: five about human-computer interaction (labeled c1-c5) and four about graph theory (labeled m1-m4). The entries in the term by document matrix are simply

the frequencies with which each term actually occurred in each document. Such a matrix could be used directly for keyword-based retrievals or, as here, for the initial input of the SVD analysis.

Technical Memo Example

Titles:
c1: *Human* machine *interface* for Lab ABC *computer* applications
c2: A *survey* of *user* opinion of *computer system response time*
c3: The *EPS user interface* management *system*
c4: *System* and *human system* engineering testing of *EPS*
c5: Relation of *user*-perceived *response time* to error measurement

m1: The generation of random, binary, unordered *trees*
m2: The intersection *graph* of paths in *trees*
m3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering
m4: *Graph minors*: A *survey*

Terms	**Documents**								
	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

A sample dataset consisting of the titles of 9 technical memoranda. Terms occurring in more than one title are italicized. There are two classes of documents - five about human-computer interaction (c1-c5) and four about graphs (m1-m4). This dataset can be described by means of a term by document matrix where each cell entry indicates the frequency with which a term occurs in a document.

Table 2

For this example we carefully chose documents and terms so that SVD would produce a satisfactory solution using just two dimensions. Figure 1 shows the two-dimensional geometric representation for terms and documents that resulted from the SVD analysis. Details of the mathematics underlying the analysis will be presented in the next section. The numerical results of the SVD for this example are shown in the Appendix and can be used to verify the placement of terms and documents in Figure 1.

Insert Figure here - 2d example of 9 tms

Figure 1

Terms are shown as filled circles and labeled accordingly; document titles are represented by open squares, with the numbers of the terms contained in them indicated parenthetically. Thus, each term and document can be described by its position in this two-dimensional factor space.

One test we set ourselves is to find documents relevant to the query: "human computer interaction". Simple term matching techniques would return documents c1, c2 and c4 since they share one or more terms with the query. However, two other documents which are also relevant (c3 and c5) are missed by this method since they have no terms in common with the query. The latent semantic structure method uses the derived factor representation to process the query; the first 2-dimensions are shown in Figure 1. First, the query is represented as a "pseudo-document" in the factor space. Two of the query terms, "human" and "computer", are in the factor space, so the query is placed at their centroid and scaled for comparison to documents (the point labeled q in Figure 1 represents the query). Then, we simply look for documents which are near the query, q. In this case, documents c1-c5 (but not m1-m4) are "nearby" (within a cosine of .9, as indicated by the dashed lines). Notice that even documents c3 and c5 which share no index terms at all with the query are near it in this representation. The relations among the documents expressed in the factor space depend on complex and indirect associations between terms and documents, ones that come from an analysis of the structure of the whole set of relations in the term by document matrix. This is the strength of using higher order structure in the term by document matrix to represent the underlying meaning of a single term, document, or query. It yields a more robust and economical representation than do straight term overlap or surface-level clustering methods.

4.2 Technical details

4.2.1 The Singular Value Decomposition (SVD) Model.

This section details the mathematics underlying the particular model of latent structure, singular value decomposition, that we currently use. The casual reader may wish to skip this section and proceed to Section 5.

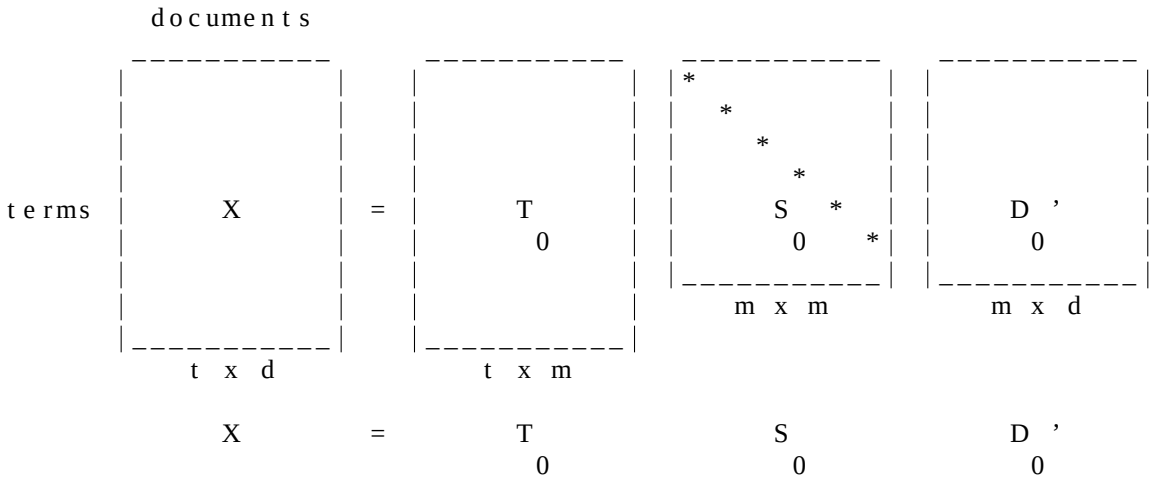
Any rectangular matrix, for example a t **x** d matrix of terms and documents, X, can be decomposed into the product of three other matrices:

$$X = T_0 S_0 D_0',$$

such that T_0 and D_0 have orthonormal columns and S_0 is diagonal. This is called the *singular value decomposition of* X. T_0 and D_0 are the matrices of *left* and *right singular vectors* and S_0 is the diagonal matrix of *singular values*.[4] Singular value decomposition (SVD) is unique up to certain

row, column and sign permutations[5] and by convention the diagonal elements of S_0 are constructed to be all positive and ordered in decreasing magnitude.

Figure 2 presents a schematic of the singular value decomposition for a t **x** d matrix of terms by documents.



Singular value decomposition of the term x document matrix, X. Where :

T_0 has orthogonal, unit-length columns ($T_0'T_0=I$)
D_0 has orthogonal, unit-length columns ($D_0'D_0=I$)
S_0 is the diagonal matrix of singular values
t is the number of rows of X
d is the number of columns of X
m is the rank of X ($\leq \min(t, d)$)

Figure 2

In general, for $X=T_0S_0D_0'$ the matrices T_0, D_0, and S_0 must all be of full rank. The beauty of an SVD, however, is that it allows a simple strategy for optimal approximate fit using smaller matrices. If the singular values in S_0 are ordered by size, the first k largest may be kept and the remaining smaller ones set to zero. The product of the resulting matrices is a matrix *Xhihat* which is only approximately equal to X, and is of rank k. It can be shown that the new matrix *Xhihat* is the matrix of rank k which is closest in the least squares sense to X. Since zeros were introduced

---

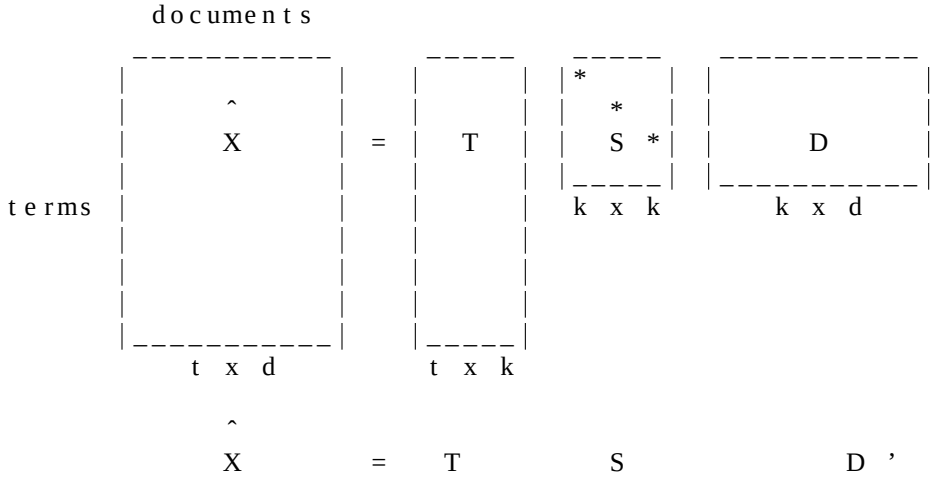
4. SVD is closely related to the standard eigenvalue-eigenvector or spectral decomposition of a square symmetric matrix, Y, into VLV', where V is orthonormal and L is diagonal. The relation between SVD and eigen analysis is more than one of analogy. In fact, T_0 is the matrix of eigen vectors of the square symmetric matrix $Y=XX'$, D_0 is the matrix of eigen vectors of $Y=X'X$, and in both cases, S_0^2 would be the matrix, L, of eigenvalues.

5. Allowable permutations are those that leave S_0 diagonal and maintain the correspondences with T_0 and D_0. That is, column i and j of S_0 may be interchanged iff row i and j of S_0 are interchanged, and columns i and j of T_0 and D_0 are interchanged.

into S_0, the representation can be simplified by deleting the zero rows and columns of S_0 to obtain a new diagonal matrix S, and then deleting the corresponding columns of T_0 and D_0 to obtain T and S respectively. The result is a reduced model:

$$X \approx Xhihat = TSD'$$

which is the rank-k model with the best possible least-squares-fit to X. It is this reduced model, presented in Figure 3, that we use to approximate our data.



Reduced singular value decomposition of the term x document matrix, X. Notation is as in the previous figure except that k $(\leq m)$ is the chosen number of dimensions (factors) in the reduced model.

Figure 3

The amount of dimension reduction, i.e., the choice of k, is critical to our work. Ideally, we want a value of k that is large enough to fit all the real structure in the data, but small enough so that we do not also fit the sampling error or unimportant details. The proper way to make such choices is an open issue in the factor analytic literature. In practice, we currently use an operational criterion - a value of k which yields good retrieval performance.

4.2.2 Geometric interpretation of the SVD model

For purposes of intuition and discussion it is useful to interpret the SVD geometrically. The rows of the reduced matrices of singular vectors are taken as coordinates of points representing the documents and terms in a k dimensional space. With appropriate rescaling of the axes, by quantities related to the associated diagonal values of S, dot products between points in the space can be used to compare the corresponding objects. The next section details these comparisons.

4.2.3 Computing fundamental comparison quantities from the SVD model

There are basically three sorts of comparisons of interest: those comparing two terms ("How similar are terms i and j?"), those comparing two documents ("How similar are documents i and j?"), and those comparing a term and a document ("How associated are term i and document j?"). In standard information retrieval approaches, these amount respectively, to comparing two rows, comparing two columns, or examining individual cells of the original matrix of term by document data, X. Here we make similar comparisons, but use the matrix $Xhihat$, since it is presumed to

represent the important and reliable patterns underlying the data in X. Since $Xhihat = TSD'$, the relevant quantities can be computed just using the smaller matrices, T, D, and S.

4.2.3.1 Comparing two terms. The dot product between two row vectors of *Xhihat* reflects the extent to which two terms have a similar pattern of occurrence across the set of documents. The matrix $XhihatXhihat'$ is the square symmetric matrix containing all these term-to-term dot products. Since S is diagonal and D is orthonormal It is easy to verify that:

$$XhihatXhihat' = TS^2T'$$

Note that this means that the i,j cell of $XhihatXhihat'$ can be obtained by taking the dot product between the i and j rows of the matrix TS. That is, if one considers the rows of TS as coordinates for terms, dot products between these points give the comparison between terms. Note that the relation between taking T as coordinates and taking TS as coordinates is simple since S is diagonal; the positions of the points are the same except that each of the axes has been stretched or shrunk in proportion to the corresponding diagonal element of S.

4.2.3.2 Comparing two documents. The analysis for comparing two documents is similar, except that in this case it is the dot product between two *column* vectors of the matrix *Xhihat* which tells the extent to which two documents have a similar profile of terms. Thus the matrix $Xhihat'Xhihat$ contains the document-to-document dot products. The definitions of the matrices T, S and D again guarantee:

$$Xhihat'Xhihat = DS^2D'$$

Here the i,j cell of $Xhihat'Xhihat$ is obtained by taking the dot product between the i and j rows of the matrix DS. So one can consider rows of a DS matrix as coordinates for documents, and take dot products in this space. (Again note that the DS space is just a stretched version of the D space.)

4.2.3.3 Comparing a term and a document. This comparison is different. Instead of trying to estimate the dot product between rows or between columns of *Xhihat*, the fundamental comparison between a term and a document is the value of an individual cell of *Xhihat*. *Xhihat* is defined in terms of matrices T, S and D. Repeating it here:

$$Xhihat = TSD'$$

The i,j cell of *Xhihat* is therefore obtained by taking the dot product between the i-th row of the matrix $TS^{½}$ and the j-th row of the matrix $DS^{½}$. Note that while the within comparisons (i.e., term-term or document-document) involve using rows of TS and DS for coordinates, the between comparison requires $TS^{½}$ and $DS^{½}$ for coordinates. That is, it is not possible to make a single configuration of points in a space that will allow both between and within comparisons. They will be similar however, differing only by a stretching or shrinking of the axes by a factor of $S^{½}$.

4.2.4 Finding representations for pseudo-documents

The previous results show how it is possible to compute comparisons between the various objects associated with the rows or columns of *Xhihat*. It is very important in information retrieval applications to compute appropriate comparison quantities for objects that did not appear in the original analysis. For example, we want to be able to take a completely novel query, find some point for it in the space, and then look at its cosine with respect to terms or documents in the space. Another example would be trying, after-the-fact, to find representations for documents that did not

appear in the original analysis. The new objects in both these examples are very much like the documents of the matrices, X and $Xhihat$, in that they present themselves as vectors of terms. It is for this reason that we call them $pseudo-documents$. In order to compare a query or pseudo-document, q, to other documents, we need to be able to start with its term vector X_q and derive a representation D_q that we can use just like a row of D in the comparison formulas of the preceding section. One criterion for such a derivation is that putting in a real document X_i should give D_i (at least when the model is perfect, i.e., $X=Xhihat$). With this constraint, a little algebra shows that:

$$D_q = X_q' T S^{-1}$$

Note that with appropriate rescaling of the axes, this amounts to placing the pseudo-document at the centroid of its corresponding term points. This D_q then is just like a row of D and, appropriately scaled by $S^{½}$ or S, can be used like a usual document's factor vector for making between or within comparisons, respectively.

4.2.5 Preprocessing and normalization

The equations given here do not take into account any preprocessing or reweighting of the rows or columns of X. Such preprocessing might be used to prevent documents of different overall length from having differential effect on the model, or be used to impose certain preconceptions of which terms are more important. The effects of certain of these transformations can be taken into account in a straightforward way, but we will not go into the algebra here.

5. Tests of the SVD Latent Semantic Indexing (LSI) method

We have so far tried the LSI method on two standard document collections where queries and relevance judgments were available (MED and CISI). PARAFAC (Harshman & Lundy [31]), a program for the iterative numerical solution of multi-mode factor-analysis problems, was used for the studies reported below. (Other programs for more standard SVD are also available - e.g., [33] [34].)

"Documents" consist of the full text of the title and abstract. Each document is indexed automatically; all terms occurring in more than one document and not on a stop list of 439 common words used by SMART are included in the analyses.[6] We did *not* stem words or map variants of words to the same root form. The analysis begins with a term by document matrix in which each cell indicates the frequency with which each term occurs in each document. This matrix was analyzed by singular value decomposition to derive our latent structure model which was then used for indexing and retrieval. Queries were placed in the resulting space at the centroid of their constituent terms (again, all terms not on the stop list and occurring in more than one document were used). Cosines between the query vector and document vectors are then straightforward to

---

6. We have argued above that the more terms the better, but so far, computational constraints have limited us to around 7000 terms. Terms that occur in only one document, or equally frequently in all documents have little or no influence on the SVD solution. Rejecting such terms has usually been sufficient to satisfy our computational constraints. (In addition, we wanted to be as consistent with SMART as possible in indexing, thus the omission of SMART's common words.) Given greater resources, we see no reason to omit any terms from the latent structure analysis. Even given current limited computational resources, the terms omitted in indexing can be used for *retrieval* purposes by folding them back into the concept space, as we described briefly in section 3.2.

compute (see Section 4.2 for details), and documents are ordered by their distance to the query. In many senses, the current LSI method is impoverished and thus provides a conservative test of the utility of latent semantic structure in indexing and information retrieval. We have so far avoided adding refinements such as stemming, phrases, term-weighting and Boolean combinations (all of which generally result in performance improvements) in order to better evaluate the utility of the basic representation technique.

We compare the results of our latent structure indexing (LSI) method against a straightforward term matching method, a version of SMART, and against data reported by Voorhees [19] for the same standard datasets. The term overlap comparisons provide a baseline against which to assess the benefits of indexing by means of latent semantic structure rather than raw term matching. For the term matching method, we use the same term-document matrix that was the starting point for the LSI method. A query is represented as a column, and cosines between the query column and each document column are calculated. The SMART and Voorhees systems are more representative of state of the art information retrieval systems, but differences in indexing, term weighting, and query processing preclude precise comparisons of our LSI method and these systems. Nonetheless, such comparisons are of interest. For the SMART evaluations, documents were indexed using a stop list of common words, full stemming, and raw term frequencies as options. Queries were similarly processed and a vector sequential search was used for matching queries and documents. This particular invocation of SMART is the same as our term matching method except for the initial choice of index terms. The Voorhees data were obtained directly from her paper in which she used a vector retrieval system with extended Boolean queries (see Voorhees [19] for details). Her documents were indexed by removing words on a stop list, mapping word variants into the same term, and weighting terms. Weighted extended Boolean queries were used for retrieval.

Performance is evaluated by measuring precision at several different levels of recall. This is done separately for each available query and then averaged over queries. For the LSI, term matching, and SMART runs, full precision-recall curves can be calculated. For the Voorhees data, only two precision-recall pairs are available; these are the values obtained when 10 or 20 documents were returned - see her Figures 4b) and 6b). We present the values from her sequential search (SEQ) condition, since the best performance is generally observed in this condition and not in one of the retrieval conditions using document clusters.

5.1 MED

The first standard set we tried, MED, was the commonly studied collection of medical abstracts. It consists of 1033 documents, and 30 queries. Our automatic indexing on all terms occurring in more than one document and not on SMART's stoplist of common words resulted in 5823 indexing terms. Some additional characteristics of the dataset are given below:

	term & LSI	Voorhees	SMART
number of unique terms	5823	6927	6927
mean number of terms per document	50.1	51.6	51.6
mean number of terms per query	9.8	39.7[8]	10.1
mean number of relevant documents per query	23.2	23.2	23.2

The number of unique terms, terms per document and terms per query vary somewhat because different term-processing algorithms were used in the different systems. A 100-factor SVD of the 5823 term by 1033 document matrix was obtained, and retrieval effectiveness evaluated against the 30 queries available with the dataset. Figure 4 shows precision as a function of recall for a LSI 100-factor solution ("LSI-100"), term matching ("TERM"), SMART ("SMART"), and the Voorhees data ("VO"), all on the same set of documents and queries.

Insert Figure here - MED precision-recall curve

Figure 4

For all but the two lowest levels of recall (.10), precision of the LSI method lies well above that obtained with straightforward term matching, SMART, and the vector method reported by Voorhees. The average difference in precision between the LSI and the term matching method is .06 (.51 vs. .45), which represents a 13% improvement over raw term matching. (The odds against a difference this large or larger by chance is 29 to 1, $t\,(29) = 2.23$.) Thus, LSI captures some structure in the data which is obscured when raw term overlap is used. The LSI method also compares favorably with SMART ($t\,(29) = 1.96$; the odds against a difference this large or larger by chance is 16 to 1) and the Voorhees system. It is somewhat surprising that the term matching and SMART methods do not differ for this data set. There are several differences in indexing between LSI and SMART (word stemming is used in SMART but not LSI, and SMART includes word stems occurring in any document whereas LSI, for computational reasons, includes only terms occurring in more than one document) that should lead to better performance for SMART. The difference in performance between LSI and the other methods is especially impressive at higher recall levels where precision is ordinarily quite low, thus representing large proportional improvements. The comparatively poor performance of the LSI method at the lowest levels of recall can be traced to at least two factors. First, precision is quite good in all systems at low recall, leaving little room for improvement. Second, latent semantic indexing is designed primarily to handle synonymy problems (thus improving recall); it is less successful in dealing with polysemy (precision). Synonymy is not much of a problem at low recall since any word matches will retrieve some of the relevant documents. Thus the largest benefits of the LSI method should be observed at high recall, and, indeed, this is the case.

Up to this point, we have reported LSI results from a 100-factor representation (i.e. in a 100-dimensional space). This raises the important issue of choosing the dimensionality. Ideally we want enough dimensions to capture all the real structure in the term-document matrix, but not too many, or we may start modeling noise or irrelevant detail in the data. How to choose the appropriate number of dimensions is an open research issue. In our tests, we have been guided by the operational criterion of "what works best". That is, we examine performance for several different

---

8. The value 39.7 is reported in Table 1 of the Voorhees paper. We suspect this is in error, and that the correct value may be 9.7 which would be in line with the other measures of mean number of terms per query.

numbers of factors, and select the dimensionality which maximizes retrieval performance.[9] The results for the MED dataset are shown in Figure 5 which presents average precision as a function of number of factors.

Insert Figure here - MED avg precision as fcn of NFACT

Figure 5

As can be seen, mean precision more than doubles (from .25 to .52) as the number of factors increases from 10 to 100, with a maximum at 100. We therefore use the 100-factor space for the results we report. In this particular dataset, performance might improve a bit if solutions with more than 100 factors were explored, but, in general, it is not the case that more factors necessarily means better performance. (In other small applications, we have seen much clearer maxima; performance increases up to some point, and then decreases when too many factors are used. One interpretation of this decrease is that the extra parameters are modeling the sampling noise or peculiarities of the sample rather than important underlying relationships in the pattern of term usage over documents.) It is also important to note that previous attempts to use factor analytic techniques for information retrieval have used small numbers of factors (Koll [20] , 7 dimensions; and Ossario [17] , 13 dimensions; Borko & Bernick [16] , 21 dimensions). We show a more than 50% improvement in performance beyond this range, and therefore suspect that some of the limited utility of previous factor analytic approaches may be the result of an impoverished representation.

Unfortunately this MED dataset was specially constructed in a way that may have resulted in unrealistically good results. From what we can determine, the test collection was made up by taking the union of the returns of a set of thorough keyword searches for documents relevant to the 30 queries in the set. It thus may be an unrepresentatively well-segmented collection. The sets of documents for particular queries are probably isolated to an abnormal extent in the multidimensional manifold of concepts. In such a circumstance our method does an excellent job of defining the isolated subdomains and separating them for retrieval. This is probably not the way most natural document collections are structured. It is worth noting, however, that other automatic techniques applied to the same dataset are not able to capitalize as well on this same abnormal structural property. Thus the fact that LSI greatly outperforms the rest is still quite significant. (Note also that the use of keyword searches to define the document test set probably biases results in favor of methods based on surface term matching, such as SMART, since no documents that do not contain any of the keywords are included.)

---

9. This is actually quite easy to do since the SVD solutions are nested. To explore performance in a 10-dimensional solution, for example, cosines are calculated using only the first 10 coordinates of the 100-factor solution.

5.2 CISI

Our second test case is the CISI set of 1460 information science abstracts. This set has been consistently difficult for automatic retrieval methods. It consists of 1460 documents and 35 queries. Our automatic indexing, which excluded words on SMART's stop list of common words and words occurring in only one document, resulted in 5135 index terms. Some additional characteristics of the dataset are given below:

	term & LSI	Voorhees	SMART
number of unique terms	5135	4941	5019
mean number of terms per document	45.4	43.9	45.2
mean number of terms per query	7.7	7.2	7.8
mean number of relevant documents per query	49.8	49.8	49.8

A 100-factor SVD solution was obtained from the 5135 term by 1460 document matrix, and evaluated using the first 35 queries available with the dataset. LSI results for a 100-factor solution ("LSI-100") along with those for term matching ("TERM"), SMART ("SMART") and Voorhees ("VO") are shown in Figure 6.

Insert Figure here - CISI precision-recall curve

Figure 6

All the methods do quite poorly on this dataset, with precision never rising above .30, even for the lowest levels of recall. Average precision for is .11 for both LSI and term matching (t = 1). For this data set, the latent structure captured by the SVD analysis is no more useful than raw term overlap in capturing the distinctions between relevant and irrelevant documents for the available queries. The Voorhees data cover only a very limited range of low recall levels, but for these values precision is similar to that for LSI and term matching. SMART, on the other hand, results in reliably better performance than LSI, although the absolute levels of precision (.14) is still very low. (The odds against differences this large or larger by chance is over 1000 to 1; t (34) = 3.66.) We believe that the superiority of SMART over LSI can be traced to differences in term selection that tend to improve performance. As noted previously, SMART used stemmed words but LSI did not, and SMART included all terms whereas the LSI included only those appearing in more than one document. Since few terms which appear in only one document (and were thus excluded by LSI) are used in the queries, the omission of these words is unlikely to be a major determinant of performance. Thus, stemming appears to be the likely source of performance differences.

We have recently completed a new LSI analysis using SMART's index terms. This enabled us to explore how much of the difference between SMART and the original LSI was due differences in term selection and further to see if additional latent structure could be extracted. For this analysis, we began with a 5019 term (SMART's terms) by 1460 document matrix and obtained a 100-factor SVD solution. The 35 test queries were re-evaluated using this new LSI solution (which we refer to

as LSI-SMART). The resulting performance was indistinguishable from SMART's; average precision for both methods was .14 ($t < 1$). This suggests that much of the initial difference between LSI and SMART was due to term selection differences. Unfortunately, LSI was unable to improve upon term matching - either in the initial LSI vs. term matching comparison, or in the LSI-SMART vs. SMART comparison. Stemming, however, seems to capture some structure that LSI was unable to capture as evidenced by the superior performance of SMART relative to term matching. In theory, latent semantic analyses can extract at least some of the commonalities is usage of stemmed forms. In practice, we may often have insufficient data to do so.

A problem in evaluating the CISI dataset is the very low level of precision. Our intuition is that this database contains a very homogeneous distribution of documents that is hard to differentiate on the basis of abstracts. Moreover, many of the test queries, which were given in natural language, seem very vague and poorly stated. Thus the relevance judgments may not be sufficiently reliable to allow any retrieval system to perform well, or to provide an adequate comparison between methods. No direct evidence has been reported on the reliability (repeatability) of these relevance judgments, so this is mostly conjecture, although we do find many cases in which the relevance judgments appear to be in obvious error. In addition, it seems to us that poorly stated queries would invite excessive reliance on term overlap in judging relevance, especially if the judges were familiar with term matching as a possible retrieval strategy.

5.3 Summary of results from LSI analyses

These results are modestly encouraging. They show the latent semantic indexing method to be superior to simple term matching in one standard case and equal in another. Further, for these two databases, performance with LSI is superior to that obtained with the system described by Voorhees; it performed better than SMART in one case and equal in the other (when term selection differences were eliminated). In order to assess the value of the basic representational method, we have so far avoided the addition of refinements that one would consider in a realistic application, such as discriminative term weighting, stemming, phrase finding or a method of handling negation or disjunction in the queries. So far we have tested the method only with queries formulated to be used against other retrieval methods; the method almost certainly could do better with queries in some more appropriate format. We have projects in progress to add standard enhancements and to incorporate them in a fully automatic indexing and retrieval system. In addition, we are working on methods to incorporate the very low frequency, but often highly informative, words that were filtered out in the trial analysis procedures. It seems likely that with such improvements LSI will offer a more effective retrieval method than has previously been available.

6. Conclusions and discussion

Although factor analytic approaches have been previously suggested and tried in the literature, they have all had what we believe to be serious shortcomings which the present attempt overcomes. We have examined problems of reasonable size (1000-2000 document abstracts; and 5000-7000 index terms) using a rich, high-dimensional representation, which appears necessary for success. The explicit representation of both terms and documents in the same space makes retrieving documents relevant to user queries a straightforward matter. Previous work by Borko and his colleagues [15] [16] is similar in name to our approach, but used the factor space only for document clustering, not document retrieval, and computational simplifications reduced its representational

power. In Borko and Bernick [16] , for example, factor analysis was performed on a term-term correlation matrix (calculated from word usage over 260 abstracts), and 21 orthogonal factors were selected on the basis of their interpretability. Documents were classified into these 21 categories on the basis of normalized factor loadings for each term in the abstract, and performance was comparable to that of another automatic system. It should be noted, however, that the information used for classification is much less than that which is available in the 21-dimensional factor space, since only the factor loading of "significant" terms on each of the factors was used (e.g. one value for 5, 4, and 7 terms defining the three sample factors presented in their Appendix B). In addition, Borko's work addressed the problem of document classification, and not document retrieval. There is, for example, no discussion of how one might use the full factor space (and not just the document clusters derived from it) for document retrieval.

Koll's [20] work on concept-based information retrieval is very similar in spirit to our latent semantic indexing. Both terms and documents are represented in a single concept space on the basis of statistical term co-occurrences. Beginning with axes defined by a set of 7 non-overlapping (in terms) and almost-spanning documents, terms were placed on the appropriate axis. New documents were placed at the mean of constituent terms, and new terms were placed at the location of the document in which they occurred. The system was evaluated with only a very small database of documents and queries, but under some circumstances performance was comparable to that of SIRE for Boolean and natural language queries. Our experience with the MED dataset suggests that better performance might have been obtained with a higher dimensional representation. In addition, the latent semantics approach is not order-dependent (as is Koll's procedure), and it is a mathematically rigorous way of uncovering truly orthogonal basis axes or factors for indexing.

The representation of documents by LSI is economical; each document and term need be represented only by something on the order of 50 to 150 values. We have not explored the degree of accuracy needed in these numbers, but we guess that a small integer will probably suffice. The storage requirements for a large document collection can be reduced because much of the redundancy in the characterization of documents by terms is removed in the representation. Offsetting the storage advantage is the fact that the only way documents can be retrieved is by an exhaustive comparison of a query vector against all stored document vectors. Since search algorithms in high dimensional space are not very efficient on serial computers, this may detract from the desirability of the method for very large collections. An additional drawback involves updating. The initial SVD analysis is time consuming, so we would like a more efficient method of adding new terms and documents. We suggest that new documents be located at the centroid of their terms (appropriately scaled); and new terms be placed at the centroid of the documents in which they appear (appropriately scaled). How much of this updating can be done without having to perform a new decomposition is unknown.

While the LSI method deals nicely with the synonymy problem, it offers only a partial solution to the polysemy problem. It helps with multiple meanings because the meaning of a word can be conditioned not only by other words in the document but by other appropriate words in the query not used by the author of a particular relevant document. The failure comes in the fact that every term is represented as just one point in the space. That is, a word with more than one entirely different meaning (e.g. "bank"), is represented as a weighted average of the different meanings. If none of the real meanings is like the average meaning, this may create a serious distortion. (In classical term overlap methods, the meaning of the term is the union of all of it's meanings, which probably leads to less outright distortion, but to more imprecision.) What is needed is some way to detect the fact

that a particular term has several distinct meanings and to subcategorize it and place it in several points in the space. We have not yet found a satisfactory way to do that (but see Amsler [36] ; Choueka & Lusignan [37] ; Lesk [38]).

The latent semantic indexing methods that we have discussed, and in particular the singular-value decomposition technique that we have tested, are capable of improving the way in which we deal with the problem of multiple terms referring to the same object. They replace individual terms as the descriptors of documents by independent "artificial concepts" that can be specified by any one of several terms (or documents) or combinations thereof. In this way relevant documents that do not contain the terms of the query, or whose contained terms are qualified by other terms in the query or document but not both, can be properly characterized and identified. The method yields a retrieval scheme in which documents are ordered continuously by similarity to the query, so that a threshold can be set depending on the desires and resources of the user and service.

At this point in its development, the method should be regarded as a potential component of a retrieval system, rather than as a complete retrieval system as such. As a component it would serve much the same function as is served by raw term vector ranking and other comparison methods. It's putative advantages would be the noise reduction, as described above, and data compaction through the elimination of redundancy. In applying the method, some of the same implementation issues will arise as in raw vector methods - in particular questions of term weighting, stemming, phrasal entries, similarity measure, and counterparts for Boolean operators. Unfortunately, the value of such retrieval enhancing procedures will have to be reevaluated for use with LSI because its representation changes the nature of the problems with which these procedures were intended to deal. For example, stemming is done to capture likely synonyms. Since LSI already deals with this problem to some extent, the additional value of stemming is an open question. Likewise, LSI averages the "meaning" of polysemous words, where raw term matching maintains one-to-many mappings; as a result, phrases and other disambiguation techniques may be more important.

FIGURE CAPTIONS

Figure 1. A 2-dimensional plot of 12 Terms and 9 Documents from the sample TM set. Terms are represented by filled circles. Documents are shown as open squares, and component terms are indicated parenthetically. The query ("human computer interaction") is represented as a pseudo-document at point q. Axes are scaled for Document-Document or Term-Term comparisons (see section 4.2.3 for details about scaling). The dotted cone represents the region whose points are within a cosine of .9 from the query q. All documents about human-computer (c1-c5) are "near" the query (i.e. within this cone), but none of the graph theory documents (m1-m4) are nearby. In this reduced space, even documents c3 and c5 which share no terms with the query are near it.

Figure 2. Schematic of the Singular Value Decomposition (SVD) of a rectangular term by document matrix. The original term by document matrix is decomposed into three matrices each with linearly independent components.

Figure 3. Schematic of the *reduced* Singular Value Decomposition (SVD) of a term by document matrix. The original term by document matrix is *approximated* using the k largest singular values and their corresponding singular vectors.

Figure 4. Precision-recall curves for TERM matching, a 100-factor LSI, SMART, and Voorhees systems on the MED dataset. The data for the TERM matching, LSI, and SMART methods are obtained by measuring precision at each of 9 levels of recall (approximately .10 increments) for each query separately and then averaging over queries. The two Voorhees data points are taken from Table 4b in her paper.

Figure 5. A plot of average precision (averaged over 9 levels of recall) as a function of number of factors for the MED dataset. Precision more than doubles (from about .20 to .50) as the number of factors is increased from 10 to 100.

Figure 6. Precision-recall curves for TERM matching, a 100-factor LSI, SMART, and Voorhees systems on the CISI dataset. The data for TERM matching, LSI, and SMART are obtained by measuring precision at each of 9 levels of recall (approximately .10 increments) for each query separately and then averaging over queries. The two Voorhees data points are taken from Table 6b in her paper.

ACKNOWLEDGEMENTS

We would like to thank several colleagues who have been involved in various aspects of this project. Laura Beck has contributed extensively to programming, data collection, and analyses. Lynn Streeter, Karen Lochbaum, Bob Allen and Steve Hanson have all used our programs and provided useful feedback. We thank Mike Lesk for advice and comments on previous drafts of the paper, and Ram Gnanadesikan, John Kettenring and John Tukey for consultation on statistical questions.

Appendix - SVD Numerical Example

In section 4.2, we outlined the details of the Singular Value Decomposition (SVD) Model. This Appendix presents a numerical example using the sample term by document matrix described in section 4.1 and shown in Table 2 and Figure 1.

The example 12-term by 9-document matrix from Table 2 is presented below.

X =

1	0	0	1	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
0	1	1	0	1	0	0	0	0
0	1	1	2	0	0	0	0	0
0	1	0	0	1	0	0	0	0
0	1	0	0	1	0	0	0	0
0	0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0	1
0	0	0	0	0	1	1	1	0
0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	1	1

Recall that any rectangular matrix, for example a $t \times d$ matrix of terms and documents, X, can be decomposed into the product of three other matrices:

$$X = T_0 S_0 D_0',$$

such that T_0 and D_0 have orthonormal columns and S_0 is diagonal. This is called the *singular value decomposition (SVD) of X*.

Computing the SVD of the X matrix presented above results in the following three matrices for T_0, S_0, D_0 (rounded to two decimal places).
T_0 (9-dimensional left-singular vectors for 12 terms)
S_0 (diagonal matrix of 9 singular values)
D_0 (9-dimensional right-singular vectors for 9 documents)

T_0 =

0.22	-0.11	0.29	-0.41	-0.11	-0.34	0.52	-0.06	-0.41
0.20	-0.07	0.14	-0.55	0.28	0.50	-0.07	-0.01	-0.11
0.24	0.04	-0.16	-0.59	-0.11	-0.25	-0.30	0.06	0.49
0.40	0.06	-0.34	0.10	0.33	0.38	0.00	0.00	0.01
0.64	-0.17	0.36	0.33	-0.16	-0.21	-0.17	0.03	0.27
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05
0.30	-0.14	0.33	0.19	0.11	0.27	0.03	-0.02	-0.17
0.21	0.27	-0.18	-0.03	-0.54	0.08	-0.47	-0.04	-0.58
0.01	0.49	0.23	0.03	0.59	-0.39	-0.29	0.25	-0.23
0.04	0.62	0.22	0.00	-0.07	0.11	0.16	-0.68	0.23
0.03	0.45	0.14	-0.01	-0.30	0.28	0.34	0.68	0.18

S_0 =

```
3.34
       2.54
              2.35
                     1.64
                            1.50
                                   1.31
                                          0.85
                                                 0.56
                                                        0.36
```

D_0 =

```
 0.20 -0.06  0.11 -0.95  0.05 -0.08  0.18 -0.01 -0.06
 0.61  0.17 -0.50 -0.03 -0.21 -0.26 -0.43  0.05  0.24
 0.46 -0.13  0.21  0.04  0.38  0.72 -0.24  0.01  0.02
 0.54 -0.23  0.57  0.27 -0.21 -0.37  0.26 -0.02 -0.08
 0.28  0.11 -0.51  0.15  0.33  0.03  0.67 -0.06 -0.26
 0.00  0.19  0.10  0.02  0.39 -0.30 -0.34  0.45 -0.62
 0.01  0.44  0.19  0.02  0.35 -0.21 -0.15 -0.76  0.02
 0.02  0.62  0.25  0.01  0.15  0.00  0.25  0.45  0.52
 0.08  0.53  0.08 -0.03 -0.60  0.36  0.04 -0.07 -0.45
```

The reader can verify that:

$X = T_0 S_0 D_0'$ (except for small rounding errors)
T_0 has orthogonal, unit length columns so $T_0 T_0' = I$
D_0 has orthogonal, unit length columns so $D_0 D_0' = I$

*****

We now approximate X keeping only the first two singular values and the corresponding columns from the T and D matrices. (Note that these are the T and D coordinates used to position the 12 terms and 9 documents, respectively, in the 2-dimensional representation of Figure 1.) In this *reduced model*,

$$X \approx Xhihat = TSD'$$

$X \approx$ T S D'

```
0.22 -0.11     3.34              0.20 0.61  0.46  0.54 0.28 0.00 0.02 0.02 0.08
0.20 -0.07              2.54    -0.06 0.17 -0.13 -0.23 0.11 0.19 0.44 0.62 0.53
0.24  0.04
0.40  0.06
0.64 -0.17
0.27  0.11
0.27  0.11
0.30 -0.14
0.21  0.27
0.01  0.49
0.04  0.62
0.03  0.45
```

Multiplying out the matrices TSD' gives the following estimate of X, $Xhihat$.

Xhihat =

0.16	0.40	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
0.14	0.37	0.33	0.40	0.16	-0.03	-0.07	-0.10	-0.04
0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.20	-0.11
0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
-0.06	0.34	-0.15	-0.30	0.20	0.31	0.69	0.98	0.85
-0.04	0.25	-0.10	-0.21	0.15	0.22	0.50	0.71	0.62

There are two things to note about the *Xhihat* matrix. (1) It does not exactly match the original term by document matrix X (it gets closer and closer as more and more singular values are kept). (2) This is what we want; we don't want perfect fit since we think some of the 0's in X should be 1 and vice versa.

REFERENCES

1. Furnas, G.W., Landauer, T.K., Gomez, L.M., and Dumais, S.T. Statistical semantics: Analysis of the potential performance of key-word information systems. *Bell System Technical Journal*, 1983, *62(6)*, 1753-1806.
2. Tarr, D. and Borko, H. Factors influencing inter-indexer consistency. In *Proceedings of the ASIS 37th Annual Meeting, Vol. 11*, 1974, 50-55.
3. Fidel, R. Individual variability in online searching behavior. In C.A. Parkhurst (Ed.). *ASIS'85: Proceedings of the ASIS 48th Annual Meeting, Vol. 22*, October 20-24, 1985, 69-72.
4. Liley, O. Evaluation of the subject catalog. *American Documentation*, 1954, *5(2)*, 41-60.
5. Bates, M.J. Subject access in online catalogs: A design model. *JASIS*, 1986, *37 (6)*, 357-376.
6. Sparck Jones, K. A statistical interpretation of term specificity and its applications in retrieval. *Journal of Documentation*, 1972, *28(1)*, 11-21.
7. Gomez, L. M. and Lochbaum, C. C. People can retrieve more objects with enriched key-word vocabularies. But is there a human performance cost? In *Proceedings of Interact 84*, London, England, September 1984.
8. Furnas, G.W. Experience with an adaptive indexing scheme. In *Human Factors in Computer Systems, CHI'85 Proceedings*. San Francisco, Ca., April 15-18, 1985.
9. van Rijsbergen, C.J. A theoretical basis for the use of co-occurrence data in information retrieval. *Journal of Documentation*, 1977, *33(2)*, 106-119.
10. Carroll, J.D. and Arabie, P. Multidimensional scaling. In M.R. Rosenzweig and L.W. Porter (Eds.). *Annual Review of Psychology*, 1980, *31*, 607-649.
11. Sparck Jones, K. *Automatic Keyword Classification for Information Retrieval*, Buttersworth, London, 1971.
12. Salton, G. *Automatic Information Organization and Retrieval*. McGraw Hill, 1968.
13. Jardin, N. and van Rijsbergen, C.J. The use of hierarchic clustering in information retrieval. *Information Storage and Retrieval*, 1971, *7*, 217-240.
14. Baker, F.B. Information retrieval based on latent class analysis. *Journal of the ACM*, 1962, *9*, 512-521.
15. Atherton, P. and Borko, H. A test of factor-analytically derived automated classification methods. AIP rept AIP-DRP 65-1, Jan. 1965.
16. Borko, H and Bernick, M.D. Automatic document classification. *Journal of the ACM*, April 1963, *10(3)*, 151-162.
17. Ossorio, P.G. Classification space: A multivariate procedure for automatic document indexing and retrieval. *Multivariate Behavioral Research*, October 1966, 479-524.

18. Salton, G. and McGill, M.J. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

19. Voorhees, E. The cluster hypothesis revisited. *SIGIR*, 1985, 188-196.

20. Koll, M. An approach to concept-based information retrieval. *ACM SIGIR Forum, XIII32-50*, 1979.

21. Raghavan, V. and Wong, S. A critical analysis of vector space model for information retrieval. *JASIS*, 1986, *37(5)*, 279-288.

22. Coombs, C.H. *A Theory of Data.* New York: Wiley, 1964.

23. Heiser, W.J. *Unfolding Analysis of Proximity Data.* Leiden, The Netherlands: Reprodienst Psychologie RUL, 1981.

24. Desarbo, W.S., and Carroll, J.D. Three-way metric unfolding via alternating weighted least squares. *Psychometrika*, 1985, *50(3)*, 275-300.

25. Harshman, R.A. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis. *UCLA Work Papers Phonetics*, 1970, *16*, 86pp.

26. Harshman, R.A. and Lundy, M.E. The PARAFAC model for three-way factor analysis and multi-dimensional scaling. In H.G. Law, C.W. Snyder, Jr., J.A. Hattie, and R.P. McDonald (Eds.). *Research Methods for Multimode Data Analysis*, Praeger, 1984a.

27. Carroll, J.D. and Chang, J.J. Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 1970, *35*, 283-319.

28. Kruskal, J.B. Factor analysis and principal components: Bilinear methods. In H. Kruskal and J.M. Tanur (Eds.). *International Encyclopedia of Statistics*, New York: Free Press, 1978.

29. Furnas, G.W. Objects and their features: The metric representation of two-class data. Ph.D. Dissertation. Stanford University, 1980.

30. Forsythe, G.E., Malcolm, M.A., and Moler, C.B. *Computer Methods for Mathematical Computations* (Chapter 9: Least squares and the singular value decomposition). Englewood Cliffs, NJ: Prentice Hall, 1977.

31. Harshman, R.A. and Lundy, M.E. Data preprocessing and the extended PARAFAC model. In H.G. Law, C.W. Snyder, Jr., J.A. Hattie, and R.P. McDonald (Eds.). *Research Methods for Multimode Data Analysis*, Praeger, 1984b.

32. Jones, W.P. and Furnas, G.W. Pictures of relevance. *JASIS*, 1987, *38(6)*, 420-442.

33. Golub, G.H., Luk, F.T., and Overton, M.L. A block Lanczos method for computing the singular values and corresponding singular vectors of a matrix. *ACM Transactions on Mathematical Software*, 1981, *7(2)*, 149-169.

34. Cullum, J., Willoughby, R.A., and Lake, M. A Lanczos algorithm for computing singular values and vectors of large matrices. *SIAM J. Sci. Stat. Comput.*, 1983, *4(2)*, 197-215.

35. Lesk, M.E. and Salton, G. Relevance assessments and retrieval system evaluation. *Information Storage and Retrieval*, 1969, *4(4)*, 343-359.

36. Amsler, R. Machine-readable dictionaries. In *Annual Review of Information Science and Technology (ARIST), Vol. 19*, 1984, 161-209.

37. Choueka, Y. and Lusignan, S. Disambiguation by short contexts. *Computers and the Humanities*, 1985, *19*, 147-157.

38. Lesk, M.E. How to tell a pine cone from an ice cream cone. In *Proceedings of ACM SIGDOC Conference*, Toronto, Ont., June, 1986.

CONTENTS

LIST OF FIGURES

LIST OF TABLES