Latent Semantic Indexing: An overview

INFOSYS 240 Spring 2000
Final Paper

Barbara Rosario

1) Introduction

Typically, information is retrieved by literally matching terms in documents with those of a query. However, lexical matching methods can be inaccurate when they are used to match a user's query. Since there are usually many ways to express a given concept (synonymy), the literal terms in a user's query may not match those of a relevant document. In addition, most words have multiple meanings (polysemy), so terms in a user's query will literally match terms in irrelevant documents. A better approach would allow users to retrieve information on the basis of a *conceptual topic* or *meaning* of a document.

Latent Semantic Indexing (LSI) [Deerwester et al] tries to overcome the problems of lexical matching by using statistically derived conceptual indices instead of individual words for retrieval. LSI assumes that there is some underlying or latent structure in word usage that is partially obscured by variability in word choice. A truncated singular value decomposition (SVD) is used to estimate the structure in word usage across documents. Retrieval is then performed using the database of singular values and vectors obtained from the truncated SVD. Performance data shows that these statistically derived vectors are more robust indicators of meaning than individual terms.

Section 2 is a review of basic concepts needed to understand LSI. In Section 3, a description of some of the advantages and disadvantages of LSI.

The effectiveness of LSI has been demonstrated empirically in several text collections as increased average retrieval precision but a theoretical (and quantitative) understanding beyond empirical evidence is desirable. Section 4 describes some of the attempts that have been done in this direction. Finally, in Section 5 some applications of LSI.

2) Basic concepts

Latent Semantic Indexing is a technique that projects queries and documents into a space with "latent" semantic dimensions.

In the latent semantic space, a query and a document can have high cosine similarity even if they do not share any terms - as long as their terms are semantically similar in a sense to be described later. We can look at LSI as a similarity metric that is an alternative to word overlap measures like tf.idf.

The latent semantic space that we project into has fewer dimensions than the original space (which has as many dimensions as terms). LSI is thus a method for *dimensionality reduction*. A dimensionality reduction technique takes a set of objects that exist in a high-dimensional space and represents them in a low-dimensional space, often in a two-dimensional or three-dimensional space for the purpose of visualization.

Latent semantic indexing is the application of a particular mathematical technique, called Singular Value Decomposition or SVD, to a word-by-document matrix. SVD (and hence LSI) is a least-squares method. The projection into the latent semantic space is chosen such that the representations in the original space are changed as little as possible when measured by the sum of the squares of the differences.

SVD takes a matrix A and represents it as $\hat{A}$ in a lower dimensional space such that the "distance" between the two matrices as measured by the 2-norm is minimized:

$$\Delta = \left\| A - \hat{A} \right\|_2$$

The 2-norm for matrices is the equivalent of Euclidean distance for vectors. SVD project an *n*-dimensional space onto a *k*-dimensional space where $n >> k$. In our application (word-document matrices), n is the number of word types in the collection. Values of k that are frequently chosen are 100 and 150. The projection transforms a document's vector in n-dimensional word space into a vector in the k-dimensional reduced space.

There are many different mappings from high dimensional to low-dimensional spaces. Latent Semantic Indexing chooses the mapping that is optimal in the sense that it minimizes the distance Δ. This setup has the consequence that the dimensions of the reduced space correspond to the *axes of greatest variation*.[1]

[1] This is closely related to Principal Componet Analysis (PCA), another technique for dimensionality reduction. One difference between the two techniques is that PCA can only be applied to a square matrix whereas LSI can be applied to any matrix.

The SVD projection is computed by decomposing the document-by-term matrix $A_{t\times d}$ into the product of three matrices, $T_{t\times n}$, $S_{n\times n}$, $D_{d\times n}$:

$$A_{t\times d} = T_{t\times n} S_{n\times n} (D_{d\times n})^T$$

where t is the number of terms, d is the number of documents, $n = \min(t,d)$, T and D have orthonormal columns, i.e. $TT^T = D^T D = I$, $rank(A) = r$, $S = diag(\sigma_1, \sigma_2, ..., \sigma_n,)$, $\sigma_i > 0 \text{ for } 1 \le i \le r, \sigma_j = 0 \text{ for } j \ge r+1$.
We can view SVD as a method for rotating the axes of the n-dimensional space such that the first axis runs along the direction of largest variation among the documents, the second dimension runs along the direction with the second largest variation and so forth. The matrices T and D represent terms and documents in this new space. The diagonal matrix S contains the singular values of A in descending order. The i^{th} singular value indicates the amount of variation along the i^{th} axis.

By restricting the matrixes T, S and D to their first $k < n$ rows one obtains the matrixes $T_{t\times k}, S_{k\times k}, (D_{d\times k})^T$. Their product $\hat{A}$

$$\hat{A}_{t\times k} = T_{t\times k} S_{k\times k} (D_{d\times k})^T$$

is the best square approximation of A by a matrix of rank k in the sense defined in the equation $\Delta = \left\| A - \hat{A} \right\|_2$.

Choosing the number of dimensions (k) for $\hat{A}$ is an interesting problem. While a reduction in k can remove much of the noise, keeping too few dimensions or factors may loose important information. As discussed in [Deerwester et al] using a test database of medical abstracts, LSI performance can improve considerably after 10 or 20 dimensions, peaks between 70 and 100 dimensions, and then begins to diminish slowly. This pattern of performance (initial large increase and slow decrease to word-based performance) is observed with other datasets as well. Eventually performance must approach the level of performance attained by standard vector methods, since with k = n factors $\hat{A}$ will exactly reconstruct the original term by document matrix *A*. That LSI works well with a relatively small (compared to the number of unique terms) number of dimensions or factors k shows that these dimensions are, in fact, capturing a major portion of the meaningful structure. [Berry et al.]

One can also prove that SVD is unique, that is, there is only one possible decomposition of a given matrix. That SVD finds the optimal projection to a low-dimensional space is the key property for exploiting word co-occurrence patterns. It is important for the LSI method that the derived $\hat{A}$ matrix does not reconstruct the original term document matrix A exactly. The truncated SVD, in one sense, captures most of the important underlying structure in the association of terms and documents, yet at the same time removes the noise or variability in word usage that plagues word-based retrieval methods. Intuitively, since the number of dimensions, *k*, is much smaller than the number of unique terms, t, minor differences in terminology will be ignored. Terms which occur in similar documents, for example, will be near each other in the k-dimensional factor space even if they never co-occur in the same document. This means that some documents, which do not share any words with a user's query, may nonetheless be near it in *k*-space. This derived representation, which captures term-term associations, is used for retrieval.

2.1) Queries

For purposes of information retrieval, a user's query must be represented as a vector in k-dimensional space and compared to documents. A query (like a document) is a set of words. For example, the user query can be represented by

$$\hat{q} = q^T T_{t\times k} S^{-1}{}_{k\times k}$$

where q is simply the vector of words in the users query, multiplied by the appropriate term weights. The sum of these k-dimensional terms vectors is reflected by the term $q^T T_{t\times k}$ in the above equation, and the right multiplication by $S^{-1}{}_{k\times k}$ differentially weights the separate dimensions. Thus, the query vector is located at the weighted sum of its constituent term vectors. The query vector can then be compared to all existing document vectors, and the documents ranked by their similarity (nearness) to the query. One common measure of similarity is the cosine between the query vector and document vector. Typically, the z closest documents or all documents exceeding some cosine threshold are returned to the user.

2.2) Updating

One remaining problem for a practical application is how to fold queries and new documents into the reduced space. The SVD computation only gives us reduced representations for the document vectors in matrix A. We do not want to do a completely new SVD every time a new query is launched or new documents and terms are added to the collection. There are two alternatives for incorporating new documents and terms currently: recomputing the SVD of a new term-document matrix or folding-in the new terms and documents.
Lets define some terms that are used when discussing updating.

- *Updating* refers to the general process of adding new terms and/or documents to an existing LSI-generated database. Updating can mean either folding-in or SVD-updating.
- *S VD-updating* is the new method of updating developed in [O' Brien].
- *Folding-in* terms or documents is a much simpler alternative that uses an existing SVD to represent new information.
- *Recomputing* the SVD is not an updating method, but a way of creating an LSI-generated database with new terms and/or documents from scratch which can be compared to either updating method.

Recomputing the SVD of a larger term-document matrix requires more computation time and, for large problems, may be impossible due to memory constraints. In contrast, folding-in is based on the existing latent semantic structure, the current $\hat{A}$, and hence new terms and documents have no effect on the representation of the pre-existing terms and documents. Folding-in requires less time and memory but can have deteriorating effects on the representation of the new terms and documents.

In addition, in order to handle large corpora efficiently we may want to do SVD for only a sample of the documents (for example a third or a fourth). The remaining documents would then be folded in.

Folding-in documents is essentially the process described in the previous section for query representation. Each new document is represented as a weighted sum of its component term vectors. Once a new document vector has been computed it is appended to the set of existing document vectors. Similarly, new terms can be represented as a weighted sum of the vectors for documents in which they appear.

The equation for folding documents into the space can again be derived from the basic SVD equation:

$$A = TSD^T$$

$$T^T A = T^T TSD^T$$

$$T^T A = SD^T$$

So we just multiply the query or document vector with the transpose of the term matrix T (after it has been truncated to the desired dimensionality).

3) Advantages and disadvantages

3.1) Advantages

1. True (latent) dimensions

The assumption in LSI (and similarly for other forms of dimensionality reduction like principal component analysis) is that the new dimensions are a better representation of documents and queries. The metaphor underlying the term "latent" is that these new dimensions are the true representation. This true representation was then obscured by a generation process that expressed a particular dimension with one set of words in some documents and a different set of words in another document. LSI analysis recovers the original semantic structure of the space and its original dimensions.

[Deerwester et al] describe the three major advantages of using the LSI representation with the following labels: synonymy, polysemy, and term dependence.

2. Synonymy

Synonymy refers to the fact that the same underlying concept can be described using different terms. Traditional retrieval strategies have trouble discovering documents on the same topic that use a different vocabulary. In LSI, the concept in question as well as all documents that are related to it are all likely to be represented by a similar weighted combination of indexing variables.

3. Polysemy

Polysemy describes words that have more than one meaning, which is common property of language. Large numbers of polysemous words in the query can reduce the precision of a search significantly. By using a reduced representation in LSI, one hopes to remove some "noise" from the data, which could be described as rare and less important usages of certain terms. (Note however that this would work only when the real meaning is close to the average meaning. Since the LSI term vector is just a weighted average of the different meanings of the term, when the real meaning differs from the average meaning, LSI may actually reduce the quality of the search).

4. Term Dependence

The traditional vector space model assumes term independence and terms serve as the orthogonal basis vectors of the vector space. Since there are strong associations between terms in language, this assumption is never satisfied. While term independence represents the most reasonable first-order approximation, it should be possible to obtain improved performance by using term associations in the retrieval process. Adding common phrases as search items is a simple application of this approach. On the other hand, the LSI factors are orthogonal by definition, and terms are positioned in the reduced space in a way that reflects the correlations in their use across documents. It is very difficult to take advantage of term associations without dramatically increasing the computational requirements of the retrieval problem. While the LSI solution is difficult to compute for large collections, it need only be constructed once for the entire collection and performance at retrieval time is not affected.

3.1) Disadvantages

1. Storage

One could also argue that the SVD representation is more compact. Many documents have more than 150 unique terms. So the sparse vector representation will take up more storage space than the compact SVD representation if we reduce to 150 dimensions. In reality, the opposite is actually true [Hull]. For example, the document by term matrix for the Cranfield collection used in Hull's experiments had 90,441 non-zero entries (after stemming and stop word removal). Retaining only 100 of the possible 1399 LSI vectors requires storing 139,900 values for the documents alone. The term vectors require the storage of roughly 400,000 additional values. In addition, the LSI values are real numbers while the

original term frequencies are integers, adding to the storage costs. Using LSI vectors, we can no longer take advantage of the fact that each term occurs in a limited number of documents, which accounts for the sparse nature of the term by document matrix. With recent advances in electronic storage media, the storage requirements of LSI are not a critical problem, but the loss of sparseness has other, more serious implications.

2. Efficiency

One of the most important speed-ups in vector space search comes from using an inverted index. As a consequence, only documents that have some terms in common with the query must be examined during the search. With LSI, however, the query must be compared to every document in the collection. There are, however, several factors that can reduce or eliminate this drawback. If the query has more terms than its representation in the LSI vector space, then inner product similarity scores will take more time to compute in term space. For example, if relevance feedback is conducted using the full text of the relevant documents, the number of terms in the query is likely to grow to be many times the number of LSI vectors, leading to a corresponding increase in search time. In addition, using a data structure such as the k-d tree in conjunction with LSI would greatly speed the search for nearest neighbors, provided only a partial ordering of the documents is required. Most of the additional costs come in the pre-processing stage when the SVD and the k-d tree are computed, and actual search time should not be significantly degraded. Other query expansion techniques suffer even more heavily from the difficulties described above, and LSI performs relatively well for long documents due to the small number of context vectors used to describe each document. However, implementation of LSI does require an additional investment of storage and computing time. [Hull]

3. LSI and normally-distributed data

Another objection to SVD is that, along with all other least-squares methods, it is really designed for normally-distributed data, but such a distribution is inappropriate for count data, and count data is what a term-by-document matrix consists of. The link between least squares and normal distribution can be easily seen by looking at the definition of the normal distribution:

$$n(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma} \right)^2 \right]$$

where μ is the mean and σ the covariance. The smaller the squared deviation from the mean $(x-\mu)^2$, the higher the probability $n(x;\mu,\sigma)$. So the least squares solution is the maximum likelihood solution. But this is only true if the underlying data distribution is normal. Other distributions like Poisson or negative binomial are more appropriate for term counts. One problematic feature of SVD is that, since the reconstruction $\hat{A}$ of the term-by-document matrix A is based on a normal distribution, it can have negative entries, clearly an inappropriate approximation for counts. A dimensionality reduction based on Poisson would not predict such impossible negative counts. In defense of LSI (and the vector space model in general which can also be argued to assume a normal distrthution), one can say that the matrix entries are not counts, but weights. Although this is not an issue that has been investigated systematically, the normal distribution could be appropriate for the weighted vectors even if it is not for count vectors. [Manning and Schultze]

Ultimately, to decide weather the advantages outweigh the disadvantages, we need to look at the retrieval performance. While [Deerwester et al] have obtained some promising results, they do not show conclusively that retrieval using LSI is superior to the basic vector space model. [Hull] addresses this issue in the context of the routing problem and provides evidence that LSI slightly improves performance for the routing task.

In other words, the winner has still to be proclaimed.

4) Toward a theoretical foundation

Although (little) empirical improved performance has been observed, there is very little in the literature in the way of a mathematical theory that predicts this improved performance. In this session I briefly describe one paper that is an attempt at using mathematical techniques to rigorously explain the empirically observed improved performance of LSI, [Papadimitriou et al.]

Papadimitriou starts citating an interesting mathematical fact due to Eckart and Young, often cited as an explanation of the improved performance of LSI, that states, informally, that LSI retains as much as possible the relative position (and distances) of the document vectors while projecting it to a lower-dimensional space. This may only provide an explanation of why LSI does not deteriorate too much in performance over conventional vector-space methods; it fails to justify the observed improvement in precision and recall.

The theorem says that $\hat{A}$ is a "good approximation" of A in the sense that:

Theorem (Eckart and Young) Among all *t x d* matrices *C* of rank at most *k*, $\hat{A}$ is the one that minimizes

$$\|A - C\|^2 = \sum_{i,j} \left(A_{i,j} - C_{i,j}\right)^2$$

This is what we saw in Section 1. It remains to be seen in what way it improves these retrieval capabilities.

Since LSI seems to exploit and reveal the statistical properties of a corpus, [Papadimitriou et al.] starts with a rigorous probabilistic model of the corpus (i.e. a mathematical model of how corpora are generated). They then model topics as probability distributions on terms. A document is a probability distribution that is a combination of a small number of topics, a corpus is a collection of documents obtained by repeatedly drawing sample documents. Once they have a corpus model, they would like to determine under what conditions LSI results in enhanced retrieval. They would like to prove a theorem stating essentially that *if the corpus is a reasonably focused collection of meaningfully correlated documents, then LSI performs well*. The problem is to define these terms so that (1) there is a reasonably close correspondence with what they mean intuitively and in practice, and (2) the theorem can be proved.

[Papadimitriou et al.] prove results that, although not quite as general as the statement above, definitely point to this direction. In particular, they show that:

In the special case in which

- *(a) there is no style modifier;*
- *(b) each document is on a single topic;*
- *(c) the terms are partitioned among the topics so that each topic distribution has high probability on its own terms, and low probability on all others;*

then LSI, projecting to a subspace of dimension equal to the number of topics, will discover these topics exactly, with high probability assuming that the length of each document in the corpus is large enough.

[Papadimitriou et al.] also shows that, projecting the term-document matrix on a completely random low-dimensional subspace, then with high probability one has a distance-preservation property akin to that enjoyed by LSI. This suggests that random projection may yield an interesting improvement on LSI: we can perform the LSI precomputation not on the original term-document matrix, but on

a low-dimensional projection, at great computational savings and no great loss of accuracy.

Random projection can be seen as an alternative to (and a justification of) sampling in LSI. Reports on LSI experiments in the literature seem to suggest that LSI is often done not on the entire corpus, but on a randomly selected sub corpus (both terms and documents may be sampled, although it appears that most often documents are). There is very little non-empirical evidence of the accuracy of such sampling. [Papadimitriou et al.] suggests a different and more elaborate (and computationally intensive) approach (projection on a random low-dimensional subspace) which can be rigorously proved to be accurate.

Another paper [Ding] establishes a statistical framework for LSI and justifies dimensionality reduction by the statistical significance of latent semantic vectors as measures by the likelihood of the model. The model proposed provides a mechanism to clarify and quantify the argument that LSI reduces the noise while preserving the true dimensions and it does so by checking the statistical significance of the semantic dimensions: if a few semantic dimensions can effectively characterize the data statistically, as indicated by the likelihood of the model, we can believe that they also effectively represent the semantic meaning/relationships as defined by the cosine similarity. The likelihood is the key to the verification of optimal semantic subspace that LSI advocates. [Ding] gives theoretical results to support the existence of such semantic subspace.

5) Applications of LSI

This session surveys several promising application of LSI.

5.1) Information retrieval

The application of Singular Value Decomposition to information retrieval was originally proposed by a group of researchers at Bellcore [Deerwester et al] and called Latent Semantic Indexing in this context.

At this point it should be clear how to use LSI for IR. Regarding the performances, [Berry et al.] reports that for several information science test collections, the average precision using LSI ranged from comparable to 30% better than that obtained using standard keyword vector methods. The LSI method performs best relative to standard vector methods when the queries and relevant documents do not share many words, and at high levels of recall.

5.2) Relevance Feedback

Most of the tests of Relevance Feedback using LSI have involved a method in which the initial query is replaced with the vector sum of the documents the users has selected as relevant. The use of negative information has not yet been exploited in LSI; for example, by moving the query away from documents which the user has indicated are irrelevant. Replacing the users' query with the first relevant document improves performance by an average of 33% and replacing it with the average of the first three relevant documents improves performance by an average of 67%. Relevance feedback provides sizable and consistent retrieval advantages. One way of thinking about the success of these methods is that many words (those from relevant documents) augment the initial query that is usually quite impoverished. LSI does some of this kind of query expansion or enhancement even without relevance information, but can be augmented with relevance information. [Berry et al.]

5.3) Information Filtering

Applying LSI to information filtering applications is straightforward. An initial sample of documents is analyzed using standard LSI/SVD tools. A users' interest is represented as one (or more) vectors in this reduced-dimension LSI space. Each new document is matched against the vector and if it is similar enough to the interest vector it is recommended to the user. Learning methods like relevance feedback can be used to improve the representation of interest vectors over time. Performances studies are encouraging.

5.4) TREC

Recently, LSI has been used for both information filtering and information retrieval in TREC. The queries are very long and detailed descriptions, averaging more than 50 words in length. The fact that the TREC queries are quite rich means that smaller advantages would be expected for LSI or any other methods that attempts to enhance users queries. The big challenge in this collection was to extend the LSI tools to handle collections of this size. The results were quite encouraging. At the time of the TREC conferences it was not reasonable to compute $\hat{A}$ for the complete collection. Instead, a sample of about 70,000 documents and 90,000 terms was used. Such term by document matrices (A) are

quite sparse, containing only .001 - .002% non-zero entries. Computing A_{200}, i.e. the 200-largest singular values and corresponding singular vectors, required about 18 hours of CPU time on a SUN SPARCstation 10 workstation. Documents not in the original LSI analysis were folded-in as previously described in Section 2.2.

Although it is very difficult to compare across systems in any detail because of large pre-processing, representation and matching differences, LSI performance was quite good [Dumais 94]. For filtering tasks, using information about known relevant documents to create a vector for each query was beneficial. The retrieval advantage of 31% was somewhat smaller than that observed for other filtering tests and is attributable to the good initial queries in TREC. For retrieval tasks, LSI showed 16% improvement when compared with the keyword vector methods. Again the detailed original queries account for the somewhat smaller advantages than previously observed. [Berry et al.]

5.5) Cross-Language Retrieval

It is important to note that the LSI analysis makes no use of English syntax or semantics. This means that LSI is applicable to any language. In addition, it can be used for cross-language retrieval - documents are in several languages and user queries (again in several languages) can match documents in any language. What is required for cross-language applications is a common space in which words from many languages are represented.

[Landauer and Littman] describes one method for creating such an LSI space. The original term-document matrix is formed using a collection of abstracts that have versions in more than one language (French and English, in their experiments). Each abstract is treated as the combination of its French English versions. The truncated SVD is computed for this term by combined-abstract matrix A. The resulting space consists of combined-language abstracts, English words and French words. English words and French words that occur in similar combined abstracts will be near each other in the reduced-dimension LSI space. After this analysis, monolingual abstracts can be folded-in: a French abstract will simply be located at the vector sum of its constituent words that are already in the LSI space. Queries in either French or English can be matched to French or English abstracts. There is no difficult translation involved in retrieval from the multilingual LSI space. Experiments showed that the completely automatic multilingual space was more effective than single-language spaces. The retrieval of French documents in response to English queries (and vice versa) was as effective as first translating the queries into French and searching a French-only database. The method has shown almost as good results for retrieving English

abstracts and Japanese Kanji ideographs, and for multilingual translations (English and Greek) of the Bible [Young]. [Berry et al.]

5.6) Matching People Instead of Documents

In a couple of applications, LSI has been used to return the best matching people instead of documents. In these applications, people were represented by articles they had written. In one application [Furnas et al], known as the Bellcore Advisor, a system was developed to find local experts relevant to users' queries. A query was matched to the nearest documents and project descriptions and the authors' organization was returned as the most relevant internal group. In another application, LSI was used to automate the assignment of reviewers to submitted conference papers. Several hundred reviewers were described by means of texts they had written, and this formed the basis of the LSI analysis. Hundreds of submitted papers were represented by their abstracts, and matched to the closest reviewers. These LSI similarities were used to assign papers to reviewers for a major human-computer interaction conference. Subsequent analyses suggested that these completely automatic assignments (which took less than 1 hour) were as good as those of human experts. [Berry et al.]

5.7) Noisy Input

Because LSI does not depend on literal keyword matching, it is especially useful when the text input is noisy, as in OCR (Optical Character Reader), open input, or spelling errors. If there are scanning errors and a word (Dumais) is misspelled (as Duniais), many of the other words in the document will be spelled correctly. If these correctly spelled context words also occur in documents that contained a correctly spelled version of Dumais, then Dumais will probably be near Dunials in the k-dimensional space determined by $\hat{A}$. [Berry et al.]

5.8) Others

[Schutze] and [Gallant] have used SVD and related dimension reduction ideas for word sense disambiguation and information retrieval work. [Hull] and [Yang and

Chute] have used LSI/SVD as the first step in conjunction with statistical classification (e.g. discriminant analysis). Using the LSI-derived dimensions effectively reduces the number of predictor variables for classification. [Wu et al.] in also used LSI/SVD to reduce the training set dimension for a neural network protein classification system used in human genome research. [Berry et al.]

5.10) Open Computational/Statistical Issues

There are a number of computational/statistical improvements that would make LSI even more useful, especially for large collections:

- Computing in efficient way the truncated SVD of extremely large sparse matrices
- Perform SVD-updating in real-time for databases that change frequently, and
- Efficiently comparing queries to documents (i.e., finding near neighbors in high-dimension spaces)

6) References

[Berry et al.] M.W. Berry , S. T. Dumais, G. W. O' Brien, *Using linear algebra for Intelligent Information Retrieval*, 1994

[Deerwester et al] Deerwester, Dumais, Furnas, Lanouauer, and Harshman, *Indexing by latent semantic analysis*, Journal of the American Society for Information Science, 41 (1990), pp. 391-407.

[Ding] Chris H. Q. Ding, *A Similarity-Based Probability Model for Latent Semantic Indexing*, Proc. of SIGIR'99, Berkeley, August 1999

[Dumais 94], S. T. Dumais, *Latent Semantic Indexing (LSI) and TREC-2*., in The Second Text REtrieval Conference (TREC2), D. Harman, ed., March 1994, pp. 105-116. National Institute of Standards and Technology Special Publication.

[Furnas et al] G.W. Furnas et al., *Information retrieval using a singular value decomposition model of latent semantic structure*, in Proceedings of SIGIR, 1988.

[Gallant] I. Gallant, *A practical approach for representing contexts and for performing word sense disambiguation using neural networks*, Neural Computation, 3 (1991).

[Hull] D. Hull, *Improving text retrieval for the routing problem using Latent Semantic Indexing*, in Proceedings of the Seventeenth Annual International ACM-SIGIR Conference, 1994.

[Landauer and Littman], T. K. Landauer and M. L. Littman, *Fully automatic cross-language document retrieval using latent semantic indexing*, in Proceedings of the Sixth Annual Conference of the UW Centre for the New Oxford English Dictionary and Text Research, UW Centre for the New OED and Text Research, Waterloo Ontario, 1990.

[Manning and Schultze] C. D. Manning and H.Schutze, *Foundations of statistical Natural Language Processing*, MIT Press.

[O' Brien] O' Brien, *Information management Tools for Updating an SVD-Encoded Indexing Scheme*. Master's thesis, 1994.

[Papadimitriou et al.] C. H. Papadimitriou, P. Raghavan, H. Tamaki, S. Vempala, *Latent semantic indexing: A probabilistic analysis*, Proc. of Symposium on Principles of Database Systems (PODS), Seattle, Washington, June 1998. ACM Press

[Schutze] H. Schutze, *Dimensions of meaning*, in Proc. of Supercomputing '92.

[Wu et al.] Wu et al. *Neural Networks for full-scale protein sequence classification: Sequence encoding with SVD*, Machine Learning 1994.

[Yang and Chute] Y. Yang and C. G. Chute, *An application of least square fit mapping to text information retrieval,* in Proceedings of ACM-SIGIR Conference, 1993.

[Young] P.G. Young, *Cross-Language Information Retrieval Using Latent Semantic Indexing*, Master's thesis, The University of Knoxville, TN, 1994.