Improving Text Retrieval for the Routing Problem using Latent Semantic Indexing

David Hull

Xerox PARC and Stanford University
3333 Coyote Hill Rd. Palo Alto, CA 94304
internet: hull@parc.xerox.com

Abstract

Latent Semantic Indexing (LSI) is a novel approach to information retrieval that attempts to model the underlying structure of term associations by transforming the traditional representation of documents as vectors of weighted term frequencies to a new coordinate space where both documents and terms are represented as linear combinations of underlying semantic factors. In previous research, LSI has produced a small improvement in retrieval performance. In this paper, we apply LSI to the routing task, which operates under the assumption that a sample of relevant and non-relevant documents is available to use in constructing the query. Once again, LSI slightly improves performance. However, when LSI is used is conjuction with statistical classification, there is a dramatic improvement in performance.

1 Introduction

The vector space model (VSM) [1], which measures the similarity between the query and each document by the weighted inner product of overlapping terms, has long been a standard in information retrieval. The VSM has its flaws, since it ignores both the order and association between terms, but it is hard to find a better method with an equivalent computational complexity. The recent development of Latent Semantic Indexing (LSI) [2] opens a promising avenue of research into methods for using term associations. LSI is a method designed to refine and improve vector space retrieval by transforming the search space to a new coordinate system that captures the most important underlying structure in the association matrix between terms and documents. The method reduces the full term-document matrix to a small number of information-rich LSI vectors, which can then be used in a traditional retrieval model or as the basis for more advanced statistical classification algorithms.

In this paper, we will examine LSI and the vector space model in a slightly different setting than that of the traditional query-based retrieval problem. Instead of basing the analysis only on the initial query, we will assume that the user has a sample of relevant and non-relevant documents which can be used to construct an improved query. The goal is then to find the relevant documents in a new collection or the remaining relevant documents in the collection that the sample is drawn from. This task is equivalent to the routing problem used for system evaluation at the TREC retrieval conference [3]. One can also imagine this task as the second stage in a retrieval algorithm in place of the the traditional strategy of relevance feedback [4].

Since this approach starts with an information-rich environment, we can concentrate on overcoming the problems associated with using term frequencies as the underlying variables in the retrieval model. If our retrieval strategy does not improve performance for the routing task, then it will not produce good results for query-based information retrieval. If an improved variable set is found, it will then be possible to address the question of improving the model in an information-poor environment.

In the following sections, we: (1) present a brief overview of Latent Semantic Indexing, describing the advantages and drawbacks of using it for document retrieval; (2) evaluate the performance of LSI and the vector space model for the routing task using the Cranfield collection; (3) show how the singular value decomposition can be used to generalize the traditional form of the query for the routing problem; (4) describe a statistical classification technique for ranking the documents; (5) evaluate the statistical model on the routing task and show the conditions under which it improves retrieval performance; and (6) make suggestions for future research.

2 Latent Semantic Indexing

Since most retrieval strategies not based on natural language processing ignore the order of terms in a document, the document by term matrix, where each entry gives the term frequency in the corresponding document, is sufficient to represent the collection. However, some of the information contained there could actually hinder the process of document retrieval. A smaller, more tractable representation of terms and documents that retains only the most important information from the original matrix may actually improve both the quality and the speed of the retrieval system. The singular value decomposition (SVD) is a matrix decomposition that produces a reasonable solution to this problem by describing terms and documents as linear combinations of orthogonal (uncorrelated) indexing variables. From Deerwester et al. [2]:

> . . . 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. [. . .] Our aim is . . . to represent terms, documents, and queries in a way that escapes the unreliability, ambiguity, and redundancy of individual terms as descriptors.

The SVD represents the original document by term matrix as the product of three other matrices: $X = DST'$, where the rows of D and T represent the documents and terms respectively in the new coordinate space defined by the columns of the these matrices. The diagonal elements of matrix S assign weights to the factors according to their significance, while the off-diagonal elements are zero.

The power of this decomposition comes from the fact that the new factors are presented in the order of their importance (as measured by the diagonal of S). Therefore, the least important factors can easily be removed by truncating the matrices D, T, and S, i.e. by deleting some of the right-most columns of these matrices. The remaining columns of D will be called the LSI factors. In fact, the truncated matrix $X^* = D^*S^*T'^*$ is the matrix of reduced rank which is most similar to X in terms of the least-squares norm. In subsequent experiments, performance will be measured for varying numbers of LSI factors. The following section presents reasons why using a reduced number of LSI factors could be helpful for retrieval.

2.1 Advantages of LSI

Deerwester et al. [2] describe the three major advantages of using the LSI representation with the following labels: synonymy, polysemy, and term dependence. 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. In essence, LSI can be described as a method for automatic query expansion. It makes use of similar information to the technique proposed by Qiu and Frei [5], which performs query expansion using a term-term similarity matrix.

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. However, using a reduced representation does not guarantee improved performance. 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. An SVD of the term similarity matrix can be used in conjunction with cluster analysis to directly determine the sense of a particular word as demonstrated by Schütze [6].

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. Some alternative methods for incorporating term associations into a retrieval model are given by Wong in [7, 18]. 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.

The theoretical advantages listed above will only be considered valuable if they can be translated directly into improved retrieval peformance. 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 [2]. In the following section, we address this issue in the context of the routing problem. Our experiments will provide evidence that LSI slightly improves performance for the routing task.

2.2 Drawbacks of LSI

Unless LSI substantially improves retrieval performance, the storage and computational costs of the matrix decomposition may outweigh its benefits. While a reduced representation based on a small number of orthogonal variables might appear to cut storage costs substantially, the opposite is actually true. For example, the document by term matrix for the Cranfield collection used in our experiments has 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.

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. Fortunately, several factors 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 [8] in conjuction 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. The disadvantages of LSI in terms of retrieval performance are difficult to quantify. Other useful references to LSI include [9, 10].

2.3 Comparing LSI to the vector space model

Can LSI provide better performance than the vector space model? Deerwester et al. obtain experimental results for two collections, MED and CISI, comparing LSI to SMART [1] and several other term-matching strategies. For MED, LSI improves average precision from .45 to .51 (odds of 29 to 1 of this difference happening by chance), with the largest benefits being found at high recall. The authors note that they perform no stemming or term weighting in their LSI solution. Since term-weighting has a large positive impact on retrieval performance, the improvement found due to LSI appears to be promising. However, the authors also mention that the MED data set was constructed in a way likely to produce unrealistically good results for LSI.

Their second experiment, performed on the CISI collection, found no significant differences between LSI and SMART. In this example, LSI used the same set of index terms as SMART, i.e. stemming was included. In addition, when they compared average performance to the number of LSI factors included for the MED collection, they found found that performance improved with the number of factors until 80-100 factors were used in the model. These experiments suggest that LSI has the potential to improve search performance.

3 Evaluating LSI for the routing problem

We now address the issue of whether LSI improves performance when applied to the routing task. Our experiments use the Cranfield collection, a corpus of documents on aeronautics that is routinely studied in IR experiments. The collection consists of 225 queries, each with a list of relevant documents, and a total of 1399 documents. There are two reasons for choosing this collection: (1) it is small enough that the LSI representation can be computed and stored relatively easily; (2) it has a large number of queries and is considered a fair testing ground for new retrieval methods. However, the conclusions reached in this paper should be tested on larger, more varied document collections in the future.

Unfortunately, the Cranfield collection is not ideal for experiments on the routing problem. In an ideal experiment, the corpus would be divided into a training set of known relevant and nonrelevant documents and a test set of unknown documents to use for evaluation. However, with an average of only eight relevant documents per query, the Cranfield collection does not have enough relevant documents for this strategy to be feasible. We can still conduct a reasonable analysis by using the technique of cross-validation. The strategy is to remove one document at a time from the collection, and then use the remaining documents to try to predict the relevance of the missing document. With cross-validation, we can use nearly all of the collection without biasing the results of the experiment, given a willingness to adjust the routing query to evaluate each document.

Given a set of relevant and non-relevant documents, one must determine the best way to incorporate this information into the routing query. Rocchio [11] suggests using the difference between the mean of the relevant and the mean of the non-relevant documents.

$$Q_{opt} = \frac{1}{n} \sum_{i \in rel} \frac{D_i}{|D_i|} - \frac{1}{N-n} \sum_{j \in nrel} \frac{D_j}{|D_j|}$$

where N is the size of the collection, n is the number of relevant documents, and *rel* and *nonrel* refer to relevant and nonrelevant documents respectively. Subsequent research on relevance feedback [4] has obtained good results using the difference of weighted means to update the original query.

We choose to represent the routing query simply by the mean of the relevant documents for two reasons. First, previous experiments by the author on this collection have found no evidence that using non-relevant documents in the query improves performance. Second, if the non-relevant documents are ignored, cross-validation need only be performed over the relevant documents. Because relevant documents are such a small fraction of the collection, this substantially reduces the computational cost of the experiment.

Cross-validation differs from using the mean of the relevant documents as the query only in the following manner. Each relevant document is compared to the mean of the remaining relevant documents after excluding the document being examined. Therefore, a relevant document cannot favorably influence its own performance. The different queries are normalized to have equal length, and documents are ranked according to their inner product similarity to the routing query. Performance is evaluated using precision and recall.

In this experiment, we compare the performance of LSI with different numbers of factors to the traditional vector space model. A number of pre-processing steps are applied to the data. The raw terms are analyzed using a stemming algorithm and the most frequent function words are removed using a stop list. The terms in each document are then weighted by a function which has shown good performance in previous term-weighting experiments by the author:

$$w_{ij} = \frac{\sqrt{f_{ij}} * \log \frac{N}{n_i}}{\sqrt{|D_i|}}$$

where f_{ij} is the frequency of term j in document i, $|D_i|$ is the length of document i, $\log \frac{N}{n_i}$ is the traditional inverse document frequecy (idf) weighting. This weight function strongly resembles the tf-idf weighting structure and its variants which have been used frequently in other retrieval experiments [12]. The similarity between the routing queries and the relevant documents is measured by the inner product of the corresponding weighted query and document vectors.

In order to provide the best match between our LSI solution and the vector-space solution described earlier, we apply the same weighting scheme to the elements of the term-document matrix before computing the SVD. By matching the weight functions, we insure that the LSI and term-matching solutions are directly comparable. The SVD is then computed using SVDPACK, a sparse matrix SVD program written by Michael Berry and available through NETLIB [13]. The routing query for the LSI solution is chosen to be the average of the relevant documents as represented in LSI space. The similarities are then computed directly from the LSI representation, using cross-validation to compute the scores for the relevant documents.

$$sim(Q, D) = \sum_i q_i * s_i * d_i$$

where q_i and d_i are the weights for the i'th LSI factor for the query and document respectively, and s_i is the i'th diagonal element of the matrix S.

Performance is evaluated using several different strategies. We present scores that are the average of a 10-pt (10%-100%) precision-recall curve (avg.pr.curve), the average of precision evaluated at 1-20 documents retrieved (avg.precision) and the average of recall evaluated at 21-50 documents retrieved (avg.recall) [14]. These measures represent a number of different retrieval strategies that could be adopted by the user. The first scoring method is the traditional measure of evaluation in the IR community. The second method assumes the user is only searching a few documents and is primarily interested in precision. The third method assumes the user is examining a larger sample of documents and wants to maximize recall.

Figure 1 shows the results of the first experiment. The values listed are the average of the appropriate evaluation measure over 219 queries in the Cranfield collection. The six queries with only two relevant

Evaluation Method	VSM	← 40	80	LSI 120	→ 160	200
avg.pr.curve	0.509	0.406	0.507	0.544	0.556	0.567
avg.precision	0.405	0.328	0.403	0.433	0.444	0.451
avg.recall	0.758	0.732	0.796	0.811	0.809	0.811

Figure 1. Performance of LSI and the vector space model

documents are excluded, as cross-validation means that the query would be based on only a single relevant document, which is not a good demonstration of the routing problem. The performance of LSI is measured for 40, 80, 120, 160, and 200 factors retained.

The results from Figure 1 suggest that LSI improves performance when at least 100 LSI factors are included in the model. Performance decreases rapidly as the number of factors drops below that level. Performance increases gradually but levels off as the number of factors approaches 200. The performance curve for average recall rises faster, surpassing the vector space model at fewer than 80 factors and levels off by 120 factors. The best LSI model using 200 factors produces a 5% improvement in performance.

The statistical significance of the differences observed in Figure 1 was judged using the Friedman test and the two-way ANOVA [14]. All LSI models with 120 factors or more are found to be significantly better than the vector space model. The Friedman test suggests that 200 factors is better than 120 when the results are evaluated using the average precision-recall curve, but otherwise no significant differences are found between using 120, 160, and 200 LSI factors.

These results are fairly consistent with those obtained by Deerwester et al. even though the query-based retrieval model relies more on query expansion than the routing problem. This is a bit surprising since query expansion is probably the most significant contribution of the LSI model. For the routing problem, we already use a significant number of relevant documents in the query, so query expansion should not be so helpful. Harman [15] suggests that relevance feedback can be improved by selectively choosing the most important terms to add to the query. Perhaps in the routing problem LSI performs this term selection process in reverse by eliminating the less important components from the relevant documents.

In conclusion, LSI does not greatly improve performance over the vector space model for the routing problem, although the difference is measurable. In the next section, we examine an alternative application of LSI that can be used in conjunction with statistical classification to obtain a significant improvement in retrieval performance.

4 An improved model using LSI and statistical classification

In most retrieval models, the system ranks documents according to their inner product similarity with respect to a query. Most solutions to the routing problem use the same method, although information from known relevant and non-relevant documents is incorporated into the query construction process. Ranking documents according to their inner product similarity to a query is equivalent to applying a one-dimensional linear projection to the document space. A well-constructed query will provide a nice separation between the relevant and non-relevant documents. In general, there is no a priori reason to believe that a good linear separation of the relevant documents exists. If only the original query is available, it is hard to imagine a better method for identifying relevant documents. However, the routing problem has far more information that can be exploited. In particular, we can take advantage of the distribution of the relevant documents.

Statistical classification methods are designed to distinguish between groups (relevant and non-relevant) based on the distribution of the elements (documents) within each group. The general statistical classification problem can be described as follows. A population consists of two or more groups, and there exists a training sample for which the class of each element is known and a test sample for which the class is unknown. The goal is to produce a classification rule which will predict the class of the unknown elements. The classification rule is generated from the training sample based on a number of predictor variables that have been measured for both the known and unknown populations. The routing problem is merely a special case of the classification problem, since there are only two groups, relevant and nonrelevant.

The challenge comes in producing a good set of predictor variables. If there are too many predictor variables, the classifier will overfit the training data, which will lead to poor results on the test data. To give an extreme example, consider using terms as predictor variables. It is likely that every relevant document in the training sample has one or more unique terms. If one uses these terms to define the classification rule, then the classifier will perfectly identify the relevant documents in the training sample, but it will have no power to discriminate over the test sample.

For the routing problem, there must be significantly fewer predictor variables than relevant documents before it is possible to obtain good estimates of the parameters in the classification model. Based on this criterion, terms could not be used as predictor variables for classification. The LSI decomposition substantially reduces the dimension of document space. But even when the document representation is based on 100-200 LSI factors, it is very hard to produce a good classification rule for queries with 30 or fewer relevant documents.

Current methods produce a good classification rule based on a single predictor variable, the inner product similarity score with respect to the mean. It must be possible to produce a better rule using two or more predictor variables. A variant of Latent Semantic Indexing can be used to produce these variables. One could consider using a small subset of the factors from the original LSI solution, but these variables are based on the distribution of the documents across the entire collection and thus are poorly suited to distinguish between documents in the local region that contains the majority of the relevant documents. Therefore, we choose instead to compute a new LSI decomposition based only on the relevant documents.

In order to obtain the local LSI solution, we apply the singular value decomposition to a matrix whose rows are the relevant documents and whose columns are the factors of the original LSI matrix representation. This produces a new set of local LSI factors. We can retain as many local LSI factors as we wish to have predictor variables. The document scores with respect to each predictor variable are simply the inner product of the document representation with the corresponding local LSI factor.

One might ask why these newly derived variables should effectively separate the relevant and non-relevant documents. Each LSI factor is chosen to maximize the variance explained subject to the constraint that it must be orthogonal to all previously computed factors. In a sense, we are maximizing the information about the relevant documents that can be retained in a lower dimensional representation. The effectiveness of this method is best demonstrated using a picture.

Figure 4 plots the document scores for two local LSI factors taken from the results for the first query from the Cranfield collection. The large circles are relevant documents while the smaller squares represent non-relevant documents. The x and y coordinates of the points are the inner product scores of the document with respect to the first and second local LSI factors. Since these dimensions are not particularly meaningful for the non-relevant documents, their scores tend to fall in a cluster around the origin. The relevant documents are much more widely distributed, illustrating how these dimensions manage to capture the structure of the relevant documents.

For comparison, Figure 5 shows a one-dimensional projection onto the mean of the relevant documents. The non-relevant documents are replaced by a smooth estimate of their density function to improve the clarity of the picture. It is also relatively successful at separating the relevant and non-relevant documents, although the separation is not as clean as the two-dimensional representation in Figure 4.

The local LSI method has an increased cost because a separate SVD be computed for each query. However, the SVD is only being applied to the relevant documents, which means that the matrix is far smaller than the one used in the initial LSI decomposition. In addition, since documents are being represented in LSI space, the matrix has at most a few hundred columns. For a matrix of this size, the first few singular vectors that are required can be computed in only a few CPU seconds. However, a similarity score must now be computed for each factor of the new projection, which multiplies the compute time and the storage requirement by the number of local LSI factors. We now describe how statistical classification is used to identify the relevant documents.

Discriminant analysis [16] is a commonly used statistical classification technique. Essentially, it characterizes each population by its estimated mean vector and covariance matrix measured over the known observations. Each unknown observation is classified into the group with the nearest mean vector, scaled for the shape of the covariance matrix. For observation vectors x_j and groups g_i with n_i known members, the mean vector $\bar{x}_i$ and covariance matrix S_i are defined as:

$$\bar{x}_i = \sum_{j \in g_i} \frac{x_j}{n_i} \quad \text{and} \quad S_i = \frac{1}{n_i - 1} \sum_{j \in g_i} (x_j - \bar{x}_i)(x_j - \bar{x}_i)'$$

An unknown observation x is classified into the group with the nearest mean vector based on the Mahalanobis distance metric:

$$\text{dist}(x) = (x - \bar{x}_i)' S_i^{-1} (x - \bar{x}_i)$$

In the case of text-based retrieval, we are less interested in classifying a document as relevant or non-relevant than with ranking all the documents in terms of their probability of relevance. This can be accomplished by assigning each document a score as follows:

$$\text{score}(x_p) = \text{dist}(x_p, \bar{x}_{rel}) - \text{dist}(x_p, \bar{x}_{nrel})$$

Documents are ranked in increasing order of their scores, which reflect their relative likelihood of belonging to the relevant population. For convenience, we will refer to this method as TDA, for text-based discriminant analysis. Note that any other method of classification could be substituted at the this point, such as classification trees or neural networks.

5 Testing the performance of TDA

We return to the Cranfield collection to obtain an objective evaluation of the performance of TDA. Cross-validation [16] will again be used to obtain unbiased estimates of the scores of the relevant documents. The initial experiment will compare the VSM and LSI with 200 factors to a TDA model using two local LSI factors.

Eval. Method	VSM	LSI	TDA2
avg.pr.curve	0.509	0.567	0.760
avg.precision	0.405	0.451	0.604
avg.recall	0.758	0.811	0.830

Figure 2. Performance of TDA vs. LSI and the vector space model

The results of this experiment are presented in Figure 2. When evaluation is based on average precision at fixed recall, TDA combined with local LSI achieves more than three times the amount of improvement as was attained by using LSI instead of the VSM. Precision averaged over documents retained yields similar performance. Recall averaged over documents retained shows a much smaller improvement for TDA. The differences for the first two measures are statistically significant according to both an ANOVA and the Friedman test. The difference in average recall is only significant for the Friedman test. This discrepancy indicates that TDA is performing worse than LSI for a small number of queries, but better for most of the remaining queries.

We repeated the experiment using three, four, and five local LSI factors with TDA. The results, broken down by the number of relevant documents in the query, are presented in Figure 3. The local LSI decomposition cannot produce more dimensions than there are relevant documents. Since cross-validation removes one relevant document, there must be more relevant documents than local LSI factors to obtain a valid solution. In addition, when the number of dimensions selected approaches the number of relevant documents, there is overfitting to the training data.

num.rel	queries	LSI	TDA2	TDA3	TDA4	TDA5
3	29	0.568	0.363	NA	NA	NA
4	19	0.665	0.754	0.465	NA	NA
5	26	0.560	0.854	0.693	0.433	NA
6	28	0.485	0.853	0.795	0.642	0.463
7-8	36	0.631	0.877	0.906	0.843	0.693
9-10	29	0.606	0.852	0.908	0.928	0.894
11+	52	0.515	0.747	0.811	0.855	0.880

Figure 3. Precision averaged at 10%-100% recall, broken down by number of relevant documents

Figure 3 demonstrates that performance continues to improve as additional factors are added to the model, provided that the number of dimensions is less than half the number of relevant documents. However, the improvement becomes less and less significant as more dimensions are added. If both a training and test set are available, one can estimate the optimal number of local LSI factors using cross-validation over the training set. The final results come as no surprise. TDA makes use of much more information from the sample of relevant documents than is contained in a simple one dimensional projection such as the vector mean. More information in the model translates directly into better performance.

6 Conclusions and directions for future work

The vector space model has long been used as a basic framework for developing new retrieval methods. It is difficult to devise a retrieval strategy that performs better with an equivalent amount of computation. However, the vector space model has some significant problems. It assumes that terms are independent and thus ignores term associations. Latent Semantic Indexing addresses this problem by re-expressing the term-document matrix in a new coordinate system designed to capture the most significant components of the term association structure. Experiments run on the Cranfield collection demonstrate that LSI can improve retrieval performance for the routing problem if enough LSI factors are included in the analysis. However, the small performance gains measured may not justify the additional computing and storage costs associated with constructing and maintaining the LSI factors.

However, the LSI solution only needs to be as good as the VSM in order to useful as a component of the TDA model. When we apply the SVD again to the matrix of relevant documents, we can obtain a projection of the documents into a small number of local LSI factors that nicely separates the relevant and non-relevant documents. When this document space is analyzed using discriminant analysis, retrieval performance improves significantly. Performance continues to improve when more factors are added to the model, as long as there are more than twice as many relevant documents as factors.

TDA does take considerably more time to compute than the vector space model. Even so, TDA is still much faster than many of the alternative strategies for incorporating term associations into the retrieval model such as current applications of neural networks [7, 17] or the generalized vector space model [18], since it is applied to such a low dimensional subspace. In addition, time is a much less important issue for the routing problem. Of course, the SVD of the term-document matrix must also be computed, but this need only be done once, prior to constructing any routing queries. Future research will extend this strategy to query-based information retrieval, for which far less information is available. Although TDA cannot be applied without a set of known relevant documents, one can envision a form of relevance feedback based on TDA that has the potential to substantially improve performance.

References

1. Gerard Salton, editor. *The SMART retrieval system: Experiments in Automatic Document Processing.* Prentice-Hall, 1971.
2. S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.
3. Donna Harman. Overview of the first TREC conference. In *Proc. of the 16th ACM/SIGIR Conference*, pages 36–47, 1993.
4. Gerard Salton and Christopher Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288–297, 1990.
5. Yonggang Qiu and H.P. Frei. Concept based query expansion. In *Proc. of the 16th ACM/SIGIR Conference*, pages 160–169, 1993.
6. Hinrich Schütze. Dimensions of meaning. In *Proceedings of Supercomputing '92*, pages 787–796, 1992.
7. S.K.M. Wong, Y.J. Cai, and Y.Y. Yao. Computation of term associations by a neural network. In *Proc. of the 16th ACM/SIGIR Conference*, pages 107–115, 1993.
8. J. Friedman, J. Bentley, and R. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977.
9. G. Furnas, S. Deerwester, S. Dumais, T. Landauer, R. Harshman, L. Streeter, and K. Lochbaum. Information retrieval using a singular value decomposition model of latent semantic structure. In *Proc. of the 11th ACM/SIGIR Conference*, pages 465–480, 1988.
10. B.T. Bartell, G.W. Cottrell, and R.K. Belew. Latent semantic indexing is an optimal special case of multidimensional scaling. In *Proc. of the 15th ACM/SIGIR Conference*, pages 161–167, 1992.
11. J.J. Rocchio. Relevance feedback in information retrieval. In Gerard Salton, editor, *The SMART retrieval system: Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall, 1971.
12. Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.
13. M. Berry. Large scale singular value computations. *International Journal of Supercomputer Applications*, 6(1):13–49, 1992.
14. David Hull. Using statistical testing in the evaluation of retrieval performance. In *Proc. of the 16th ACM/SIGIR Conference*, pages 329–338, 1993.
15. Donna Harman. Relevance feedback revisited. In *Proc. of the 15th ACM/SIGIR Conference*, pages 1–10, 1993.
16. Geoffrey J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*, pages 52–64, 341–346. Wiley, 1992.
17. Ross Wilkinson and Philip Hingston. Using the cosine measure in a neural network for document retrieval. In *Proc. of the 14th ACM/SIGIR Conference*, pages 202–210, 1991.
18. S.K.M. Wong, W. Ziarko, and P.C.N. Wong. Generalized vector space model in information retrieval. In *Proc. of the 8th ACM/SIGIR Conference*, pages 18–25, 1985.

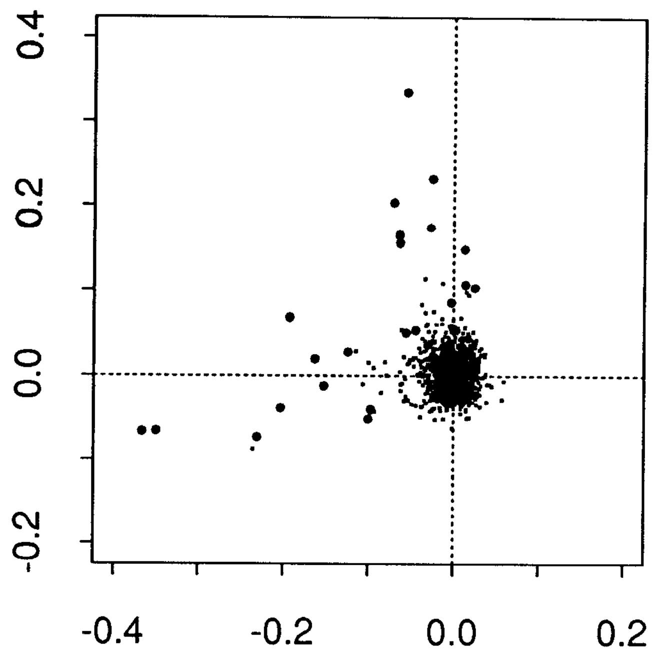

Figure 4: LSI projection

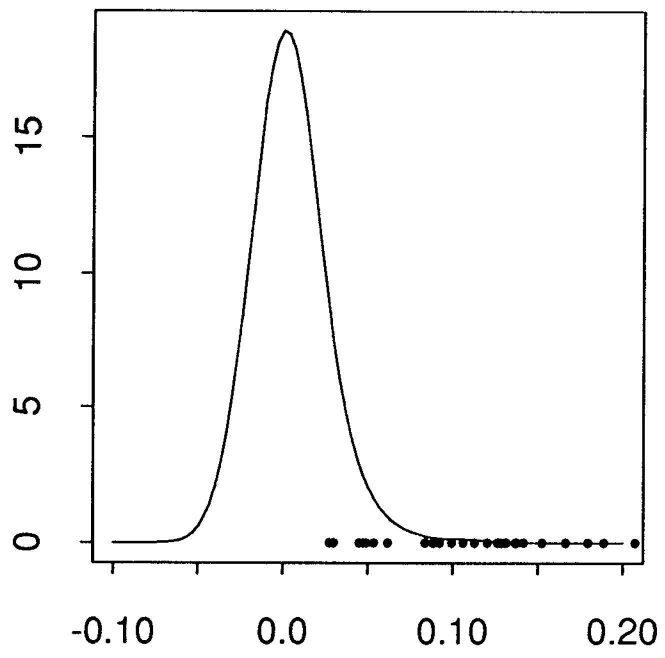

Figure 5: Mean projection