On the Use of the Singular Value Decomposition for Text Retrieval [*]

Parry Husbands, Horst Simon, and Chris Ding [†]

1 Introduction

The use of the Singular Value Decomposition (SVD) has been proposed for text retrieval in several recent works [2, 6]. This technique uses the SVD to project very high dimensional document and query vectors into a low dimensional space. In this new space it is hoped that the underlying structure of the collection is revealed thus enhancing retrieval performance.

Theoretical results [9, 3] have provided some evidence for this claim and to some extent experiments have confirmed this. However, these studies have mostly used small test collections and simplified document models. In this work we investigate the use of the SVD on large document collections. We show that, if interpreted as a mechanism for representing the terms of the collection, this technique alone is insufficient for dealing with the variability in term occurrence.

Section 2 introduces the text retrieval concepts necessary for our work. A short description of our experimental architecture is presented in Section 3. Section 4 describes how term occurrence variability affects the SVD and then shows how the decomposition influences retrieval performance. A possible way of improving

[*]This work is supported by the Director, Office of Science, Office of Laboratory Policy and Infrastructure Management, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098. Computing resources were supported by the Director, Office of Advanced Scientific Computing Research, Division of Mathematical, Information, and Computational Sciences, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098. Work was also supported by the NSFC under Project No. 19771073 and the National Science Foundation under Grant Nos. CCR-9619542 and CCR-9901986.

[†]NERSC Division, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA, 94720 ({prjhusbands,hdsimon,chqding}@lbl.gov)

SVD-based techniques is presented in Section 5 and we conclude in Section 6.

2 Text Retrieval Concepts

In text retrieval (see [4, 10, 1] for treatments of some of the issues), a simple way to represent a collection of documents is with a term-document matrix D where $D(i,j)$ gives the number of occurrences of term i in document j. Queries (over the same set of terms) are similarly represented. The similarity between document vectors (the columns of term-document matrices) can be found by their inner product. This corresponds to determining the number of term matches (weighted by frequency) in the respective documents. Another commonly used similarity measure is the cosine of the angle between the document vectors. This can be achieved computationally by first normalising (to 1) the columns of the term-document matrices before computing inner products.

2.1 Term Weighting

In the above description frequency counts were used as the entries of the term-document matrix. In practice these counts are typically scaled using various term weightings in order to cancel out the dominating effects of frequent terms. One scheme that is commonly used is Inverse Document Frequency (IDF) weighting. This technique multiplies $D(i,j)$ by $w(i)$ where

$$w(i) = \log_2(\frac{\text{Number of documents}}{\text{Number of documents with term } i} + 1)$$

This scheme gives very frequent terms low weight and elevates rare (and hopefully more discriminating) terms.

In the discussion to follow we will denote by "term matching" the retrieval scheme where IDF weighting is used prior to document length normalisation on both the matrix of documents (D) and queries (Q). The matrix of scores is then computed by:

```
Scores = D'*Q
```

2.2 LSI

Latent Semantic Indexing (LSI, [2]) attempts to project term and document vectors into a lower dimensional space spanned by the true "factors" of the collection. This uses a truncated Singular Value Decomposition (SVD) of the term-document matrix D.

If D is an $m \times n$ matrix, then the SVD of D is

$$D = USV'$$

where U is $m \times n$ with orthonormal columns, V is $n \times n$ with orthonormal columns, and S is diagonal with the main diagonal entries sorted in decreasing order. LSI uses a truncated SVD of the term-document matrix where D is approximated by

$$D \approx U_k S_k V_k'$$

where $U_k = U(:, 1 : k)$ (the first k columns of U), $V_k = V(:, 1 : k)$, and $S_k = S(1 : k, 1 : k)$ (the upper left k by k part of S). This gives the best rank k approximation to the original matrix.

Because a full SVD is not required the truncated SVD is usually computed by an iterative technique such as the Lanczos method. The SVDs in this report were computed with the PARPACK software package [8] (as well as TRLAN [12] for verification). See [5] for a more complete treatment of the SVD and related decompositions.

The matrix of scores can then be computed by the product $V_k S_k U_k' Q$. Traditionally these scores are computed by first projecting the queries into k-dimensional space (by $S_k U_k' Q$) and then finding the cosines of the angles with V_k. In this representation the columns of $S_k U_k'$ are identified as the "projected terms" and the columns of V_k are identified as the "projected documents". [1]

Note that the new representation of term i is $S_k U_k' e_i$ and the new representation of document j is $S_k^{-1} U_k' D(:, j)$ ($D(:, j)$ denotes the jth column of matrix D). Note that because

$$D(:, j) = \sum_{i=1}^{m} D(i, j) e_i$$

the new representation of document j can be written as

$$\sum_{i=1}^{m} D(i, j) S_k^{-1} U_k' e_i.$$

Ignoring for now the diagonal scaling we see that the "projected documents" are simple linear combinations of the projected terms.

2.3 Evaluation

In response to a query, a text retrieval system returns an ordered list l of the documents where $l(1)$ is the most relevant, $l(2)$ is the second most relevant, and so on. The standard way to evaluate the performance of a system is to obtain these lists on pre-judged queries and compute precision and recall. At point i, the precision is the number of relevant documents in the first i elements of l (denoted by $l(1 : i)$) divided by i. This is a measure of the "accuracy" of the retrieval: the fraction of the documents returned that are relevant. The recall is the number of relevant documents in $l(1 : i)$ divided by the total number of relevant documents. This is a measure of the "completeness" of the retrieval: the fraction of all relevant documents returned. For each query these measures are computed at each i from 1 to the number of documents. Precision values at fixed recall levels (typically

[1] There is some disagreement about using U_k', $S_k U_k'$, or $S_k^{-1} U_k'$ as the "projected terms". In this work we use $S_k U_k$ primarily because the term-term similarity matrix DD' can be decomposed as US^2U' if $D = USV'$. Hence the rows of US naturally correspond to the rows of D.

interpolated to 0,.1,.2,...,1) are noted and then averaged. A sample precision/recall curve for the MEDLINE test set (with 8847 terms and 1033 documents) using term matching and LSI is shown is Figure 1.

In precision/recall terms, higher curves are better as they indicate a higher percentage of relevant documents at each recall level. In the discussion that follows we will be evaluating various algorithms for text retrieval based on their precision/recall performance.

2.4 LSI Performance

Experiments with LSI have primarily used small data sets. The primary reason for this is the complexity (in both time and space) of computing the SVD of large, sparse term-document matrices. Nevertheless, early results were encouraging. Figure 1 compares LSI using with $k = 100$ to term matching for the small MEDLINE collection. Here IDF weighting was used and the term-document matrix was normalized prior to decomposition. The cosine similarity measure was used in both cases.

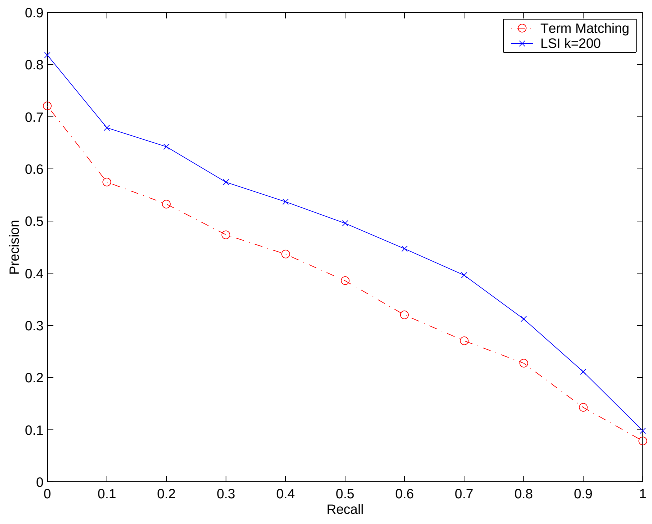

Figure 1. *LSI vs. Term Matching on MEDLINE*

Performance on very large collections is not as good. Figure 2 shows LSI using $k = 200$ on TREC6 [11], a collection with 115000 terms and 528155 documents. Experiments with different numbers of factors up to 1000 have shown similar performance. Note that the computational resources needed for using more than 1000 factors make this impractical for all but the largest supercomputers.

In the rest of this paper, we will investigate reasons for this drop in perfor-

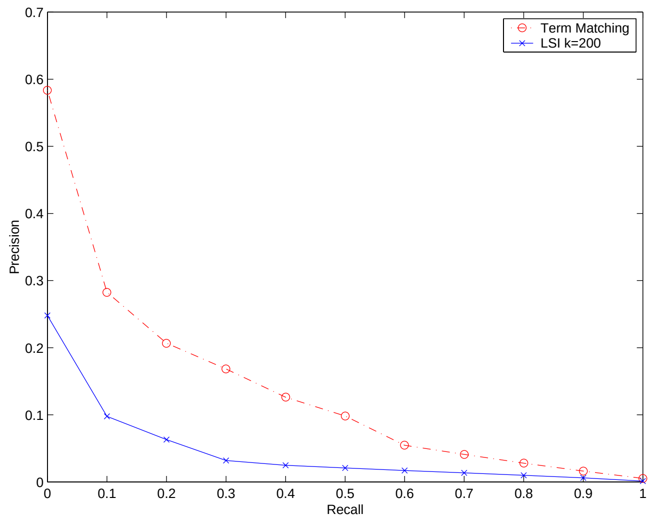

Figure 2. *LSI vs. Term Matching on TREC6*

mance and attempt to change the projection process in order to rectify this problem. A major factor will be the norm distribution of the projected terms, discussed in Section 4.

3 Software Used

For the experiments in this paper we used the MATLAB*P system [7]. MATLAB*P enables users of supercomputers to transparently work on large data sets within Matlab. Through the use of an external server (that stores and operates on data) and Matlab's object oriented features we can handle data as though it were "in" Matlab. In this way, we were able to run our experiments in parallel on NERSC's Cray T3E and no changes had to be made when moving from small to large collections. For example, if `A` is the term document matrix and `Q` is a matrix of queries, to investigate LSI we can type,

```
[U,S,V]=svds(A,k);      % Perform a truncated SVD
newTerms=U*diag(S);     % Compute the projected terms
newA=V';
newQ=newTerms'*Q;       % Get new representation for queries
% Use normcols for cosine measure and find the similarities
Scores=normcols(newA)'*normcols(newQ);
```

For the TREC6 collection, computing the SVD above for ($k = 1000$) takes approximately 104 minutes using 64 T3E processors. Computing and graphing precision/recall curves from pre-judged queries also takes place in MATLAB*P using simple m-file scripts.

4 Norm Distribution of Terms and Impact on Retrieval Performance

The norms (lengths) of the rows of $U_k S_k$ (in addition to their directions) have great influence on the representations of the documents and queries. As Figure 3 and Table 1 show, there is great variability in term norm. In this section we will attempt to explain this variability and its effect on retrieval performance.

Because projected documents and queries are simple *linear combinations* (c.f. Section 2.2) of the projected terms, terms with low norm contribute very little to the representations of documents and queries. The cosine similarity measure comes into play too late: *after* the documents and queries have been projected. Thus, if searching for a term that happens to have low norm, the documents that contain it will have only a small component of that term and be dominated by other terms making it difficult for retrieval.

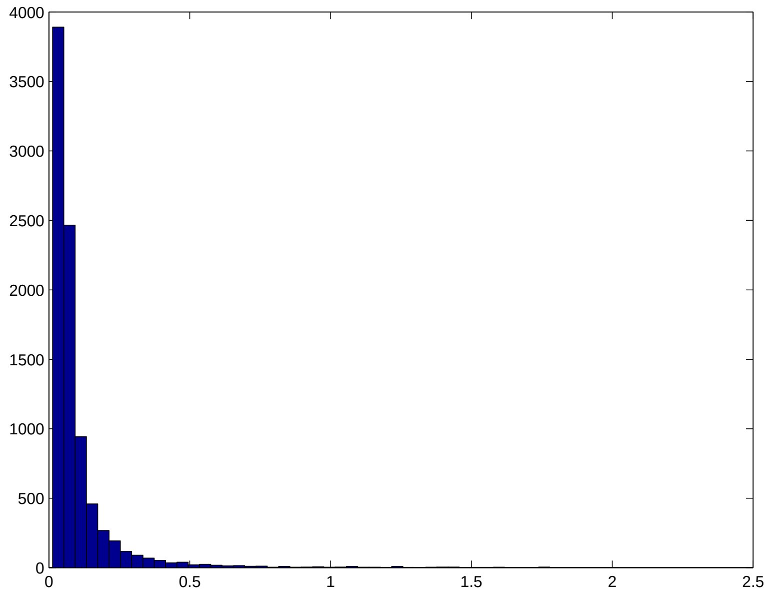

Figure 3. *Histogram of MEDLINE ($k = 100$) term norms.*

Currently a theoretical explanation for the norm distribution has not been proposed. However, we can empirically study the phenomenon in an attempt to determine its cause. Figure 4 plots IDF weight vs. term norm for the MEDLINE test set. We see that the lowest norm terms have the highest IDF weights. This implies that lowest norm terms are those with the *lowest frequencies* in the collection.

As an example of this, consider the TREC6 query that contains the words "polio, poliomyelitis, disease, world, control, post". For this query, the word "polio" is clearly the most important word. It has IDF weight 11.75 but norm 0.16 (k=300). The word "disease" has weight 6.17 and a much higher norm of 3.44. It comes as no surprise, therefore, that the top documents returned for this query are all about disease eradication efforts, but for diseases other than polio (malaria, tuberculosis, AIDS, etc.).

The popular TFIDF weighting scheme does little to mitigate the effect of low term norm. Table 1 shows the range of norms and IDF weights for a few test collections. The lowest term norms are typically orders of magnitude away from the highest IDF weights, hinting at IDF's inadequacy. We can therefore see that the effect of IDF is lost after projection.

Because the columns of U_k are scaled by the singular values, these have a contributing effect on term norm distribution and the projected documents. Figure 5 plots the singular values of the MEDLINE and TREC6 collections. It is interesting to note that after an initial drop the singular values decay very slowly over the displayed range.

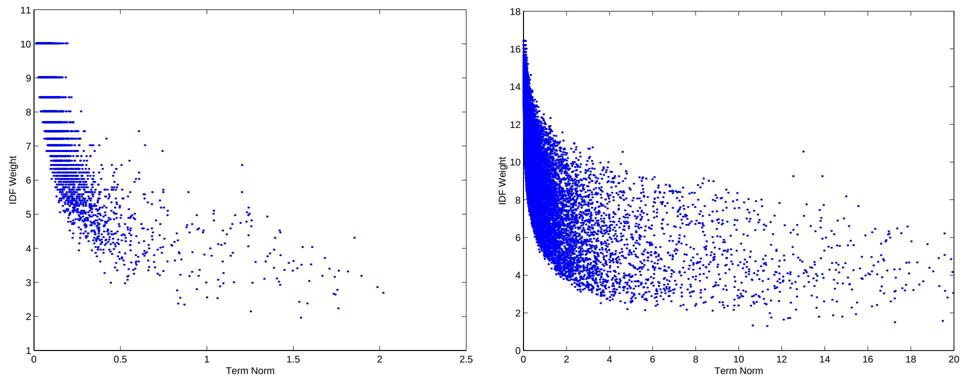

Figure 4. *IDF weight and term norm for the MEDLINE (left) and TREC6 (right) collections. For TREC6 terms with norm > 20 (42 in total) were are not displayed.*

Collection	k	Min norm	Max norm	Min IDF	Max IDF
NPL	100	$2.5e-3$	$5.4e+0$	2.5	13.5
MED	100	$1.3e-2$	$2.0e+0$	1.9	10.0
TREC6	300	$1.5e-4$	$1.5e+2$	1.3	16.4

Table 1. *Term norms and IDF weights for text collections*

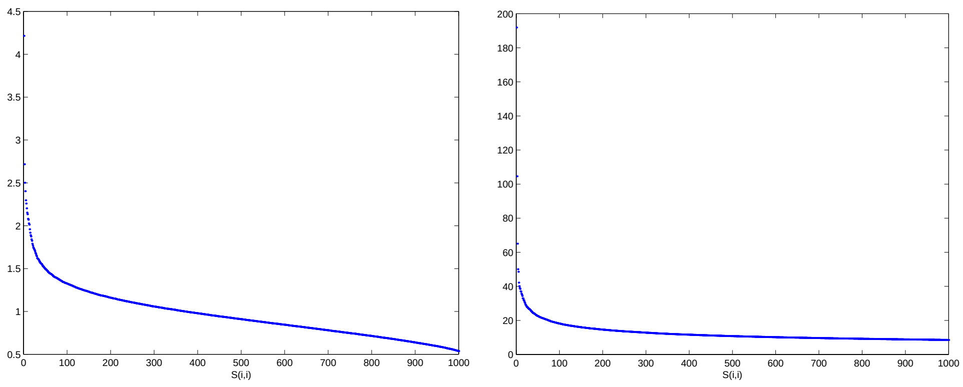

Figure 5. *The first 1000 singular values of the MEDLINE (left) and TREC6 (right) collections.*

5 A Remedy

In this section we attempt to remedy the situation by recapturing the effectiveness of IDF. We do this by first re-examining the way documents are created in term matching. In term matching, we start with unit orthogonal vectors as terms (e_j, see Section 2.2) that are then scaled using term weighting. Finally, documents are created by frequency weighted sums of these (scaled) terms. When we use LSI we perform a similar procedure: we create documents by frequency weighted sums of the *projected* terms. The major difference with term matching, however, is that these projected terms are neither orthogonal nor scaled in proportion to any term weights. The non-orthogonality is desirable: the whole motivation is to have similar terms come closer together. However, as discussed in Section 4 the scaling (or norms) of the projected terms can have a negative impact on retrieval performance.

One simple fix is to re-scale the projected terms so that they are all unit vectors. In this way we can benefit again from term weighting. This scheme, denoted by NLSI (for Normalised LSI), is described below:

- Compute the SVD with k factors U_k, S_k, V_k
- Compute the projected terms $U_k \times S_k$
- Normalise the rows of the projected terms
- Project the documents and queries using the normalised projected terms (note that the document matrix already incorporates term weighting, and so we do not need to scale again)
- Find the scores using the cosine similarity measure

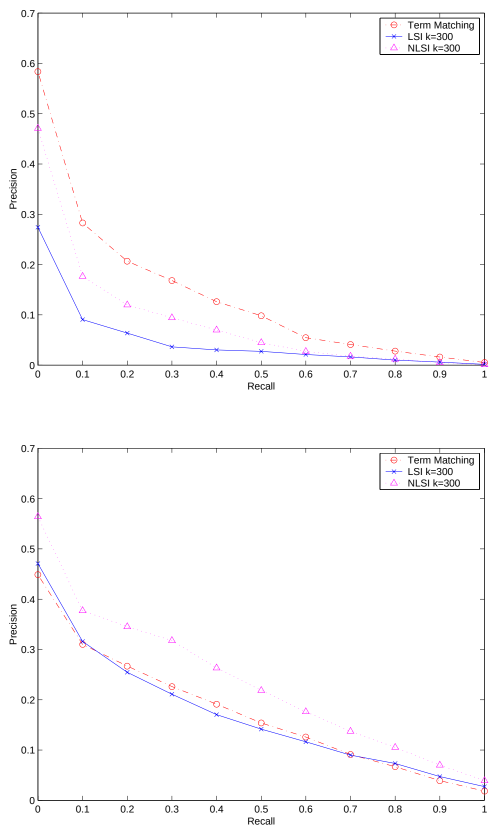

Figure 6. *TREC6 (top) and NPL (bottom) with projected term normalization*

Figure 6 shows the results of using this procedure on the TREC6 and NPL (7491 terms and 11429 documents) test sets. While not a panacea, re-scaling the projected terms has a positive effect on LSI performance for the NPL and TREC6 collections. For NPL, we outperform term matching and for TREC6 we improve on LSI, but still fall short of term matching. For MED, performance seems to depend on the number of factors used as Figure 7 shows. This suggests that we may also need to investigate the orientations (positions in k-dimensional space) of the projected terms in addition to their lengths.

6 Conclusions

LSI attempts to project the documents of a collection into a lower dimensional space in order to improve retrieval performance. This work examines the properties of SVD-based projections in order to determine whether they agree with our intuition

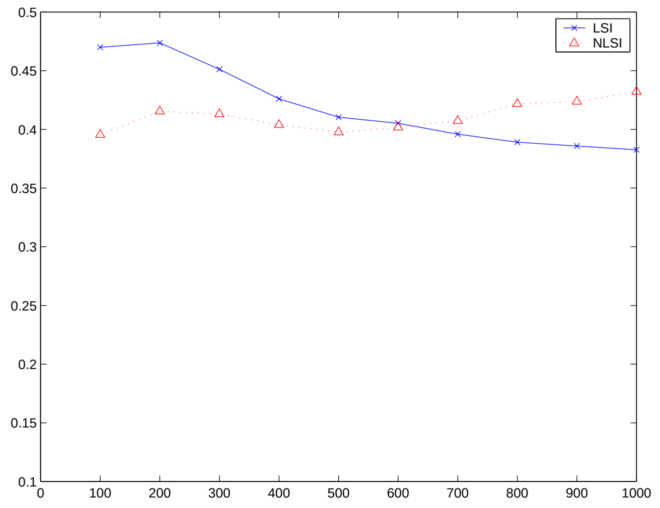

Figure 7. *Average precision for MED with projected term normalization. Each point represents, for each scheme and value of k, the mean of the precision at 11 recall levels (0,0.1,0.2,...,1)*

about IR concepts. The lower dimensionality of the space is intuitively desirable; terms that are related "should" be brought closer together (the cluster hypothesis). However, other properties of the SVD may not match our intuition. The main focus of this paper is the examination of the influence of term norm on retrieval performance. We have seen that rare terms (with low norm) contribute very little to the final LSI representation of documents sometimes resulting in poor retrieval performance.

The properties described above are by no means exhaustive. Others include the enforcement of orthogonality of the columns of U and V, the distribution of the projected documents along each axis, and the interpretability of the singular vectors as "topics". All of these are candidates for future exploration in an effort to fully understand the nature not only of LSI, but of other projection-based approaches to text retrieval.

Bibliography

[1] M. W. Berry and M. Browne. *Understanding Search Engines: Mathematical Modeling and Text Retrieval.* SIAM, 1999.

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

[3] Chris H. Q. Ding. A similarity-based probability model for latent semantic indexing. In *Proceedings of the 22nd ACM/SIGIR Conference*, pages 58–65, 1999.

[4] William B. Frakes and Ricardo Baeza-Yates, editors. *Information Retrieval: Data Structures and Algorithms.* Prentice-Hall, 1992.

[5] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, 1993.

[6] David Hull. Improving text retrieval for the routing problem using latent semantic indexing. In *Proceedings of the 17th ACM/SIGIR Conference*, pages 282–290, 1994.

[7] Parry Husbands, Charles Isbell, and Alan Edelman. MATLAB*P: A tool for interactive supercomputing. In *Proceedings of the 9th SIAM Conference on Parallel Processing for Scientific Computing*, 1999.

[8] K. J. Maschhoff and D. C. Sorensen. A portable implementation of ARPACK for distributed memory parallel computers. In *Preliminary Proceedings of the Copper Mountain Conference on Iterative Methods*, 1996.

[9] Christos Papadimitriou, Prabhakar Raghavan, Hisao Tamaki, and Santosh Vempala. Latent semantic indexing: A probabilistic analysis. In *Proceedings of the 17th ACM Symposium on Principles of Database Systems*, 1998.

[10] Gerald Salton, editor. *The SMART Retrieval System: Experiments in Automatic Document Processing.* Prentice-Hall, 1971.

[11] E. M. Voorhees and D. K. Harman, editors. *The Sixth Text Retrieval Conference.* National Institute of Standards and Technology, August 1998.

[12] Keshang Wu and Horst Simon. TRLAN users guide. Technical Report LBNL-42828, Lawrence Berkeley National Laboratory, 1999.