Term Norm Distribution and its Effects on Latent Semantic Indexing

Parry Husbands*, Horst Simon†, and Chris Ding‡
Lawrence Berkeley National Laboratory
Berkeley, CA 94720, USA

March 4, 2004

Abstract

Latent Semantic Indexing (LSI) uses the Singular Value Decomposition to reduce noisy dimensions and improve the performance of text retrieval systems. Preliminary results have shown modest improvements in retrieval accuracy and recall, but these have mainly explored *small* collections. In this paper we investigate text retrieval on a *larger* document collections (TREC) and focus on distribution of word *norm* (magnitude). Our results indicate the inadequacy of word representations in LSI space on large collections. We emphasize the query expansion interpretation of LSI and propose a LSI term normalization that achieves better performance on larger collections.

Keywords: Information Retrieval, LSI, TREC.

*Email: PJRHusbands@lbl.gov
†Email: HDSimon@lbl.gov
‡Email: CHQDing@lbl.gov

1 Introduction

The use of Latent Semantic Indexing (LSI) has been proposed for text retrieval in several recent works (Deerwester et al., 1990; Dumais, 1991; Hull, 1994; Berry et al., 1995). This technique uses the Singular Value Decomposition (SVD)(Golub and Loan, 1996) to project very high dimensional document and query vectors into a low dimensional space. In this new space it is reasoned that the underlying structure of the collection is revealed thus enhancing retrieval performance. Furthermore, LSI can be alternatively reviewed as a query expansion method (see sections 2.2 and 5), so that *recall* is generally improved. Experiments indicate both improved retrieval precision and recall when LSI is adopted (Deerwester et al., 1990; Dumais, 1991; Hull, 1994; Berry et al., 1995; B.T. Bartell and Belew, 1995; Zha et al., 1998; Ando and Lee, 2001). LSI also improves text categorization (Dumais, 1995; Yang, 1999; Baker and McCallum, 1998) and word sense disambiguation (Schutze, 1998). Theoretical results (B.T. Bartell and Belew, 1995; Papadimitriou et al., 1998; Ding, 1999; Zha et al., 1998) have also provided some understanding on the effectiveness of LSI.

These LSI studies have, however, mostly used relatively small text collections and simplified document models. In this work we investigate the use of the LSI on a larger document collection (TREC). Our initial finding is that on larger text collections, retrieval precision is not enhanced because the LSI mechanism for representing the terms is not sufficient for dealing with the variability in term occurrence. We focus on the norm (the magnitudes) of terms and study the term norm distribution in detail. We propose a term normalization scheme for LSI which improves retrieval precision on the TREC and NPL text collections.

In Section 2 we introduce the text retrieval concepts and LSI necessary for our work. A short description of our experimental setup 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 SVD-based techniques is presented in Section 5 and we conclude in Section 6. A preliminary version of this report appeared in (Husbands et al., 2001).

2 The Vector Space Model and LSI

In text retrieval (see (Frakes and Baeza-Yates, 1992; Salton and Buckley, 1988; Berry et al., 1995) for treatments of some of the issues), a simple way to represent a collection of documents is with a term-document matrix X with

$$X(i,j) = L(i,j) * G(i)$$

where $L(i,j)$ is a local weighting and $G(i)$ is a global weighting depending on term i. The local weight depends on $tf(i,j)$, the number of occurrences of term i in document j. In a very simple weighting scheme, one simply uses $X(i,j) = tf(i,j)$ as the entries of the term-document matrix. However, this scheme is incorrectly dominated by frequent terms.

2.1 Term Weighting

Perhaps the most commonly used term weighting scheme is the `tf.idf` weighting scheme. This scheme uses the standard term frequency $tf(i,j)$ as the local weighting, but weighted by the global inverse document frequency (idf). This scheme is specified by

$$L(i,j) = tf(i,j),\ G(i) = \mathrm{idf(i)} = \log_2(\frac{\mathrm{n}}{\mathrm{df(i)}} + 1) \quad (1)$$

where n is total number of documents, and $\mathrm{df}(i)$ is the document frequency of term i, the number of documents in which term i occurs. This scheme gives very frequent terms low weight and assigns large weight for infrequent (and hopefully more discriminating) terms.

For comparison pursposes we also study the `log.entropy` weighting scheme (Dumais, 1991). In this weighting, the local term weight is the logarithm of the term frequency. The global weighting uses the *entropy* $E(i)$ of term i. This scheme is specified by:

$$L(i,j) = \log_2(tf(i,j)+1),\ G(i) = 1 - E(i),\ E(i) = -\sum_{j=1}^{m} \frac{p_{ij}\log_2(p_{ij})}{\log_2(n)} \quad (2)$$

where $p_{ij} = \frac{tf(i,j)}{\sum_j tf(i,j)}$.

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 normalizing (to 1) the columns of the term-document matrices before computing inner products.

In the discussion to follow we will denote by "term matching" the standard retrieval scheme: for query q, the relevance scores for each document form a vector r and is computed as $r = X^T * q$. For ℓ queries $Q = [q_1, q_2, \cdots, q_\ell]$, the corresponding relevance vectors $R = [r_1, r_2, \cdots, r_\ell]$ are then computed by:

$$R = X^T * Q$$

2.2 LSI and a query expansion interpretation

Latent Semantic Indexing (LSI, (Deerwester et al., 1990; Berry et al., 1995)) 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 X with m terms and n documents.

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

$$X = USV^T$$

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 X is approximated by

$$X \approx X_k \equiv U_k S_k V_k^T$$

where $U_k \equiv [u_1, u_2, \ldots, u_k]$ (the first k columns of U), $V_k = [v_1, v_2, \ldots, v_k]$, and $S_k = diag(s_1, s_2, \ldots, s_k)$ (the upper left k by k part of S). This gives the best rank k approximation to the original matrix. Now the relevance score matrix becomes

$$R = X_k^T Q = (U_k S_k V_k^T)^T Q = (U_k^T X)^T (U_k^T Q) \quad (3)$$

U_k is the projection operator that projects an n-dimensional document or query into k-dimensional LSI subspace. Note that even if the columns of X are normalized to 1, the columns of $X^T U_k$ are not automatically normalized, and so we compute cosines between the projected documents and projected queries.

We may also write $R = X^T(U_k U_k^T q)$ and investigate an alternative view of LSI. Given a term in q, $U_k U_k^T q$ expands this term into many other terms. (If we retain all LSI index vectors, i.e, set $k = m$, by a standard linear algebra theorem, $U_m U_m^T = I_m$, the identity matrix. This implies LSI returns to standard keyword matching, no query expansion.) Thus LSI can be alternatively viewed as a *query expansion* method. This explains why retrieval *recall* is improved in LSI. This query expansion interpretation of LSI by approximate word co-occurence is important for our modification of LSI in section 5.

In the LSI representation the rows of $U_k S_k$ are identified as the "projected terms" and the columns of $U_k^T X = V_k S_k$ are identified as the "projected documents". [1] Note that the "projected documents" are simple linear combinations of the projected terms (see Section 5).

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 (Maschhoff and Sorensen, 1996) (as well as TRLAN (Wu and Simon, 1999) for verification). Another popular software package for computing SVDs is SVDPACK (Berry, 1992).

2.3 Evaluation

In response to a query, a text retrieval system returns an ordered list r of the documents where $r(1)$ is the most relevant, $r(2)$ is the second most relevant, and so on. Here r is obtained by sorting the relevance scores previously defined. 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 r (denoted by $r(1:i)$) 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 $r(1:i)$ 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 interpolated to 0, 0.1, 0.2, ..., 1.0) 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,

[1] There is some disagreement about using U_k^T, $S_k U_k^T$, or $S_k^{-1} U_k^T$ as the "projected terms". In this work we use $S_k U_k$ primarily because the term-term similarity matrix XX^T can be decomposed as US^2U^T if $X = USV^T$. Hence the rows of $U_k S_k$ naturally correspond to the rows of X (see (Ding, 1999)).

early results were encouraging. Figure 1 compares LSI using with $k = 200$ 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.

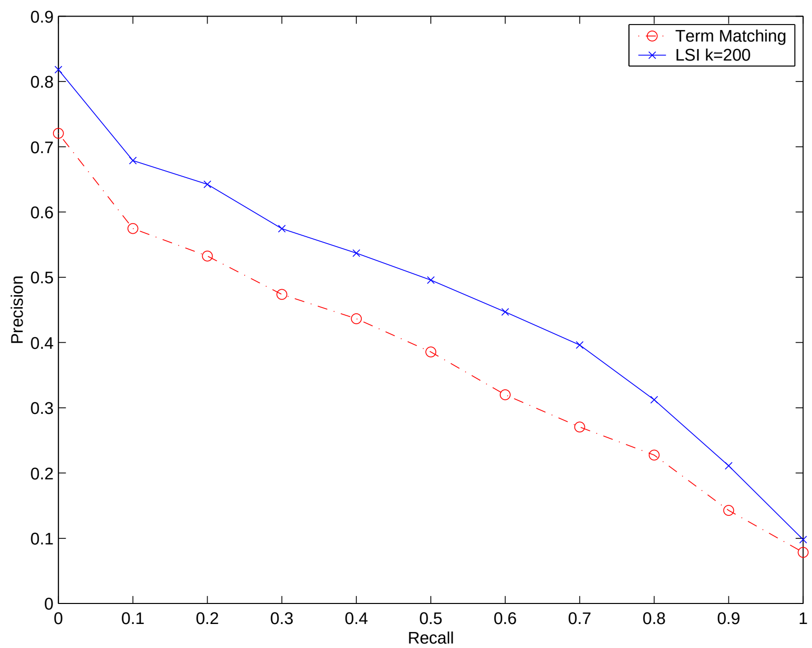

Figure 1: LSI vs. Term Matching on MEDLINE (`tf.idf`)

Performance on very large collections is not as good. Figure 2 shows LSI using $k = 300$ on TREC6 (Voorhees and Harman, 1998), a collection with 115000 terms and 528155 documents. As the field has not entirely settled on the optimal k, 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 performance 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 (Husbands et al., 1999). 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 IBM SP 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,

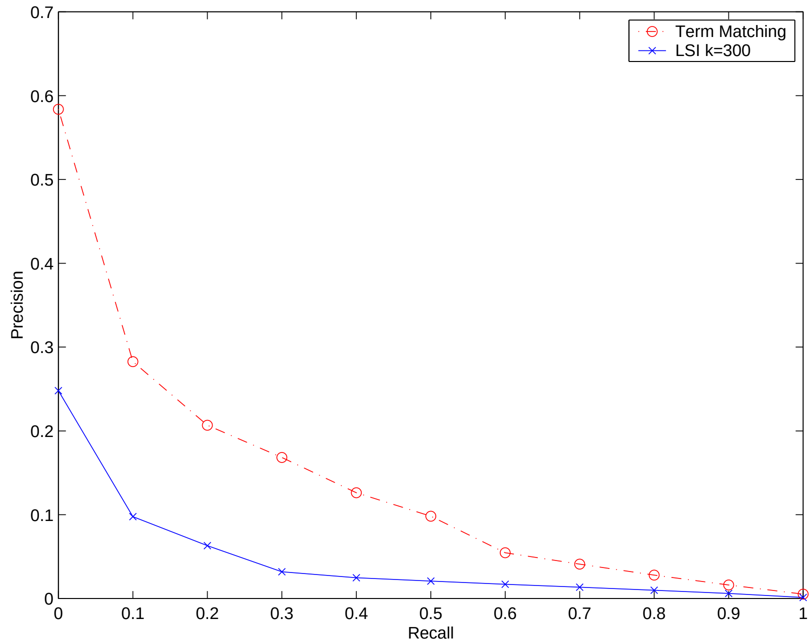

Figure 2: LSI vs. Term Matching on TREC6 (**tf.idf**)

```
[U,S,V]=svds(X,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);
```

Computing and graphing precision/recall curves from pre-judged queries also takes place in MATLAB*P using simple m-file scripts.

4 Term Norms vs. Inverse Document Frequency

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 shows a histogram of term norms for the TREC collection and Figure 4 plots IDF vs. term

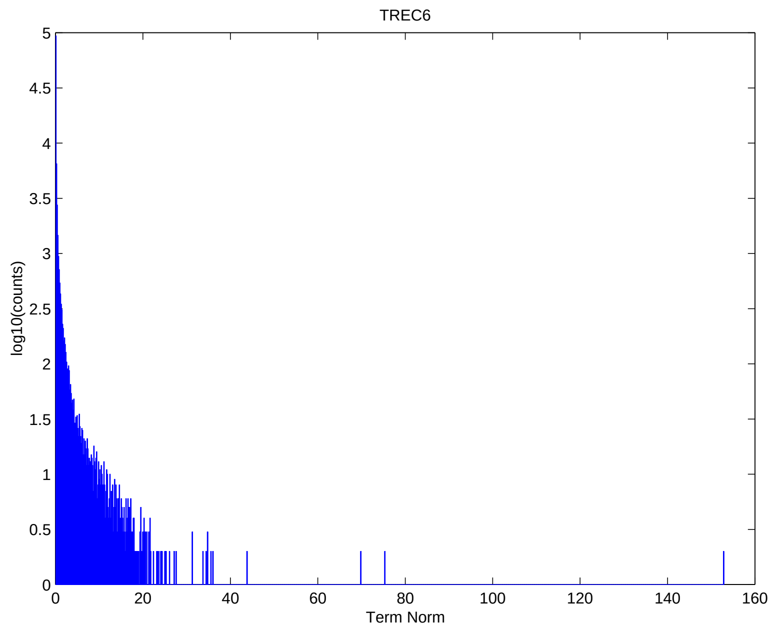

Figure 3: Histogram of TREC6 ($k = 300$) term norms (with `tf.idf` weighting).

norm for the NPL (7491 terms and 11429 documents) and TREC text collections. We see the wide range of norms and 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.

A term with low frequency is often important to discriminate among relevant documents. Therefore, its effect in IR should be magnified, and so its weight in IDF is large. However, our experiments indicate that the correction power of IDF is not sufficient.

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 more specific than "disease" (like "tennis" is more specific than "sports"). The word "polio" 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 happens that the top documents returned for this query are all about disease eradication efforts, but for diseases other than polio (malaria, tuberculosis, AIDS, etc.). It seems that as far as *disease* goes, malaria, tuberculosis, AIDS are more prevalent in the collection whereas polio is relatively less frequent. If the original intent of the query is to find the specific disease *polio*, the small norm of polio is not helpful for this goal.

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

Table 1: Term norms and IDF for text collections

The popular `tf.idf` global weighting scheme appears to be inadequate to mitigate the effect of low term

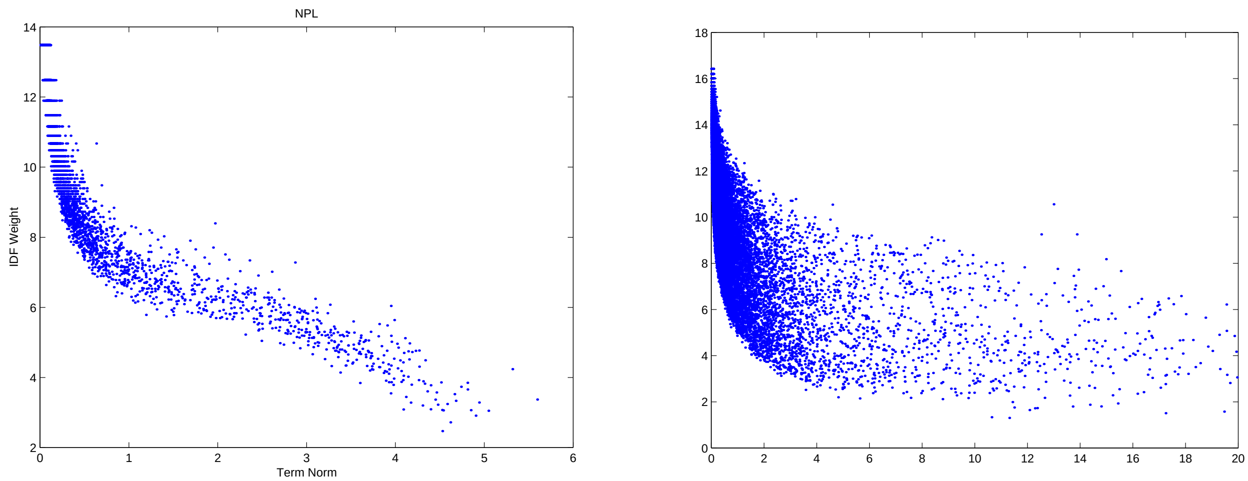

Figure 4: IDF and term norm in LSI-space for the NPL (left) and TREC6 (right) collections (**`tf.idf`**). For TREC6 terms with norm > 20 (42 in total) were are not displayed.

norm. Table 1 shows the range of norms and IDF for a few test collections. The lowest term norms are typically orders of magnitude away from the highest IDF , hinting at IDF's inadequacy. We can therefore say that the effect of IDF is lost after projection. The situation with entropy term weighting is similar. The entropy weights have an even smaller range than the IDF weights and so also are unable to compensate for large variations in term norm.

This situation could be understood in the following way. The occurrence probability of a term is approximately proportional to the document frequency of the term, $P(t) \propto \mathrm{df}(t)$. The norm of t is approximately proportional to $P(t)$. Thus, for two terms t_1, t_2, their norms have a ratio of

$$n(t_1)/n(t_2) \sim \mathrm{df}(t_1)/\mathrm{df}(t_2) \approx 100 - 10000$$

whereas the ratio of their IDFs is approximately

$$log[N/\mathrm{df}(t_1)]/log[N/\mathrm{df}(t_2)] \approx 10.$$

The magnitudes of these two ratios are given in Table 1. They underly the level of importance of the discriminanting power of rare words using norm or IDF. Clearly IDF is not enough to amplify the effect of these rare words.

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 NPL and TREC6 collections. It is interesting to note that after an initial drop the singular values decay very slowly over the displayed range.

5 Normalized LSI

In this section we attempt to remedy the situation by (1) examining how documents are represented in LSI space and (2) proposing a normalization to compensate for infrequent terms.

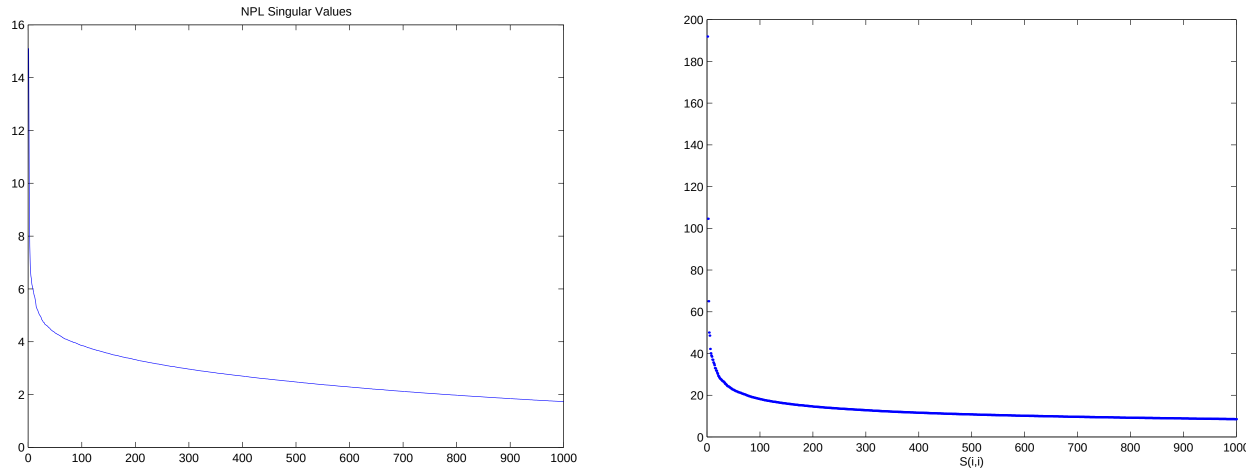

Figure 5: The first 1000 singular values of the NPL (left) and TREC6 (right) collections.

In LSI space, the documents are represented by columns of $U_k^T X$ and terms are represented by rows of $U_k S_k$ (column vectors of $S_k U_k^T \equiv [t_1, \cdots, t_m]$ with t_j the jth term). We can write $U_k^T X$ as $S_k^{-1}(S_k U_k^T)X = S_k^{-1}[t_1, \cdots, t_m]X$. Therefore, each document in LSI space is a linear combination of $\{t_1, \cdots, t_m\}$. For example, the jth document is

$$S_k^{-1} \sum_{i=1}^{m} c_{ij} t_i,$$

$c_{ij} = X(i, j)$.

As discussed in previous sections, `idf` and `entropy` are inadequate in compensating for the effects of infrequent terms in $(t_1, \cdots, t_m)$. Motivated by this we therefore propose to normalize $(t_1, \cdots, t_m)$,[2] i.e., rows of $U_k S_k$ as

$$T_k = \text{row-normalized}(U_k S_k) = D^{-1}(U_k S_k). \tag{4}$$

where $D = diag(d_1, \ldots, d_k)$ is a diagonal matrix, d_i = norm of ith row of $U_k S_k$. In the normalized LSI, we compute the relevance scores using

$$R = (T_k^T X)^T (T_k^T Q), \tag{5}$$

instead of using Eq.3 in standard LSI,

The normalized LSI can be alternatively viewed as an improvement to the query expansion interpretation of LSI (see section 2.2). If we use cosine similarity between words (rows of $U_k S_k$) in LSI, the word-word co-occurrence matrix is

$$C = D^{-1}(U_k S_k)(U_k S_k)^T D^{-1} = T_k T_k^T.$$

Consider the query vector q. The expanded query is $q_{\text{expanded}} = Cq = T_k T_k^T q$. If q contains a term t_i, this term is now expanded to all *similar* terms through $TT^T q$. If term t_i has a small norm, its expanded weight in the query will be almost negligible. In Normalized LSI, we in effect normalize each term in LSI space to 1 so that terms with small norm are promoted to be equals.

[2]Throughout this paper we use the L_2 norm. Given a vector v, $||v||_2 = \sqrt{\sum_{i=0}^{n} v_i^2}$. Normalizing v to 1 is achieved by multiplication by the constant $\frac{1}{||v||_2}$.

The complete Normalized LSI is described below:

- Compute the SVD with k factors U_k, S_k, V_k
- Normalize the rows of the projected terms $T = D^{-1}U_kS_k$.
- Compute the relevance score using Eq.(5) (in practice to ensure accuracy, we compute cosine between rows of X^TT and columns of T^TQ.)

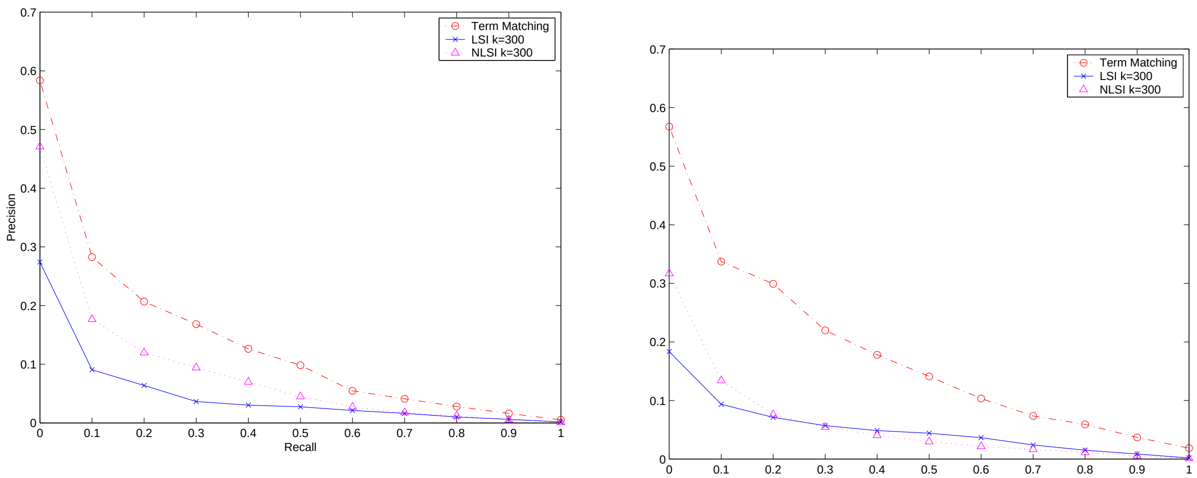

Figure 6: Precision-recall curves on TREC6 collection for term matching, LSI, and NLSI. We use **tf.idf** (left panel) and **log.entropy** (right panel) term weighting schemes.

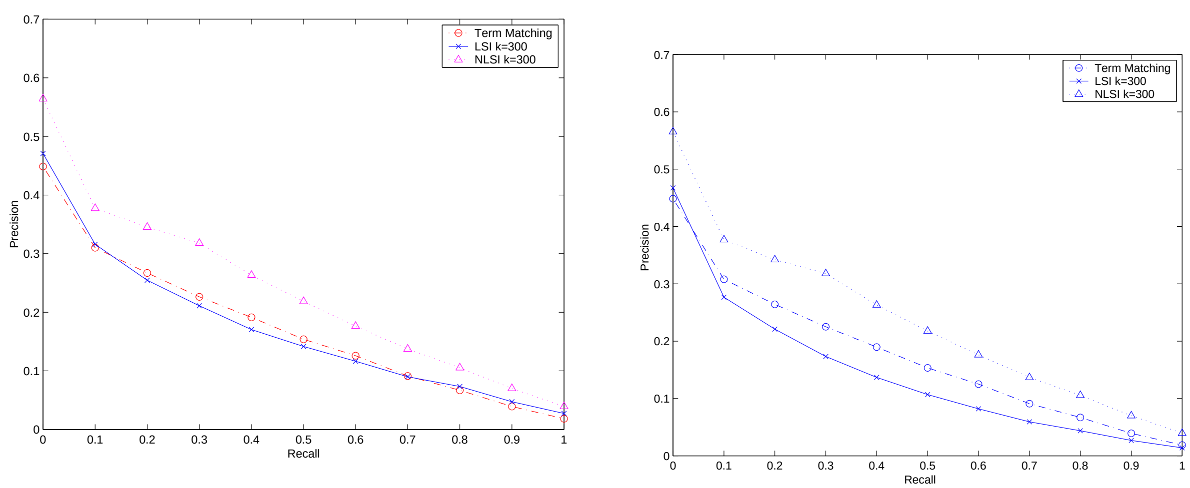

Figure 7: Precision-recall curves on NPL datatset for term matching, LSI, and NLSI. We use **tf.idf** (left) and **log.entropy** (right) term weighting schemes.

The results of using NLSI on the TREC6 are shown in Figure 6 both **tf.idf** and **log.entropy** weighting schemes. NLSI results on NPL are shown in Figure 7. Re-scaling the projected terms has a positive effect on

LSI performance for the NPL and TREC6 collections. For NPL, NLSI outperform both term matching and LSI. For TREC6 NLSI improves upon standard LSI, but still fall short of term matching. We also applied NLSI on MED, where NLSI performance is not as good as LSI.

6 Conclusions

LSI projects 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 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, we also see that in LSI representation of words, rare terms (with low norm) are not adequately weighted, sometimes resulting in poor retrieval performance. We proposed a LSI normalization scheme based on query expansion interpretation of LSI. The normalized LSI partially compensated this inadequacy and resulting in better retrieval precision on NPL and TREC.

Acknowledgements. This work is supported by the U.S. Department of Energy (Office of Science, Office of Laboratory Policy and Infrastructure Management, and Office of Advanced Scientific Computing Research, Division of Mathematical, Information, and Computational Sciences) under Contract No. DE-AC03-76SF00098.

References

Ando, R. and Lee, L. (2001). Iterative residual rescaling: An analysis and generalization of LSI. *Proc. ACM Conf. on Research and Develop. IR(SIGIR)*, pages 154–162.

Baker, L. and McCallum, A. (1998). Distributional clustering of words for text classification. *Proc. ACM Conf. on Research and Develop. Info. Retrieval (SIGIR)*.

Berry, M., Dumais, S., and O'Brien, G. W. (1995). Using linear algebra for intelligent information retrieval. *SIAM Review*, 37:573–595.

Berry, M. W. (1992). Large scale singular value computations. *Int'l J. Supercomputer Applications*, pages 13–49.

B.T. Bartell, G. C. and Belew, R. (1995). Representing documents using an explicit model of their similarities. *J.Amer.Soc.Info.Sci,* **46***, 251-271, 1995*, pages 251–271.

Deerwester, S., Dumais, S., Landauer, T., Furnas, G., and Harshman, R. (1990). Indexing by latent semantic analysis. *J. Amer. Soc. Info. Sci*, 41:391–407.

Ding, C. (1999). A similarity-based probability model for latent semantic indexing. *Proc. 22nd ACM SIGIR Conference*, pages 59–65.

Dumais, S. (1991). Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments and Computers*, 23:229–236.

Dumais, S. (1995). Using lsi for information filtering: Trec-3 experiments. *Third Text REtrieval Conference (TREC3), D Harman, Ed, National Institute of Standards and Technology Special Publication.*

Frakes, W. B. and Baeza-Yates, R., editors (1992). *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall.

Golub, G. and Loan, C. V. (1996). *Matrix Computations, 3rd edition.* Johns Hopkins, Baltimore.

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

Husbands, P., Isbell, C., and Edelman, A. (1999). MATLAB*P: A tool for interactive supercomputing. In *Proceedings of the 9th SIAM Conference on Parallel Processing for Scientific Computing.*

Husbands, P., Simon, H., and Ding, C. (2001). On the use of singular value decomposition for text retrieval. *Proc. of SIAM Workshop on Comp. Info. Retrieval.*

Maschhoff, K. J. and Sorensen, D. C. (1996). A portable implementation of ARPACK for distributed memory parallel computers. In *Proc. Copper Mountain Conf. on Iterative Methods.*

Papadimitriou, C., Raghavan, P., Tamaki, H., and Vempala, S. (1998). Latent semantic indexing: A probabilistic analysis. In *Proceedings of the 17th ACM Symposium on Principles of Database Systems.*

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5).

Schutze, H. (1998). Automatic word sense discrimination. *Computational Linguistics*, pages 97–124.

Voorhees, E. M. and Harman, D. K., editors (1998). *The Sixth Text Retrieval Conference.* National Institute of Standards and Technology.

Wu, K. and Simon, H. (1999). TRLAN users guide. Technical Report LBNL-42828, Lawrence Berkeley National Laboratory.

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *J. Information Retrieval*, 1:67–88.

Zha, H., Marques, O., and Simon, H. (1998). A subspace-based model for information retrieval with applications in latent semantic indexing. *Proc. Irregular '98, Lecture Notes in Computer Science, Vol. 1457. pp.29-42.*