LATENT SEMANTIC INDEXING USING EIGENVALUE ANALYSIS FOR EFFICIENT INFORMATION RETRIEVAL

CHERUKURI ASWANI KUMAR[*], SURIPEDDI SRINIVAS[**]

[*] School of Computing Sciences
Vellore Institute of Technology
Deemed University, Vellore – 632014, India
e-mail: aswanis@gmail.com

[**] School of Science and Humanities
Vellore Institute of Technology
Deemed University, Vellore – 632014, India
e-mail: srinusuripeddi@hotmail.com

Text retrieval using Latent Semantic Indexing (LSI) with truncated Singular Value Decomposition (SVD) has been intensively studied in recent years. However, the expensive complexity involved in computing truncated SVD constitutes a major drawback of the LSI method. In this paper, we demonstrate how matrix rank approximation can influence the effectiveness of information retrieval systems. Besides, we present an implementation of the LSI method based on an eigenvalue analysis for rank approximation without computing truncated SVD, along with its computational details. Significant improvements in computational time while maintaining retrieval accuracy are observed over the tested document collections.

Keywords: information retrieval, latent semantic indexing, eigenvalues, rank reduction, singular value decomposition, vector space method

1. Introduction

Several studies have reported that Latent Semantic Indexing (LSI) based on truncated Singular Value Decomposition (SVD) could be compared favorably with other Information Retrieval (IR) techniques in terms of retrieval accuracy (April and Pottenger, 2006; Berry *et al.*, 1995; 1999; Gao and Zhang, 2005; Parry Husbands *et al.*, 2001; Yun Qing Ye, 2000). LSI became famous as one of the first IR techniques exhibiting effectiveness in dealing with the problems of synonymy and polysemy. The basic idea of LSI is that if two document vectors represent the same topic, they will share many associated words with a keyword and they will have very close semantic structures after dimension reduction via truncated SVD (Deerwester, 1990; Landauer *et al.*, 1998). However, for large-scale datasets, the computing and storage costs associated with truncated SVD representation may be prohibitive. Recent studies also indicate that retrieval accuracy of the truncated SVD technique can deteriorate if the document sets are large (Balinski and Damilowicz, 2005; Berry and Shakhina, 2005). Several strategies have been proposed to deal with LSI on large datasets. The sparsification strategy was used to remove less important entries in truncated SVD matrices (Gao and Zhang, 2003). Clustered and distributed SVD strategies were proposed to partition large datasets (Bass and Behrems, 2003; Tang, 2003). Unfortunately, the investigations of both clustered and distributed SVD strategies are incomplete. Fan *et al.* (1999) examined a random sampling based approach to SVD approximation and presented their results. Even though their algorithm performed well in reducing computational time, the retrieval quality was not similar to that of truncated SVD. Yingbo Hua (2000) proposed an alternative power method for rank reduction without computing truncated SVD. The alternate power method is an iterative algorithm that is globally and exponentially convergent under a weak condition.

Numerous researchers in LSI have devoted a lot of time to testing the effectiveness of SVD in solving the problems of synonymy and polysemy (Berry, 1992; Berry *et al.*, 1999; Husbands and Ding, 2005). Empirical results show a general increase in the retrieval quality, but to the best of our knowledge, such algorithms come with no guarantee regarding the quality of the approximation produced. Most of the rank reductions achieved via truncated SVD concern some properly chosen characteristic matrices. This fact led to a common practice where the computation of SVD is first carried out before rank reduction is accomplished.

In this paper, we present an implementation of matrix rank approximation using an eigenvalue analysis for LSI. This methodology was initially introduced and discussed by Praks *et al.* (2003) for image retrieval in a hard industrial environment. We expand upon this work, showing that LSI can exploit the eigenvalues and eigenvectors produced from term-by-document matrices, thereby decreasing computational time and preserving retrieval accuracy. In this paper, we demonstrate the application of this methodology to LSI based text retrieval and present experimental results using this method. We compare this method with the truncated SVD approach of rank approximation and discuss the experimental results using standard testing document collections. We show that in the case of LSI, we can achieve similar optimal approximations of truncated SVD using considerably fewer computations by the eigenvalue and eigenvector based method. This work is also aimed at studying the effect of rank approximation on IR and performance improvement in using LSI over the traditional VSM. Our paper is structured as follows: Section 2 presents the vector space method. In Section 3, we discuss LSI and SVD. Section 4 presents the methodology for matrix rank approximation using eigenvalues in the LSI context for text retrieval. Section 5 presents experimental results regarding the effect of rank approximation on IR, the superiority of LSI over the VSM, the numerical aspects and computational details of the eigenvalue methodology of rank approximation in comparison with the truncated SVD method, followed by conclusions and references.

2. Vector Space Method (VSM)

A vector-based information retrieval method represents both documents and queries with high-dimensional vectors, while computing their similarities by the vector inner product. When the vectors are normalized to unit lengths, the inner product measures the cosine of the angle between the two vectors in the vector space (Yates and Neto, 1999). In the VSM, each document $\vec{d_j}$ is represented by a weight vector $\vec{d_j} = (w_{1j}, w_{2j}, \dots, w_{tj})^T$, where w_{zj} is the weight or importance of the term z in the representation of the document $\vec{d_j}$, and t is the size of the indexing term set. A collection of n documents is then represented by a term-by-document matrix with t rows and n columns.

In the VSM, two main components of the term weight are used to represent the elements of the term-by-document matrix, the frequency (TF) of occurrences of each word in a particular document and the inverse document frequency (IDF) of each word, which varies inversely with the number of documents to which a word is assigned. So, the weight of a term i in a document j is given by the following equation:

$$w_{i,j} = \mathrm{tf}_{i,j}\mathrm{idf}_i = \mathrm{tf}_{i,j} \log\left(\frac{N}{\mathrm{df}_i}\right), \tag{1}$$

where $\mathrm{tf}_{i,j}$ is the frequency of the i-th term in the j-th document, df_i is the number of documents in which the term i appears at least once, and N is the number of documents in the collection. This method assigns the highest weight to those terms which appear frequently in a small number of documents in the documents set. For queries, also the same vector representation is given as $\vec{q_i} = (q_{1i}, q_{2i}, \dots, q_{ti})^T$, where q_{zi} is the weight of term z in the representation of the query $\vec{q_i}$. We measure the similarity between a document and a query where both are normalized to unit lengths in the underlying vector space (Yates and Neto, 1999),

$$\mathrm{Sim}_{\mathrm{VSM}}(\vec{q_i}, \vec{d_j}) = \frac{\sum\limits_{z=1}^{t} (q_{zi} w_{zj})}{\sqrt{\sum\limits_{z=1}^{t} q_{zi}^2}\sqrt{\sum\limits_{z=1}^{t} w_{zj}^2}}. \tag{2}$$

The advantages of this approach are adaptability, robustness and minimal user intervention.

3. Latent Semantic Indexing (LSI) and Singular Value Decomposition (SVD)

LSI is a variant of the VSM which maps a high dimensional space into a low dimensional space. LSI replaces the original matrix by another matrix whose column space is only a subspace of the column space of the original matrix. In the VSM, since every word does not appear in each document, the document matrix is usually of a high dimension and sparse. High dimensional and sparse matrices are susceptible to noise and have difficulty in capturing the underlying semantic structure. Additionally, the storage and processing of such data place great demands on computing resources (Berry *et al.*, 1999; Deerwester, 1990; Kontostathis and Pottenger, 2002a).

Reduction in model dimensionality is one way to address this problem (Park and Elden, 2003; Schislen, 2004). SVD takes advantage of the implicit higher order structure in the association of terms within documents by largest singular vectors. The vectors representing the documents and queries are projected onto a new, low-dimensional space obtained by truncated SVD (April and Pottenger, 2006; Kontostathis and Pottenger, 2002b; Landauer *et al.*, 1998).

The dimensionality reduction is accomplished by approximating the original term-by-document A with a new matrix A_k, where $\mathrm{rank}(A) = r > \mathrm{rank}(A_k) = k$. In SVD, a large term-by-document matrix is decomposed into a set of orthogonal factors from which the original term-by-document matrix can be approximated by a linear combination. Vectors of factor weights represent documents. The SVD of a matrix A is written as

$$A = U\Sigma V^T. \tag{3}$$

If the term-by-document matrix A is $t \times d$, and then U is a $t \times d$ orthogonal matrix, V is a $d \times d$ orthogonal matrix, and Σ is a $d \times d$ diagonal matrix, where the values on the diagonal of Σ are called the singular values. The singular values can then be sorted by magnitude and the top k values are selected as a means of developing a 'latent semantic' representation of the original matrix. The geometric interpretation of SVD is to regard the rows of V, i.e., the columns of V^T as defining the new axes, the rows of U as coordinates of the objects in the space spanned by these new axes, and Σ as a scaling factor indicating the relative importance of each new axis. By changing all but the top k rows of Σ to zero rows, low rank approximation to A, called A_k, can be created through the truncated SVD as

$$A_k = U_k \Sigma_k V_k^T, \tag{4}$$

where U_k is the $t \times k$ term-by-concept matrix, Σ_k is the $k \times k$ concept-by-concept matrix; V_k is the $k \times d$ concept-by-document matrix (AswaniKumar *et al.*, 2005; Berry and Shakhina, 2005; Landauer *et al.*, 1998). Only the first k columns are kept in U_k and only the first k columns are recorded in V_k^T. Each row of U_k is a k-dimensional vector representing a term in the original collection. To each of the k reduced dimensions there is associated a latent concept which may not have any explicit semantic content, yet helps to discriminate documents. By analogy to the VSM, the vector representation of a document is a weighted sum of vector representations of its constituent terms. That is, for an original document vector e_j in A, it can be represented in the reduced dimension vector space as

$$U_k^T e_j = \Sigma_k V_k^T e_j. \tag{5}$$

The rank of A has been lowered from r to k. A key property of this reduced rank approximation is that it achieves the best possible approximation with respect to the Frobenius norm. This low rank approximation removes redundancy from the original data and allows us to uncover latent semantic relations among terms as well as documents. A query can be considered as just another document. Specifically, the $m \times 1$ user query vector q is located at the weighted sum of its component term vectors in the k-space. Queries are formed into pseudo-documents that specify the location of a query in the reduced term-document space (Bast and Weber, 2005). Given q, a vector whose non-zero elements contain the weighted term-frequency counts of the terms that appear in the query, the pseudo-document, $\hat{q}$, can be represented by

$$\hat{q} = q^T U_k \Sigma_k^{-1}. \tag{6}$$

Thus, the pseudo-document consists of the sum of the term vectors ($q^T\ U_k$) corresponding to the terms specified in the query scaled by the inverse of the singular values (Σ_k^{-1}). The singular values are used to individually weigh each dimension of the term-document space. Once the query is projected onto the reduced term-document space, one of several similarity measures can be applied to compare the position of the pseudo-document. Documents are ranked according to the results of this similarity measure, and the highest ranked documents are returned to the user (Berry *et al.*, 1999; Husbands *et al.*, 2001; Ye, 2000). To date, several theoretical explanations and results have appeared in the literature and these studies have provided a better understanding of LSI (April and Pottenger, 2005; Cheng and Lafferty, 2006; Deerwester, 1990; Landauer *et al.*, 1998).

Unfortunately, the SVD decomposition of a document matrix is a memory and time-consuming operation, especially for large data collections, where the term-by-document matrix becomes large. A primary focus of this paper is to address this problem.

4. LSI based on Eigenvalue Analysis

The SVD algorithm is $O(n^2k^3)$, where n is the number of terms plus documents and k is the number of dimensions in the concept space. However, n grows rapidly as the number of terms and documents increases. This makes the SVD algorithm unfeasible for large document collections. Consequently, the bulk of LSI processing time can be spent on computing the truncated SVD of large sparse term-by-document matrices, especially when several new terms or documents are to be added to the database (Zang *et al.*, 2002).

Hence there is a need for alternative solutions for truncated SVD in LSI. The focus of this work is to demonstrate that an alternate eigenvalue method can be used for LSI based IR at a reduced computational cost compared with truncated SVD. The concept-by-document matrix (V^T) is orthogonal and it holds the matrix identity. Similarly, the concept-by-concept matrix (Σ) is diagonal and contains strictly positive integer values. There exists a relationship between the SVD of a matrix A and the symmetric eigenproblem of symmetric square matrices A^TA (Golub and Van Loan, 1996; Praks *et al.*, 2003). Computing the SVD of an $m \times n$ matrix A is basically computing the eigenvalues and eigenvectors of symmetric matrices AA^T and A^TA. The correlation matrices AA^T (for terms) and A^TA (for documents) provide information about the relationships in the document collection.

Spectral methods are usually based on some variant of the Lanczos algorithm (Berry, 1992). For computing k eigenvectors which give k concepts, the bulk of the running time of Lanczos and related algorithms is spent on computing $O(k)$ matrix-vector products. This gives a total running time of $O(nk)$, where n is the number of nonzero entries in the given matrix, which is typically sparse. Following these observations, a methodology for LSI without computing SVD can be formulated as shown in Fig. 1.

1. Read the original term-by-document matrix (A) and the query matrix.
2. Calculate the size (numbers of rows and columns) of the term-by-document matrix.
3. Read the value (k) to which the matrix is to be approximated.
4. Compute k largest eigenvalues and eigenvectors of $A^T A$ to obtain the document-by-concept matrix (V_K) containing the eigenvectors spanning the concept-by-concept space (Σ^2) containing the eigenvalues.
5. Compute the concept-by-concept matrix (Σ_K) to get singular values.
6. Compute the term-by-concept matrix (U_K) from the original term-by-document, concept-by-document and concept-by-concept matrices using the formula $U_K = AV_K(\Sigma_K)^{-1}$.
7. Compute the coordinate (pseudo-document) of the query vector as explained in Section 3.
8. Calculate the similarity coefficients between the coordinates of the query vector and documents.
9. Rank the documents based on their similarity measures.

Fig. 1. Methodology 1 for LSI using eigenvalues.

1. Do Steps 1–5, as explained in Fig. 1, to compute the document-by-concept matrix (V_K) and the concept-by-concept matrix (Σ_K).
2. Compute the co-ordinate (pseudo-document) of the query vector using V_K, Σ_K and term-by-document matrices without computing the term-by-concept matrix (U_K) with the formula $\hat{q} = (((q^T\ A)V_K)\Sigma_k^{-1})\Sigma_k^{-1}$.
3. Calculate the similarity coefficients between the coordinates of the query vector and documents.
4. Rank the documents based on their similarity measures.

Fig. 2. Methodology 2 for LSI using eigenvalues.

Analyzing the methodology given in Fig. 1, we can observe that the matrix U_K does not need to be explicitly computed and stored in memory. This observation brings additional accelerating of speed and decreasing the memory requirements of LSI. This can be formulated as shown in Fig. 2.

5. Experimental Results

In this section, we present the details of the experiments we conducted on Medline, Cranfield, CACM, CISI and 20 News groups document collections and their results. These collections were chosen because they have query and relevance judgment sets that are already available. All the programs used for the experiments were written in MATLAB.

5.1. Effect of Rank Approximation on Information Retrieval. In order to present the effect of matrix rank reduction on IR, we conducted experiments on document collections Medline and Cranfield. The Medline document collection contains a total of 1033 documents indexed by 5735 terms. So it forms a term-by-document matrix of size 5735×1033 with rank 1033. The Cranfield data collection contains 1398 documents indexed by 4563 terms. Hence it forms a term-by-document matrix of size 4563×1398 with rank 1398. We considered 17 queries from Medline and 30 queries from the Cranfield data collection. The elements of the term-by-document matrices were weighted using the TF × IDF method explained in Section 2. We explored the possible values for ranks from $k = 100$, incremented by 100, up to 1000. As the rank of the term-by-document matrix was increased, the size of the approximated term-by-document matrix also increased proportionally. At rank 100, the approximated concept matrix of the Cranfield collection was of size 4.62 M.B. For low rank approximation 1000, it occupied the size of 53.1 M.B. Similarly, at rank 100, the approximated concept matrix of the Medline collection occupied the size of 5.23 M.B and, as the rank was increased, the size of the approximated matrix also increased. For rank-1000 approximation, the collection occupied a 59.2 MB disk space.

The retrieval quality of LSI heavily depends on its number of dimensions. We need an optimal rank that captures the underlying semantic nature of the data. If we truncate the rank further from an optimal rank, it will lose important factors, and if we keep a higher rank, it will result in modeling the unnecessary noise and lead to a poor performance by regenerating the original data. The general metrics used for testing the accuracy of the IR are precision and recall. Precision is the ratio of the number of relevant documents retrieved to the total number of documents retrieved. Recall is the ratio of the number of relevant documents retrieved to the total number of relevant documents in the collection. Among all, rank 300 yielded relatively better interpolated precision value for the Cranfield collection. Ranks 100 performed fairly well yielding better interpolated precision results for the Medline collection (AswaniKumar and Srinivas, 2006).

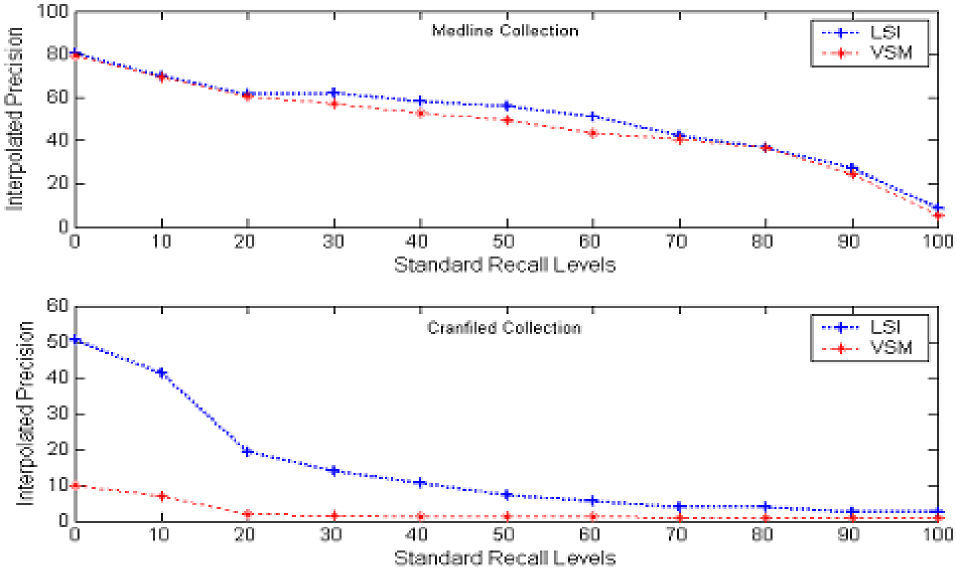

Fig. 3. Comparison of the LSI and VSM methods for the Medline document collection.

5.2. LSI vs. the VSM. In order to compare the effectiveness of the vector space method and latent semantic indexing, we conducted experiments on the Cranfield and Medline document collections. For LSI, we considered ranks 300 and 100 for the Cranfield and Medline document collections, respectively. Figure 3 presents the comparison of the VSM and LSI methods on the Medline and Cranfield document collections. LSI clearly exhibits its superiority over the VSM. There is a nearly 70% improvement in retrieval results with LSI.

The difference in performance between LSI and the VSM is especially impressive at lower recall levels, where interpolated precision is very good. But at higher recall levels, LSI yielded only a few more relevant documents from the collection than the VSM. The VSM and LSI produced similar results, as Medline is a specialized collection. But there is a marginal superiority of LSI over the VSM exhibited at higher recall levels. It is clear that LSI offers better results when compared with the traditional term matching VSM. What distinguishes LSI model from the VSM and gives LSI power in dealing with synonymy and polysemy is the refinement of the original semantic space of the VSM using the truncated SVD of the term-by-document matrix.

5.3. Rank Approximation Using SVD and Eigenvalue Analysis. To study the application of eigenvalue analysis for rank approximation in the context of LSI, we performed a set of retrieval experiments on the Cranfield, Medline, CACM, CISI and 20 Newsgroups testing document collections. The document database in the CACM test collection consists of all the 3204 articles published in *Communications of ACM* from 1958 to 1979. From this, we extracted 5763 terms, creating a term-by-document of size 5763×3204. 1460 documents in the CISI test collection were selected from the Institute of Scientific Information. This collection is indexed by 5544 terms. So we created a term-by-document of size 5544×1460. The 20 Newsgroups dataset is a collection of approximately 20,000 documents, partitioned evenly across 20 different news groups. From that, we considered a computer science dataset which is partitioned into 3 subgroups as IBM PC hardware, MAC hardware and MS-Windows datasets. It contained 2437 documents. After preprocessing, we prepared a term-by-document matrix of size 8258×2437. We evaluated the presented methodologies in terms of retrieval efficiency and accuracy, and compared it with the truncated SVD based method.

5.3.1. Computational Time. We first performed rank approximation using truncated SVD approximation and then, using eigenvalue based rank approximation, we approximated the collections to ranks ranging from 100 to 1000. For both the methods, we compared their computational timings (CPU time) in minutes on an IBM Think Pad running at 1.5 GHz with a 512 Mb memory using the MATLAB 6.5 environment.

Truncated SVD took 3.62 minutes to approximate the Cranfield collection to rank 300, whereas the first eigenvalue based method took only 0.94 minutes. But the second eigenvalue method took only 0.88 minutes. Truncated SVD took 0.59 minutes to approximate the Medline collection to rank 100, whereas the first eigenvalue methodology approximated the collection within 0.3 minutes and the second methodology took only 0.27 minutes. The difference between the computational times is big at higher

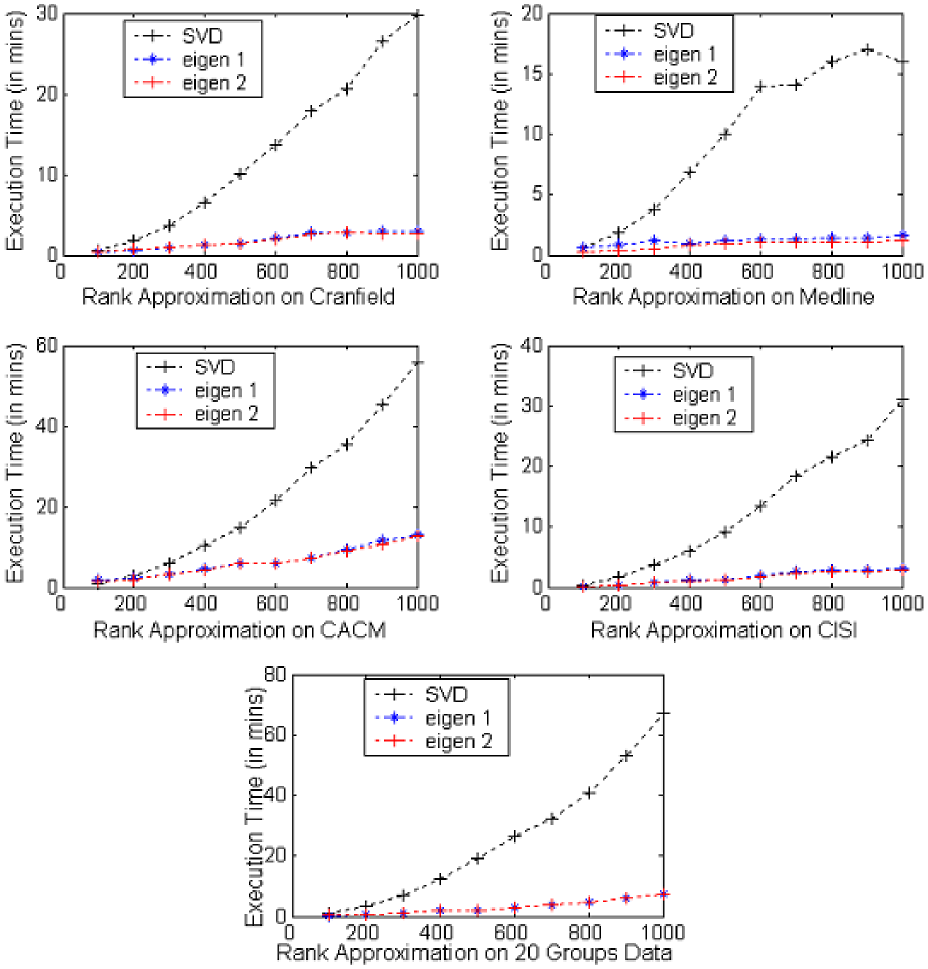

Fig. 4. Rank approximation using SVD and eigenvalue analysis.

rank approximations. For approximating the Medline collection to rank 1000, truncated SVD took 18 minutes. But the same was achieved using the first eigenvalue method within only 1.61 minutes and the second methodology within 1.14 minutes.

For the CACM document collection, truncated SVD took 10.42 minutes to approximate the collection to a rank of 300, whereas the first eigenvalue methodology took only 3.07 minutes and the second methodology took 3.06 minutes. For the CISI collection, truncated SVD took 3.62 minutes whereas the first eigenvalue methodology took only 0.75 minutes and the second eigenvalue methodology took 0.71 minutes. We focused on rank 300 as an optimal rank for this document collection. Truncated SVD took 6.78 minutes to approximate 20 groups in the computer science collection to a rank of 300. But the first eigenvalue methodology took only 1.2 minutes and the second methodology took 1 minute. These empirical results show that the eigenvalue-based method has substantially lower computational time than truncated SVD. Both methodologies based on eigenvalues approximated the collections in almost the same amount of time. Only for higher rank approximations (above 600), the second eigenvalue methodology slightly outperformed the first one. All these details are illustrated in Fig. 4. The most time consuming part of these methodologies is the computation of the correlation matrix $A^T A$. For all these experiments it took nearly 90% of computational time.

5.3.2. Retrieval Accuracy. We compared the retrieval accuracy in terms of precision and recall using the eigenvalue methods with that of SVD. The eigenvalue methods offered interpolated precision values at all standard recall levels exactly similar to that of truncated SVD. Consequently, we could not visualize the results through a graph, as there was a clear overlap of the singular values generated from truncated SVD and eigenvalue methods. But for illustration, in Figs. 5 and 6 we provide the singular values produced in the concept space using the truncated SVD and eigenvalue methods on the Cranfield and Medline collections, respectively.

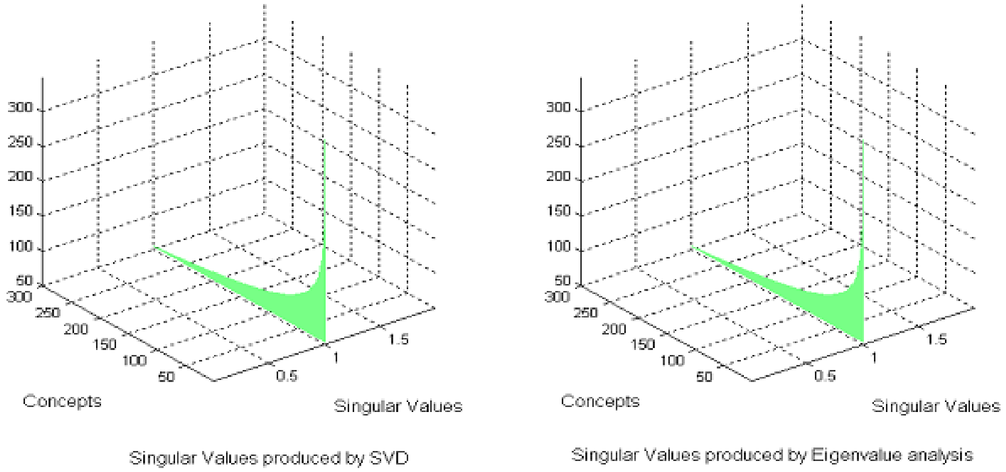

Fig. 5. Singular values produced from the Cranfield collection.

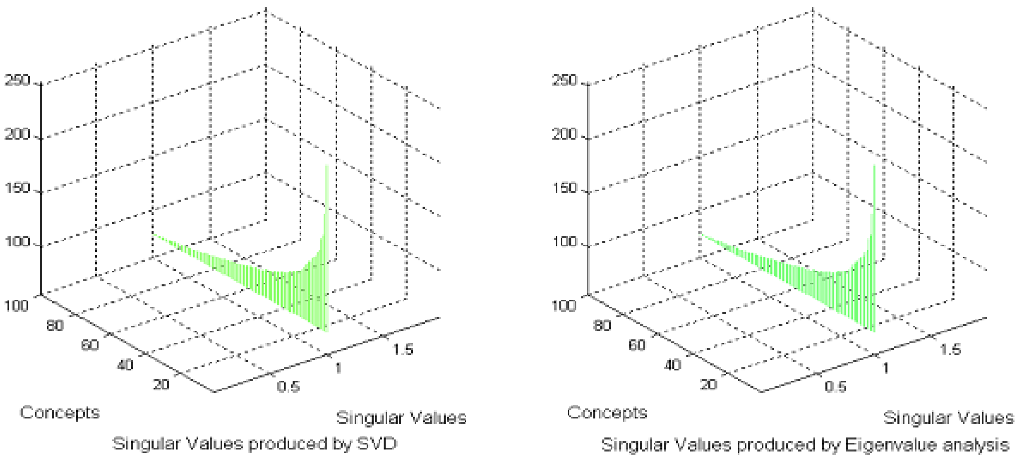

Fig. 6. Singular values produced from the Medline collection.

6. Conclusions

In this paper, an analysis and understanding of how matrix rank reduction would help to derive the latent semantic nature of information thereby improving the efficiency of IR was provided. We conducted experiments on two standard document collections and observed that the performance of LSI with matrix rank reduction is superior to the traditional vector space method. In LSI, the effects of rank on the size and retrieval quality of both collections were analyzed. We presented two methodologies for implementing LSI without truncated SVD based on the eigenvalues. We evaluated the eigenvalue method in terms of retrieval accuracy and efficiency and compared it with a truncated SVD based method. A key observation is that LSI using eigenvalues has lower computational time than truncated SVD and a similar retrieval accuracy as SVD.

Acknowledgments

This work has been supported by the Department of Science and Technology, Govt. of India, under the grant no. SR/S3/EECE/25/2005.

References

April K. and Pottenger W.M. (2006): *A framework for understanding latent semantic indexing performance*. — J. Inf. Process. Manag., Vol. 42, No. 1, pp. 56–73.

AswaniKumar Ch., Gupta A., Batool M. and Trehan S. (2005): *An information retrieval model based on latent semantic indexing with intelligent preprocessing*. — J. Inf. Knowl. Manag., Vol. 4, No. 4, pp. 1–7.

AswaniKumar Ch. and Srinivas S. (2006): *On the effect of rank approximation on information retrieval*. — Proc. Int. Conf.

Systemics Cybernetics and Informatics, Hyderabad, India, pp. 876–880.

Balinski J. and Danilowicz C. (2005): *Ranking method based on inter document distances*. — Inf. Process. Manag., Vol. 41, No. 4, pp. 759–775.

Bass D. and Behrens C. (2003): *Distributed LSI: Scalable concept based information retrieval with high semantic resolution*. — Proc. 2003 Text Mining Workshop, San Francisco, CA, USA, pp. 72–82.

Bast H. and Weber I. (2005): *Insights from viewing ranked retrieval as rank aggregation*. — Proc. Workshop *Challenges in Web Information Retrieval and Integration*, WIRI05, Tokyo, Japan, pp. 232–239.

Berry M.W. (1992): *Large scale singular value computations*. — Int. J. Super Comput. Appli., Vol. 6, No. 1, pp. 13–49.

Berry M.W. and Dumasis S.T. (1995): *Using linear algebra for intelligent information retrieval*. — SIAM Rev., Vol. 37, No. 4, pp. 573–995.

Berry M.W., Drmac Z. and Jessup E.R. (1999): *Matrices, vector spaces, and information retrieval*. — SIAM Rev., Vol. 41, No. 2, pp. 335–362.

Berry M.W. and Shakhina A.P. (2005): *Computing sparse reduced-rank approximation to sparse matrices*. — ACM Trans. Math. Software, 2005, Vol. 31, No. 2, pp. 252–269.

Cheng X.Z. and Lafferty J.(2006): *A risk minimization framework for information retrieval*. — Inf. Process. Manag., Vol. 42, No. 1, pp. 31–55.

Deerwester S. (1990): *Indexing by latent semantic analysis*. — J. Ameri. Soci. Inf. Sci., Vol. 41, No. 6, pp. 391–407.

Fan J., Ravi K., Littman M.L. and Santosh V. (1999): *Efficient singular value decomposition via document samplings*. — Tech. Rep. CS-1999-5, Dept. Computer Science, Duke University, North Carolina.

Gao J. and Zhang J. (2003): *Sparsification strategies in latent semantic indexing*. — Proc. 2003 Mining Workshop, San Francisco, CA, USA, pp. 93–103.

Gao J. and Zhang J. (2005): *Clustered SVD strategies in latent semantic indexing*. — Inf. Process. Manag., Vol. 41, No. 5, pp. 1051–1063.

Golub G.H. and Van Loan C.F. (1996): *Matrix Computations*. — Baltimore: The John Hopkins University Press.

Hua Y. (2000): *Searching beyond SVD for rank reduction*. — Proc. IEEE Workshop *Sensor Array and Multi Channel Signal Processing*, Cambridge, MA, USA, pp. 395–397.

Husbands P., Simon H. and Ding C. (2001): *On the use of singular value decomposition for text retrieval*. — SIAM Comput. Inf. Retrieval, pp. 145–156.

Husbands P. and Ding C. (2005): *Term norm distribution and its effects on latent semantic indexing*. — Inf. Process. Manag., Vol. 41, No. 4, pp. 777–787.

Kontostathis A. and Pottenger W.M. (2002a): *Detecting patterns in the latent semantic indexing term-term matrix*. — Proc. Workshop *Foundations of Data Mining and Discovery*, IEEE Int. Conf. *DataMining*, (ICDM 02), Maeybaski, Japan.

Kontostathis A. and Pottenger W.M. (2002b): *A mathematical view of latent semantic indexing: Tracing term co-occurrences*. — Tech. Report, LU-CSE-02-006, Lehigh University.

Landauer T.K., Foltz P.W. and Laham D. (1998): *Introduction to latent semantic analysis*. — Discourse Processes, Vol. 25, pp. 259–284.

Park H. and Elden L. (2003): *Matrix rank reduction for data analysis and feature extraction*. — Tech. Rep., Dept. Computer Science and Engineering, University of Minnesota.

Praks P., Dvorsky J., Snasel V. and Cernohorsky J.D. (2003): *On SVD free latent semantic indexing for image retrieval for application in a hard real time environment*. — Proc. IEEE Int. Conf. *Industrial Technology*, Maribor, Slovenia, pp. 466–471.

Schisler M.L. (2004): *Latent semantic indexing: using eigenvalues and the singular value decomposition in lexicographical search techniques*. — Tech. Rep., University of Missouri.

Tang J.T. (2003): *Application of principle direction divisive partitioning and SVD in information retrieval*. — Masters Proj. Rep., Dept. Computer Science, University of Kentucky, Lexington, KY.

Yates R.B. and Neto B.R. (1999): *Modern Information Retrieval*. — New Delhi: Pearson Education.

Ye Y.Q. (2000): *Comparing matrix methods in text-based information retrieval*. — Tech. Rep., School of Mathematical Sciences, Peking University.

Zang Z., Zha H. and Simon H. (2002): *Low rank approximations with sparse factors: Basic algorithms and error analysis*. — SIAM J. Matrix Anal. Appl., Vol. 23, No. 3, pp. 706–727.

Received: 4 January 2006
Revised: 26 July 2006