



ELSEVIER

Available online at www.sciencedirect.com

SCIENCE @ DIRECT®

Information Processing and Management 41 (2005) 1051–1063

**INFORMATION
PROCESSING
&
MANAGEMENT**

www.elsevier.com/locate/infoproman

Clustered SVD strategies in latent semantic indexing [☆]

Jing Gao, Jun Zhang *

*Laboratory for High Performance Scientific Computing and Computer Simulation, Department of Computer Science,
University of Kentucky, 773 Anderson Hall, Lexington, KY 40506-0046, USA*

Received 16 October 2003; accepted 21 October 2004

Available online 10 December 2004

Abstract

The text retrieval method using latent semantic indexing (LSI) technique with truncated singular value decomposition (SVD) has been intensively studied in recent years. The SVD reduces the noise contained in the original representation of the term–document matrix and improves the information retrieval accuracy. Recent studies indicate that SVD is mostly useful for small homogeneous data collections. For large inhomogeneous datasets, the performance of the SVD based text retrieval technique may deteriorate. We propose to partition a large inhomogeneous dataset into several smaller ones with clustered structure, on which we apply the truncated SVD. Our experimental results show that the clustered SVD strategies may enhance the retrieval accuracy and reduce the computing and storage costs.

© 2004 Elsevier Ltd. All rights reserved.

Keywords: Latent semantic indexing; SVD; Text retrieval; Clustering

1. Introduction

Latent semantic indexing (LSI) has emerged as a competitive text retrieval technique (Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990). LSI is a variant of the vector space model in which a low-rank approximation to the vector space representation of the database is computed (Berry, Drmac, & Jessup, 1999). LSI assumes that there is some underlying or latent structure in word usage that is partially obscured by variability in word choice. It uses a truncated singular value decomposition (SVD) of the

[☆] The research work of the authors was supported in part by the US National Science Foundation under grants CCR-9988165, CCR-0092532, ACR-0202934, and ACR-0234270, and by the US Department of Energy Office of Science under grant DE-FG02-02ER45961.

* Corresponding author. Tel.: +1 859 257 3892; fax: +1 859 323 1971.

E-mail addresses: jgao1@csr.uky.edu (J. Gao), jzhang@cs.uky.edu (J. Zhang).

URL: <http://www.cs.uky.edu/~jzhang>

term–document matrix to estimate the structure in word usage across the documents. Retrieval is performed using the databases of singular values and vectors obtained from the truncated SVD, not on the original term–document matrix. Several experimental tests show that these statistically derived vectors are more robust indicators of meaning than individual terms (Dumais, 1991). However, recent studies report that the truncated SVD strategy can be less effective for large inhomogeneous text collections (Husbands, Simon, & Ding, 2001).

In addition, for large datasets the SVD computation may be too expensive to be carried out on conventional computers. Also, the dense data structure of the truncated SVD matrices poses a huge challenge for both disk and memory spaces of conventional computers (Gao & Zhang, 2003). These problems seem to have caught researchers' attention very recently, and some strategies based on clustering (Bass & Behrens, 2003; Tang, 2003; Zha & Zhang, 1999; Zhang & Zha, 2001) have been proposed to reduce computational cost and enhance retrieval accuracy. However, some of these reports do not provide convincing experimental results on large inhomogeneous datasets.

In this paper, we construct a large inhomogeneous text dataset by merging three popular text datasets of moderate size. We then use a k -means clustering technique to partition the dataset into a few compactly structured datasets. The truncated SVD is performed on the clustered small datasets individually. Our experimental results show that the clustered SVD strategies may enhance the retrieval accuracy on large scale data collections and reduce the SVD computation time and storage space.

This paper is organized as follows. In Section 2 we review the truncated SVD concept on text data matrix. A description of the proposed clustered SVD strategies is outlined in Section 3. Section 4 presents our experimental results and discussions. We summarize this paper in Section 5.

2. Singular value decomposition

A text document may be represented by the terms (words) it contains. A set of documents can be represented by a term–document matrix. The entries (elements) of the term–document matrix are the occurrences of each word in a particular document. Suppose we have m terms and n documents, we can construct an $m \times n$ term–document matrix as $A = [a_{ij}]$, where a_{ij} denotes the frequency in which the term i occurs in the document j . Since every word does not normally appear in each document, the matrix A is usually sparse. It is estimated that, for typical term–document matrices, more than 99% of the entries may be zero. In practice, local and global weightings are applied to increase or decrease the importance of certain terms within or among the documents (Berry, Dumais, & O'Brien, 1995). In LSI, the matrix A is factored into the product of three matrices using SVD (Golub & van Loan, 1989) as

$$A = UHV^T, \quad (1)$$

where $U^T U = I_m$, and $V^T V = I_n$. I_m and I_n are the identity matrices of orders m and n , respectively. $H = \text{diag}[\sigma_1, \sigma_2, \dots, \sigma_{\min\{m,n\}}]$, $\sigma_i > 0$ for $1 < i \leq r$, and $\sigma_j = 0$ for $j > r$. Here r is the rank of A . Since U is an $m \times m$ orthonormal dense matrix and V is an $n \times n$ orthonormal dense matrix, the decomposition (1) actually consumes much more storage space than the original matrix A does.

LSI computes a low rank approximation to A using an truncated SVD (Golub & van Loan, 1989). Let k be an integer and $k \ll \min(m,n)$. We define U_k to be the first k columns of U , and V_k^T to be the first k rows of V^T (see Fig. 1). We let $H_k = \text{diag}[\sigma_1, \dots, \sigma_k]$ contain the first k largest singular values, and define

$$A_k = U_k H_k V_k^T$$

as a new pseudoterm–document matrix with reduced dimension.

Several studies reported that LSI based on truncated SVD has been compared favorably with other information retrieval techniques, in terms of the retrieval accuracy (Dumais, 1993). However, for large scale

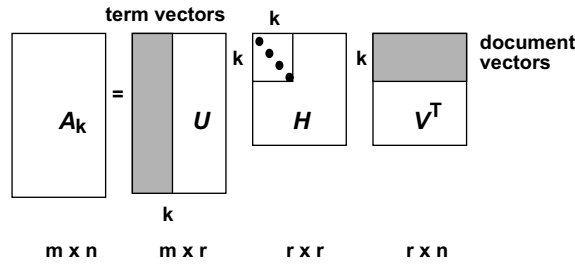


Fig. 1. Reduced dimension representation of the term–document matrix.

datasets, the computing and storage costs associated with the truncated SVD representation may be prohibitive. Recent studies also indicate that the retrieval accuracy of the truncated SVD technique can deteriorate if the document sets are large (Husbands et al., 2001).

Several strategies have been proposed to deal with LSI on large datasets. Sparsification strategy is used to remove the less important entries in the truncated SVD matrices (Gao & Zhang, 2003). Clustered and distributed SVD strategies are proposed to partition large datasets (Bass & Behrens, 2003; Tang, 2003). Unfortunately, the reports of both clustered and distributed SVD strategies are incomplete. The former (Tang, 2003) does not use large inhomogeneous datasets for experiments and the results may not valid for large unclassified datasets. The latter (Bass & Behrens, 2003) does not report any experimental results at all.

The idea used in this paper is similar to the clustered or distributed SVD strategies, but we propose some interesting retrieval strategies and conduct a detailed study with experimental tests on a large inhomogeneous dataset.

3. Clustered dataset with SVD

A large dataset can be divided into a few smaller ones, each contains data that are close in some sense. This procedure is called clustering, which is a common operation in data mining. In information and text retrieval, clustering is useful for organization and search of large text collections, since it is helpful to discover obscured words in sets of unstructured text documents.

One of the best known clustering algorithms is the k -means, with many variants (Jain, Murty, & Flynn, 1999). In our study, we use a k -means algorithm to cluster our document collection into a few tightly structured ones. Due to the high dimensionality and low sparsity of the text data, the sub-clusters usually have a certain “self-similar” behavior, i.e., documents of the similar classes are grouped into the same cluster. The centroid vector (defined below) of a tightly structured cluster can usually capture the general description of documents in that cluster. An ideal cluster contains homogeneous documents that are relevant to each other.

3.1. Document clustering

First we use the k -means algorithm to partition a large collection of documents into s subsets of tightly structured documents. Let the set of document vectors be

$$A = [a_1, a_2, \dots, a_i, \dots, a_n],$$

where a_i is the i th document in the collection. We would like to partition the documents into s sub-collections $\{\pi_i\}_{i=1}^s$ such that

$$\bigcup_{j=1}^s \pi_j = \{a_1, a_2, \dots, a_n\} \quad \text{and} \quad \pi_j \cap \pi_i = \phi \quad \text{if } j \neq i.$$

For each fixed $1 \leq j \leq s$, the centroid vector of each cluster is defined as

$$\tilde{c}_j = \frac{1}{n_j} \sum_{a_i \in \pi_j} a_i,$$

where $n_j = |\pi_j|$ is the number of documents in π_j . We normalize the centroid vectors such that

$$c_j = \frac{\tilde{c}_j}{\|\tilde{c}_j\|}, \quad j = 1, 2, \dots, s.$$

An intuitive definition of the clusters is that, if

$$a_i \in \pi_j,$$

then

$$a_i^T c_j > a_i^T c_l \quad \text{for } l = 1, 2, \dots, s, \quad l \neq j,$$

i.e., documents in π_j are closer to its centroid than to the other centroids. If the clustering is good enough and each cluster is compact enough, the centroid vector may represent the abstract concept of the cluster.

3.2. Query strategy on clustered database

After clustering the original document collection, we have

$$A' = [\pi_1, \pi_2, \dots, \pi_s] = [a'_1, a'_2, \dots, a'_n] \quad (\text{reordered}).$$

We also obtain the centroid vectors of each clusters:

$$C = [c_1, c_2, \dots, c_s].$$

Given a query vector q , we can find the closest matching clusters by computing and comparing the cosine values (similarity measures) between the query vector and the centroid vectors

$$q^T C = [q^T c_1, q^T c_2, \dots, q^T c_s].$$

We may retrieve the closest matching documents by computing and comparing all or part of

$$q^T \pi_1, q^T \pi_2, \dots, q^T \pi_s.$$

If all clusters are queried, no difference is made, compared to query the original unclustered collection. If part of the cluster list are queried, it may be considered as a faster retrieval strategy from the original database. The problems associated with querying the original document collection, such as polysemy and synonymy, will still persist.

To have better retrieval accuracy, we apply the LSI technique with truncated SVD on each individual cluster. In this way, either we query all or part of the cluster list, the truncated SVD encoded document sets are different from the original dataset.

3.3. Three strategies for retrieval

3.3.1. Non-clustered retrieval (NC + SVD)

This is to retrieve on the original large dataset, pre-processed with truncated SVD. No clustering is done.

3.3.2. Full clustered retrieval (FC + SVD)

After clustering the large dataset with a k -means algorithm, we need use SVD to approximate the matrix of the document vectors in each cluster. We sort the similarity measure of query to each document vector and return the documents which have the highest similarity measure (cosine values) in all documents. Since no clustering algorithm can group all relative documents into the same sub-cluster precisely, it is reasonable to expect that some documents may be misclassified.

3.3.3. Partial clustered retrieval (PC + SVD)

It is faster to search only a few clusters that are closely related to the given query. For a certain query, we ignore some clusters based on the inner products of this query and the centroid of each cluster. If we find that a cluster is not similar to the given query, that particular cluster will not be searched. The queried dataset consists of

$$\bar{A} = \bigcup_j \pi_j,$$

where the centroid vectors of the π_j s have the highest similarity values in $q^T C$.

3.4. Computing and storage cost analysis

3.4.1. Computing cost of truncated SVD

The cost of computing the truncated SVD of a sparse matrix A can be expressed as (Berry, 1992)

$$I \times \text{cost}(A^T A x) + k \times \text{cost}(A x), \tag{2}$$

where I is the number of iterations required by a Lanczos-type procedure to approximate the eigensystem of $A^T A$, and k is the rank of the truncated SVD. Here x is an arbitrary document vector. The dominant cost in this computation is the number of iterations I and the complexity of the sparse matrix multiplications by A (and A^T).

In the simplest case, let A be partitioned as two parts $[A_1, A_2]$, we can see, for the dominant cost of the SVD computation, that

$$A^T A = [A_1, A_2]^T [A_1, A_2] = A_1^T A_2 + A_1^T A_1 + A_2^T A_1 + A_2^T A_2.$$

To simplify the analysis, we assume that the number of iterations in the Lanczos-type procedure is the same for the large matrix and the partitioned matrices. Then we have

$$\text{cost}(A^T A x) = \text{cost}(A_1^T A_1 x) + \text{cost}(A_2^T A_2 x) + \text{cost}(A_1^T A_2 x) + \text{cost}(A_2^T A_1 x). \tag{3}$$

Note that the first two parts in the right-hand side of Eq. (3) are the corresponding cost associated with computing the truncated SVD of the two smaller matrices A_1 and A_2 .

In the second part of expression (2), the ranks of the truncated SVD of the clustered matrices are usually smaller than that of the large matrix. It follows that the cost of computing the truncated SVD on the clustered datasets is less than that of computing the truncated SVD on the combined large dataset.

3.4.2. Storage cost of clustered SVD

Let m and n be the number of terms and documents in the original dataset, respectively. Suppose there are s clustered subsets $\{\pi_1, \pi_2, \dots, \pi_i, \dots, \pi_s\}$ are generated, each one has m_i terms and n_i documents. It is obvious that

$$\sum_{i=1}^s m_i \geq m \quad \text{and} \quad \sum_{i=1}^s n_i = n.$$

Let k be the rank of the truncated SVD of the original dataset, k_i be that of the truncated SVD of the i th clustered subset. Based on our experience, we have $k_i \leq k$ for all i . In our experiments, we usually see $\sum_{i=1}^s k_i \approx k$.

The storage cost of the truncated SVD matrices of the original database is $k(m+n+1)$ data values, and that of each clustered subset is $k_i(m_i+n_i+1)$. The total storage cost for the truncated SVD matrices of all clustered subsets is

$$S_{\text{total}} = \sum_{i=1}^s k_i(m_i + n_i + 1). \quad (4)$$

In a special case, we assume that the sizes of the clustered subsets are the same, the terms of individual clusters are orthogonal, we have $m_i = m/s$ and $n_i = n/s$ for all $1 \leq i \leq s$. If we further assume that $k_i = k/s$ to have a rough idea on the approximate storage cost of the clustered SVD strategies, then we have

$$S_{\text{total}} = \sum_{i=1}^s \frac{k}{s} \left(\frac{m}{s} + \frac{n}{s} + 1 \right) = \frac{1}{s^2} \sum_{i=1}^s k(m+n+s).$$

It follows that the storage cost of the truncated SVD on the clustered subsets is inversely quadratically proportional to the storage cost of the truncated SVD on the original dataset. Since the storage cost of the truncated SVD on large scale datasets is a major disadvantage of the LSI technique, the clustered SVD strategies provide an attractive solution to reducing the storage cost.

4. Evaluation and results

To evaluate our clustered SVD strategies, we apply it to a large collective inhomogeneous database which consists of three popular document databases: CRAN, MED, and CISI. The large database collection is generated by merging terms and documents of these three databases. The terms of the large database collection are obtained by adding up all terms of these three matrices. If a term occurs in more than one dataset, it is listed only once. The documents of the large matrix are the sum of all documents of the three databases. The combined dataset is referred to as the Merged DB. The queries used are the original queries provided with the three databases. Similar strategy was used in [Dhillon and Modha \(2001\)](#) to create a large inhomogeneous dataset. The database information is shown in [Table 1](#).

A standard way to evaluate the performance of an information retrieval system is to compute precision and recall values. We use precision–recall pair to evaluate our retrieval results. The precision is the proportion of the relevant documents in the set returned to the user; the recall is the proportion of all relevant documents in the collection that are retrieved by the system. We average the precision of all queries at fixed recall values such as 10%, 20%, ..., 90%. The average precision methods that we use is the N -point interpolated average precision

Table 1
Description of the three datasets and the merged dataset

Database	Order	Number of queries	Source
CRAN	4612 × 1398	225	Science indices
MED	5831 × 1033	30	Medical documents
CISI	5609 × 1460	112	Fluid dynamics
Merged DB	11,152 × 3891	225 + 30 + 112	Merge CRAN, MED and CISI

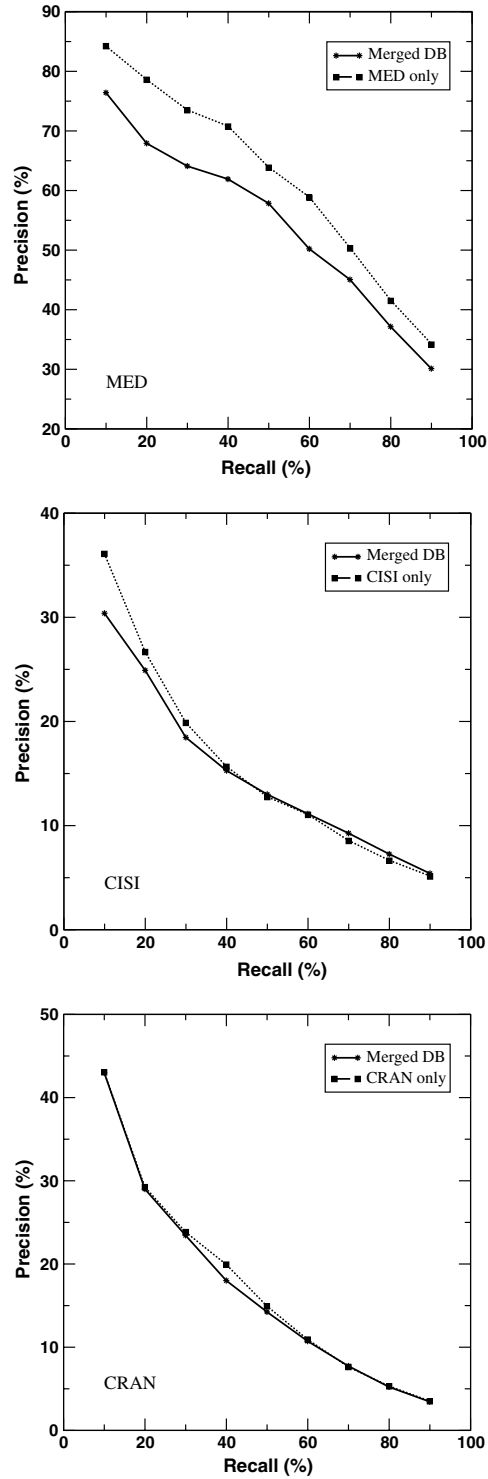


Fig. 2. Performance comparisons of the truncated SVD strategy on the Merged DB and the individual databases.

$$\frac{1}{N} \sum_{i=0}^{N-1} \bar{P}\left(\frac{i}{N-1}\right),$$

where

$$\bar{p}(x) = \max_{\frac{r_i}{m} \leq x} P_i,$$

and r_i and P_i are the recall and precision for the i th point. The precision values that we report are the average precision over the number of queries at a given recall value. In precision–recall pair, the higher curves indicate better performance of an information retrieval system. In this section, we will evaluate three strategies for text retrieval based on their precision–recall performance.

4.1. Experimental results without clustering

We first perform truncated SVD operation on the combined dataset Merged DB, and then compare the retrieval accuracy of the 367 queries with that on the original individual datasets. To obtain good performance on the Merged DB, we choose SVD rank $k = 300$ (after some experiments). Fig. 2 shows the precision–recall results of the Merged DB and the individual datasets. The retrieval accuracy of the Merged DB is worse than that of the individual homogeneous datasets. The retrieval accuracy degradation is most severe for the dataset MED, and least significant for the dataset CRAN.

4.2. Experimental result with clustering

4.2.1. Full clustered SVD retrieval (FC + SVD)

We first partition the large data collection A into a number of subsets $\{\pi_1, \pi_2, \dots, \pi_s\}$. Then, we compute the low-rank approximation of each subset with SVD. Finally, we retrieve on all subsets and compute their average precision and recall values. We point out that the full clustered SVD retrieval strategy is different from the SVD without clustering strategy. The former applies SVD on the individual (small) datasets (from the clustering), the latter on the whole (large) dataset.

To see how well the combined dataset represents the original three datasets, we partition the Merged DB into three clusters using the k -means algorithm. Table 2 shows the comparison between the computed subsets $\{\pi_1, \pi_2, \pi_3\}$ and the original individual datasets {MED, CISI, CRAN}. The data in Table 2 shows that the k -means algorithm has the ability to classify the documents with high level similarity in this particular collection of datasets. Only 23 and 4 documents in MED are misclassified into π_2 and π_3 , respectively. So the subset π_1 has 1006 documents, the subset π_2 has 1483 documents, and the subset π_3 has 1402 documents.

To test the performance of the strategy FC + SVD, we use the k -means algorithm to cluster the Merged DB into 4, 6, 8, 16, and 64 subsets. For the 4-cluster, 6-cluster, 8-cluster, and 16-cluster cases, the best rank of SVD is around 100. For the 64-cluster case, the best rank of the SVD of each cluster is 20. For each clustering (3, 4, 6, 8-cluster) case, we plot the precision–recall curve for the three different query sets (see Fig. 3). For the query set of MED, the best number of clusters for retrieval is 4. For the query set of CISI,

Table 2
Comparison of classified subsets and the original individual datasets

Databases	π_1	π_2	π_3
MED	1006	23	4
CISI	0	1460	0
CRAN	0	0	1398

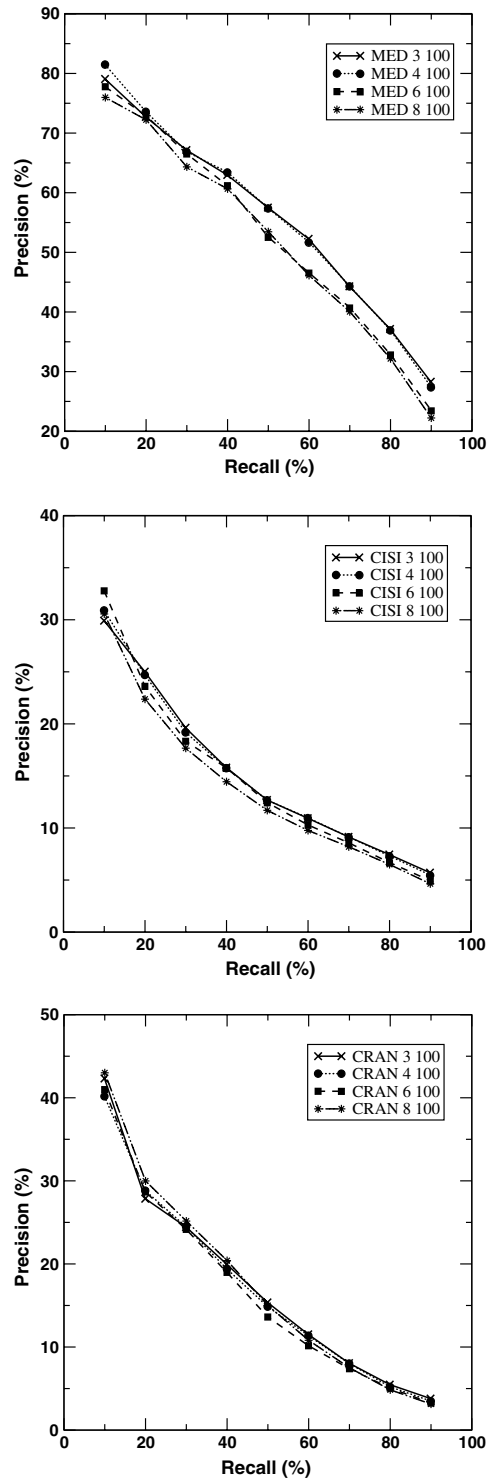


Fig. 3. Performance comparison of the FC + SVD strategy with different clustered subsets.

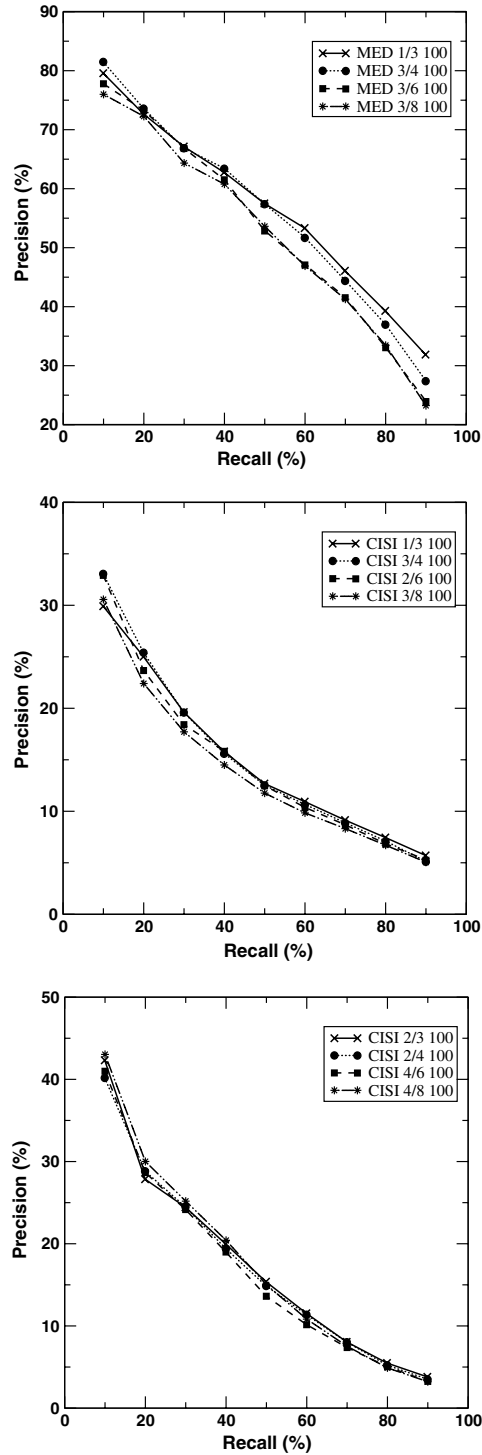


Fig. 4. Performance comparison of the PC + SVD strategy with different clustered subsets.

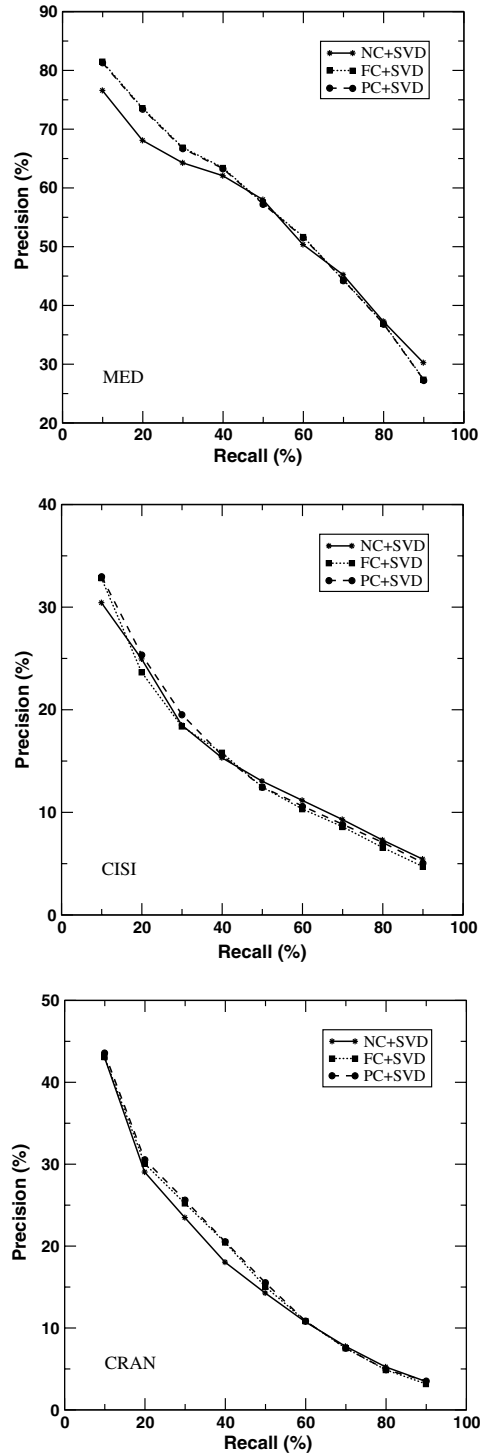


Fig. 5. Precision–recall pair comparisons for three different SVD strategies.

the 6-cluster solution yields the best performance. For the query set of CRAN, the 8-cluster solution is shown to be the best.

4.2.2. Partial clustered SVD retrieval (PC + SVD)

For this strategy, we only query part of the relevant clusters which contain most relevant documents. We split the original term–document matrix into 3-cluster, 4-cluster, 6-cluster, 8-cluster, 16-cluster datasets and compute the inner products of the query to the centroid of each cluster for each clustering case. For different queries, we do not need to search all clusters, since some clusters are irrelevant for particular queries. In Fig. 4, we plot the best query results for different clustering cases. For queries of the MED dataset, choosing 3 in the 4-cluster is the best. For queries of the CISI dataset, taking 3 in the 4-cluster case yields the best results. For queries of the CRAN dataset, we find that choosing 4 clusters from the 8-cluster case gives us the best precision (Fig. 5).

4.3. Comparison of all strategies

In Table 3, we present the retrieval precision comparison of the three database pre-processing strategies. As we did previously, the rank of the truncated SVD for the Merged DB is 300. For the clustered strategies, the rank of the truncated SVD is 100. With the FC + SVD strategy, we use a 4-cluster database for the query sets of MED and CISI, and a 8-cluster database for the query set of CRAN. In general, the retrieval precision of the FC + SVD strategy is better than that of the NC + SVD strategy. However, the retrieval precision of the PC + SVD strategy is seen to be the best in most cases.

It seems that the improvement from the clustered SVD strategies is more on the MED database than on the CISI and CRAN databases. This is probably because that MED is less homogeneous than CISI and CRAN (see Table 2). The queries associated with MED are also less homogeneous and they may mingle with some documents in CISI and CRAN. Thus, clustering helps MED restrict the influence from CISI and CRAN. On the other hand, CISI and CRAN are already highly homogeneous and their queries are also highly homogeneous and are not affected by the documents in MED. Additional clustering does not help very much.

4.4. Computing and storage costs

Table 4 lists the CPU time in seconds (on a SUN Ultra 10 workstation running at 500 MHz with 128 Mb memory under the MATLAB environment) and the storage cost in MB in computing the truncated SVD matrices in the non-clustered SVD (original database) and clustered SVD (4 clusters) strategies. It can be seen that both the computing and storage costs are reduced substantially in the clustered SVD strategy.

Table 3
Retrieval precision comparison of different strategies at several recall levels

Database/recall	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
MED (NC + SVD)	0.7620	0.6771	0.6389	0.6171	0.5764	0.5001	0.4486	0.3696	0.2993
MED (FC + SVD)	0.8108	0.7322	0.6651	0.6309	0.5707	0.5139	0.4405	0.3668	0.2717
MED (PC + SVD)	0.8108	0.7322	0.6651	0.6309	0.5708	0.5141	0.4413	0.3672	0.2719
CISI (NC + SVD)	0.3030	0.2483	0.1839	0.1521	0.1293	0.1106	0.0920	0.0720	0.0534
CISI (FC + SVD)	0.3269	0.2354	0.1827	0.1570	0.1237	0.1023	0.0851	0.0648	0.0463
CISI (PC + SVD)	0.3289	0.2527	0.1946	0.1548	0.1241	0.1055	0.0877	0.0700	0.0503
CRAN (NC + SVD)	0.4295	0.2896	0.2340	0.1797	0.1421	0.1069	0.0771	0.0519	0.0342
CRAN (FC + SVD)	0.4299	0.2998	0.2515	0.2039	0.1499	0.1080	0.0750	0.0484	0.0313
CRAN (PC + SVD)	0.4302	0.3000	0.2517	0.2041	0.1501	0.1082	0.0752	0.0489	0.0321

Table 4
Computing and storage costs of the non-clustered and clustered SVD strategies

	Non-clustered SVD	Clustered SVD
CPU time (s)	2.3×10^5	5.3×10^4
Storage (MB)	4.2	1.2

5. Summary

We merged three popular text retrieval databases to form a large inhomogeneous test dataset. We proposed to partition a large text dataset into a few tightly clustered subsets. We then performed truncated SVD computation on the clustered datasets individually. Our experimental results confirm claims by other researchers that the retrieval accuracy of the LSI technique may deteriorate when retrieval on large size inhomogeneous datasets. We show that the accuracy of the LSI technique may be improved when retrieval on clustered subsets, especially retrieval on part of the most relevant clustered subsets. We also show that clustered SVD matrices have the advantages of lower computing and storage costs, compared with the SVD matrices computed from the original large databases.

Like the classical LSI, there are some parameters that may affect the performance of a clustered SVD retrieval system. The most important parameters are the rank of the truncated SVD matrices and the number of clusters chosen in the partial clustered SVD retrieval. How to determine the best rank of the truncated SVD matrices is an unsolved classical problem in LSI. The number of clusters chosen for a particular query is determined by the initial query on the centroid vector. A suitable threshold value could be determined based on certain heuristics and experiments, which could be an interesting topic for future study.

References

- Bass, D., & Behrens, C. (2003). Distributed LSI: scalable concept-based information retrieval with high semantic resolution. In *Proceedings of the 2003 Text Mining Workshop* (pp. 72–82), San Francisco, CA.
- Berry, M. W. (1992). Large scale singular value computations. *International Journal of Supercomputer Applications and High Performance Computing*, 6, 13–49.
- Berry, M. W., Drmac, Z., & Jessup, E. R. (1999). Matrix, vector space, and information retrieval. *SIAM Review*, 41, 335–362.
- Berry, M. W., Dumais, S. T., & O'Brien, G. W. (1995). Using linear algebra for intelligent information retrieval. *SIAM Review*, 37, 73–595.
- Deerwester, S., Dumais, S. T., Furnas, G., Landauer, T., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of American Society for Information Science*, 41, 391–407.
- Dhillon, I. S., & Modha, D. S. (2001). Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42, 143–175.
- Dumais, S. T. (1991). Improving the retrieval of information from external sources. *Behaviors Research Methods, Instruments, and Computers*, 23, 229–236.
- Dumais, S. T. (1993). LSI meets TREC: A status report. In *The First Text REtrieval Conference*, NIST Special Pub. 500-207 (pp. 137–152), Gaithersburg, MD.
- Gao, J., & Zhang, J. (2003). Sparsification strategies in latent semantic indexing. In *Proceedings of the 2003 Text Mining Workshop* (pp. 93–103), San Francisco, CA.
- Golub, G., & van Loan, C. (1989). *Matrix computation* (2nd ed.). Baltimore, MD: John Hopkins.
- Husbands, P., Simon, H., & Ding, C. (2001). On the use of singular value decomposition for text retrieval. In *Computational Information Retrieval* (SIAM) (pp. 45–156), Philadelphia, PA.
- Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys*, 31(3), 264–323.
- Tang, J.-T. (2003). Application of principle direction divisive partitioning and singular value decomposition in information retrieval. *Masters Project Report*. Department of Computer Science, University of Kentucky, Lexington, KY.
- Zha, H., & Zhang, Z. (1999). On matrices with low-rank-plus-shift structures: partial SVD and latent semantic indexing. *SIAM Journal on Matrix Analysis and Applications*, 21, 522–536.
- Zhang, Z., & Zha, H. (2001). Structure and perturbation analysis of truncated SVD for column-partitioned matrices. *SIAM Journal on Matrix Analysis and Applications*, 22, 1245–1262.