

Text Retrieval Using Sparsified Concept Decomposition Matrix ^{*}

Jing Gao and 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

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

Abstract. We examine text retrieval strategies using the sparsified concept decomposition matrix. The centroid vector of a tightly structured text collection provides a general description of text documents in that collection. The union of the centroid vectors forms a concept matrix. The original text data matrix can be projected into the concept space spanned by the concept vectors. We propose a procedure to conduct text retrieval based on the sparsified concept decomposition (SCD) matrix. Our experimental results show that text retrieval based on SCD may enhance the retrieval accuracy and reduce the storage cost, compared with the popular text retrieval technique based on latent semantic indexing with singular value decomposition.

1 Introduction

Many popular text retrieval techniques are based on the vector space model. Each document is represented as a vector of certain weighted word frequencies. A text dataset is modeled as a term-document matrix whose rows are terms (words) and columns are document vectors. A user's query of a database can be represented as a document vector. Relevant documents are found by querying the database. In other words, query matching is to find the documents that are most similar to the query in some sense. A measure of similarity, e.g., the cosine value of the angles between the vectors, is used to select the documents that are most relevant to the query vector. For real-life term-document matrices, both the number of terms and the number of documents are large, which result in high dimensionality of the database.

A standard strategy in dealing with high dimensional databases is dimensionality reduction. In text retrieval community, dimensionality reduction using latent semantic indexing (LSI) based on singular value decomposition (SVD)

^{*} Technical Report No. 412-04, Department of Computer Science, University of Kentucky, Lexington, KY, 2004. The research work of the authors was supported in part by the U.S. National Science Foundation under grants CCR-0092532, and ACR-0202934, and in part by the U.S. Department of Energy Office of Science under grant DE-FG02-02ER45961.

of the term-document matrix is popular [3]. The truncated SVD low-rank approximation to the vector space representation of the database can capture the semantics of the documents [2, 3, 7], and is used to estimate the structure in word usage across the documents [2]. Experimental results show that text retrieval performance is improved by SVD. However, the truncated SVD matrices are dense, and usually consume more storage space than the original sparse term-document matrix [1, 6]. Several strategies have been proposed to reduce the memory cost of SVD, including the matrix sparsification strategies [6].

In this paper, we examine the advantages of text retrieval strategies using the sparsified concept decomposition (CD) matrix. We propose strategies to make query processing more efficient with CD. In particular, we propose to compute the inverse of the normal matrix of the concept matrix explicitly in order to facilitate real-life parallel query processing. We further propose to sparsify the dense inverse matrix to reduce storage cost. Our experimental results indicate that, compared with SVD, the new retrieval technique reduces storage cost and improves the performance of text retrieval.

Concept decomposition was proposed and analyzed in [4]. Text retrieval using concept projection from the PDDP clustering algorithm was experimented in [9]. Our contribution is mainly in advocating text retrieval using the CD matrix (not the concept projection as in [9]), the explicit computation procedure, and the sparsification of the inverse matrix.

This paper is organized as follows. In Section 2 we review the concept decomposition method and propose our sparsification strategy. Section 3 presents experimental results and comparisons. We summarize this paper in Section 4.

2 Concept Decomposition and Query Retrieval

A large data collection can be divided into a few smaller subcollections, 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 clustering [8]. In our study, we use a standard k -means algorithm to cluster a document collection into a few tightly structured ones. These subclusters may have a certain mutual similarity behavior, i.e., documents of similar classes are grouped into the same cluster. The centroid of a tightly structured cluster can usually capture the general description of the documents in that cluster.

2.1 Document Clustering Based on k -means Algorithm

Let the term-document matrix of dimension $m \times n$ 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 k subcollections $\{\pi_i, i = 1, \dots, k\}$ such that

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

For the j th cluster, the centroid vector of π_j is computed as

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

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

$$c_j = \frac{\tilde{c}_j}{\|\tilde{c}_j\|_2}, \quad j = 1, 2, \dots, k.$$

If the clustering is good enough and each cluster is compact enough, the centroid vector may represent the abstract concept of the cluster very well.

2.2 Concept Decomposition

After the clustering, we have $A' = [\pi_1, \pi_2, \dots, \pi_k]$, and a collection of the centroid vectors of the clusters $C = [c_1, c_2, \dots, c_k]$. The matrix C is called the concept matrix [4]. Based on this concept matrix, we may use a straightforward query procedure as the following. For a given query vector q , we can find the closest matching clusters by computing and comparing the cosine similarity (inner product) values $q^T C = [q^T c_1, q^T c_2, \dots, q^T c_k]$. We may obtain the closest matching documents by computing and comparing all or part of $q^T \pi_1, q^T \pi_2, \dots, q^T \pi_k$.

To have better retrieval accuracy, the LSI technique with truncated SVD can be applied to each individual clusters [5]. Although retrieval performance is improved, the clustered SVD strategy still consumes much CPU time and storage space [5]. To further alleviate the problems associated with SVD, we examine text retrieval strategies using the concept matrix [9].

According to [4], the basic idea of concept decomposition (CD) is to project a high dimensional matrix into a lower-rank concept space. The concept vector c_i obtained by normalizing the centroid vectors provides the concept projection. Without loss of generality, we assume that the matrix C is of full rank k . We can project the document matrix onto the concept space spanned by the column vectors of the concept matrix $\tilde{A}_k = C M^*$, such that M^* is a $k \times n$ matrix from the least squares problem $M^* = \arg \min_M \|A - C M\|_F^2$, where $\|\cdot\|_F$ is the matrix Frobenius norm. The closed form solution of this problem is $M^* = (C^T C)^{-1} C^T A$ [4]. The CD of the document matrix is $\tilde{A}_k = C (C^T C)^{-1} C^T A$, where C is of dimension $m \times k$, and $k \ll \{m, n\}$.

2.3 Retrieval Procedure

Applying a query vector q on the CD matrix, we obtain

$$q^T \tilde{A}_k = q^T C (C^T C)^{-1} C^T A.$$

This procedure can be carried out in a few steps. First, we can compute $q_1^T = q^T C$. Then, we obtain

$$q_2^T = q_1^T (C^T C)^{-1}, \quad (1)$$

and $q_3 = C q_2$. Finally, the query procedure is computed as $q^T \tilde{A}_k = q_3^T A$. The concept matrix is largely sparse, although denser than A . The entire procedure can be considered as expanding the query q to q_3 and then retrieval on the original matrix by $q_3^T A$.

All steps can be carried out straightforwardly, except Step (1), which requires the inverse of the normal matrix of the concept matrix. Although not directly applied to text retrieval, [4] suggests to compute the QR factorization of the concept matrix as $C = QR$, where Q is an orthogonal matrix, i.e., $Q^T Q = I$, and R is an upper triangular matrix of rank k . Because

$$C^T C = (Q^T R)^T Q^T R = R^T Q Q^T R = R^T R,$$

Step (1) becomes $q_2^T = q_1^T (R^T R)^{-1}$. Since $(R^T R)^{-1}$ is symmetric, we have $q_2 = (R^T R)^{-1} q_1$. This leads to $(R^T R) q_2 = q_1$, which can be solved to get q_2 by a forward elimination and back substitution as in standard Cholesky factorization of a symmetric positive definite matrix.

The QR factorization is fast, straightforward, and is used in the results reported in [9]. However, in realistic large scale text retrieval, such as those used in the online search engines, the query operations are usually computed by many processors with shared or distributed memory simultaneously. The forward elimination and back substitution procedure is inherently sequential and thus may present a bottleneck in the response time of a text retrieval system.

We prefer to invert the matrix $C^T C$ explicitly, so that Step (1) can be evaluated as a matrix vector operation, which can be carried out efficiently on parallel platforms. Since k is usually far smaller than both m and n , the computational and storage cost of computing $D = (C^T C)^{-1}$ explicitly is acceptable. The storage cost can be reduced by sparsification as we discuss below.

2.4 Sparsification Strategy

The matrix $(C^T C)^{-1}$ is of dimension $k \times k$ and dense. It may have many small size (or zero) entries. They play less important role during the query operations (inner product). Therefore, it is possible to replace these small size entries in the approximation matrix. If enough zeros are found in the matrix, some sparse matrix storage formats may be used to reduce the unnecessary storage space.

Following our previous work on sparsifying the SVD matrices, we propose a sparsification strategy to sparsify the matrix $(C^T C)^{-1}$. Given a threshold parameter ϵ , for any entry d_{ij} in $(C^T C)^{-1}$, if $|d_{ij}| < \epsilon$, we set $d_{ij} = 0$.

In our tests of sparsified SVD matrices, we found that the sparsification strategy may sometimes improve the performance of the SVD based text retrieval system, in addition to reduce storage cost [6]. The same effect will be demonstrated with the sparsified concept decomposition (SCD) technique.

3 Results and Discussion

To evaluate the performance of the text retrieval technique based on the concept decomposition (CD) matrix, we apply it to three popular text databases: CRAN, MED, and CISI [6]. The term-document matrices of these three databases are downloaded from <http://www.cs.utk.edu/~lsi/>.

A standard way to evaluate the performance of an information retrieval system is to compute the precision and recall. 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 0, 10%, 20%, . . . , 90%. 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.

We first use the k -means algorithm to divide the original term-document matrix into 32, 64, 128, 256, and 500 small data clusters. For each data cluster, we compute the concept vector, and form a concept matrix collectively. To study the performance of the text retrieval technique based on concept decomposition with or without sparsification, we compare the query results with those of SVD. For SVD, we choose the reduced rank $k = 100$, as previous tests show good results with LSI by choosing the reduced rank at about 1/10 of the documents in the collection [2, 6].

We tested the sparsification of $D = (C^T C)^{-1}$ with the threshold values 0.01, 0.02, 0.03, and 0.04. For all three databases divided into 256 clusters, if $\epsilon = 0.04$, more than 60% of the nonzero entries would be set to zero in D .

Fig. 1 shows the precision-recall curves from CD and SCD with different number of clusters (64, 128, 256, and 500), compared with SVD. For the method of CD without sparsification, we just provide the best query results for every database.

When the number of cluster is 128, CD without sparsification reaches its best results for MED. For CISI and CRAN, the best results are obtained when the number of clusters is 500.

After sparsification with $\epsilon = 0.03$, the best results for MED are obtained when the number of clusters is 128 and 86% of the entries in D are dropped. For both CRAN and CISI, the best number of clusters is 256, but the best ϵ values are 0.02 and 0.03, respectively, under which 56% and 64% of the entries in D are dropped.

For MED and CISI, CD with and without sparsification have better performance than SVD, especially at the low recall range. For example, SCD has around 30% improvement over SVD at recall=10% for CISI. For CRAN, the precisions of SCD are significantly better than that of SVD and CD. Compared with SVD and CD, the sparsification procedure not only reduces the CPU time and storage space, but also improves the query precision.

We also compare the storage cost and the total CPU time for the query procedure between SVD and SCD. The results are shown in Table 1. The CPU times are counted for all queries for each database. Since SCD has a small rank and

the sparsification strategy further removes many small entries, the query time and storage cost of SCD are significantly smaller, compared to SVD. The experiments were carried out in MATLAB on a SUN Ultra 10 Workstation running at 500 MHz with 128 MB memory.

Databases	MED		CISI		CRAN	
Techniques	SVD	SCD	SVD	SCD	SVD	SCD
CPU time (s)	4.5×10^2	0.3×10^2	6.2×10^2	2.1×10^2	1.4×10^3	5.5×10^2
Memory (MB)	1.3	0.26	1.6	0.31	1.4	0.29

Table 1. Comparisons of CPU time and storage cost for SVD and SCD.

4 Summary

We proposed to use a sparsification technique to enhance the performance of text retrieval using the concept decomposition matrix, which was proposed for dimensionality reduction of vector space information retrieval model [4]. We give experimental results of query performance on three well-known databases using our sparsified concept decomposition technique and compare it with the SVD and the concept decomposition technique without sparsification. We found that the sparsified concept decomposition technique usually has better performance in most cases. The query time and the storage cost of the sparsified concept decomposition technique are substantially smaller than that of SVD.

References

1. Bassu, D., Behrens, C.: Distributed LSI: scalable concept-based information retrieval with high semantic resolution. In *Proceedings of the 2003 Text Mining Workshop*, San Francisco, CA (2003) 72–82
2. Berry, M.W., Drmac, Z., Jessup, E.R.: Matrix, vector space, and information retrieval. *SIAM Rev.* **41** (1999) 335–362
3. Deerwester, S., Dumais, S.T., Furnas, G., Landauer, T., Harshman, R.: Indexing by latent semantic analysis. *J. Amer. Soc. Infor. Sci.* **41** (1990) 391–407
4. Dhillon, I.S., Modha, D.S.: Concept decompositions for large sparse text data using clustering. *Machine Learning* **42** (2001) 143–175
5. Gao, J., Zhang, J.: Clustered SVD strategies in latent semantic indexing. Technical Report No. 382-03, Department of Computer Science, University of Kentucky, Lexington, KY (2003)
6. Gao, J., Zhang, J.: Sparsification strategies in latent semantic indexing. In *Proceedings of the 2003 Text Mining Workshop*, San Francisco, CA (2003) 93–103
7. Husbands, P., Simon, H., Ding, C.: On the use of singular value decomposition for text retrieval. In *Computational Information Retrieval*, SIAM, Philadelphia, PA (2001) 145–156

8. Jain, A., Dubes, R.C.: *Algorithms for Clustering Data*. Prentice Hall (1998)
9. Sasaki, M., Kita, K.: Information retrieval system using concept projection based on PDDP algorithm. In *Pacific Association for Computational Linguistics (PACLING2001)* (2001) 243–249

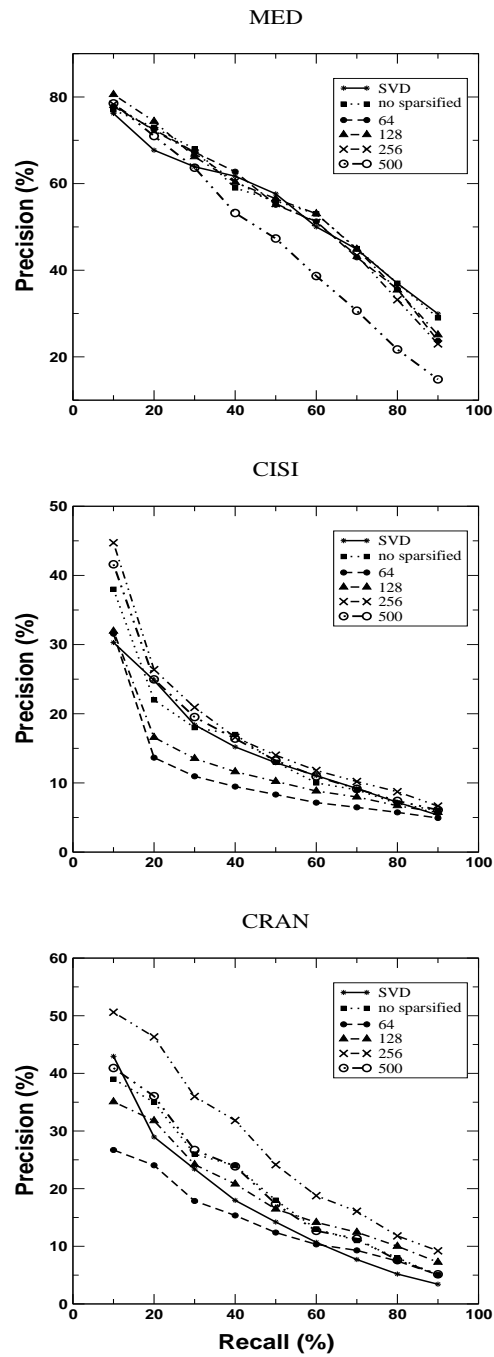


Fig. 1. Comparisons of precision-recall results using SVD, CD, and SCD.