Approximate Matrix Decomposition Techniques in Information Retrieval [*]

Chi Shen[†] and Duran Williams [‡]

Abstract— **In this paper, a new class of numerical matrix computation methods has been developed in computing the approximate decomposition matrix in concept decomposition technique. These methods utilize the knowledge of matrix sparsity pattern techniques in preconditioning field. Compared to the straightforward implementation of the concept matrix decomposition, these methods are computationally more efficient, fast in computing the ranking vector and require much less memory while maintaining retrieval accuracy.**

Keywords: ***term-document matrix, concept decomposition matrix, sparsity pattern, sparse matrix approximation, least-squares***

1 Introduction

Recently, concept decomposition based on document clustering strategies has drawn researchers' attention [3]. In this approach, data is modeled as a term-document matrix, and a large document dataset is first partitioned into tightly structured clusters. The centroid of the documents in a cluster is computed and normalized as the *concept vector* of that cluster, as it represents the general meaning of that group of tightly structured homogeneous documents. The term-document matrix projected on the concept vectors is compared favorably to the truncated Singular Value Decomposition (SVD) of the original term-document matrix. However, the numerical computation based on the straightforward implementation of the concept matrix decomposition is expensive, as there is an inverse matrix of the normal matrix formed by the concept vector matrix is required for each query computation.

[*]The research work of the authors was supported by Kentucky EPSCoR under grant NSF 4800000823

[†]E-mail: chi.shen@kysu.edu

[‡]E-mail: duran.williams@kysu.edu. Both authors are from Division of Computer and Technical Sciences, Kentucky State University, Frankfort, KY 40601, USA

In this paper, we develop sparse matrix approximation techniques to compute the matrix associated with the concept matrix decomposition. These computations are related to weighted least squares optimization and approximate pseudo matrix inversion. Our experimental results show that these approaches greatly reduce the time and memory space required to compute the decomposition matrix and perform document retrieval.

This paper is organized as follows. Section 2 introduces document clustering and concept decomposition. Section 3 discusses the approximate least squares based approaches, the key part of the numerical computation in this paper. Numerical experiments are given in Section 4. We summarize this paper in Section 5.

2 Document Clustering and Concept Vectors

Clustering is a division of data into groups of similar objects. One of the best known clustering algorithms is the k-means, with many variants [7, 8, 9]. In our study, we use a k-means algorithm, implemented in [4], to cluster our document collection into a few tightly structured ones. An ideal cluster contains homogeneous documents that are relevant to each other.

2.1 Document Clustering

Let the set of document vectors be $A = [a_1, a_2, \ldots, a_j, \ldots, a_n]$, where a_j is the jth document in the collection. We partition the documents into k sub-collections $\{\pi_j\}_{j=1}^{k}$ such that

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

For each fixed $1 \leq j \leq k$, 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

$$c_j = \frac{\tilde{c_j}}{\| \tilde{c_j} \|}, \quad j = 1, 2, \ldots, k.$$

An intuitive definition of the clusters is that, if $a_i \in \pi_j$, then

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

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. So they are also called *concept vectors* [3]. The *concept matrix* can be defined as an $m \times k$ matrix such that,

$$C_k = [c_1, c_2, \ldots, c_k].$$

The concept matrix C_k is still a sparse matrix, as each column c_j is sparse. The degree of sparsity of C_k is inversely proportional to the number of clusters generated. If we assume that the concept vectors are linearly independent, the concept matrix has rank k.

For any partitioning of the document vectors, we can define the corresponding concept decomposition A_k of the term-document matrix A as the least squares approximation of A onto the column space of the concept matrix C_k. We write the concept decomposition as an $m \times n$ matrix

$$A_k = C_k \tilde{M},$$

where $\tilde{M}$ is a $k \times n$ matrix that is to be determined by solving the following least squares problem

$$\tilde{M} = \arg\min_{M} \|A - C_k M\|_F^2. \tag{1}$$

Here the norm $\|\cdot\|_F$ is the Frobenius norm of a matrix. It is well-known that problem (1) has a closed-form solution, i.e.,

$$\tilde{M} = (C_k^T C_k)^{-1} C_k^T A.$$

2.2 Retrieval Procedure and Research Strategies

For convenience, we ignore the subscript of C_k and write the concept matrix as C.

Given a query vector q, the retrieval with the concept projection matrix (CPM) A_k can be computed as

$$r^T = q^T A_k = q^T C (C^T C)^{-1} C^T A, \tag{2}$$

where r^T is the ranking vector. After sorting the entries of r^T in a descending order, we have a ranking list of documents which are related to the query (listed from the most relevant to the least relevant). The retrieval accuracy based on CPM is comparable to the SVD technique [5]. However, the retrieval procedure in Eq. (2) is not efficient in terms of numerical computation and storage. It needs to compute $(C^T C)^{-1}$, the inverse of the normal matrix of the concept matrix, which is likely to be a dense matrix.

To make the numerical computation procedure and storage cost more competitive, a strategy which utilizes the knowledge of both preconditioning techniques and the computational theorems have been studied. The strategy is to compute a sparse matrix to approximate the projection matrix directly, i.e., we compute a matrix M to solve the least squares problem (1) approximately. The matrix M computed must be sparse in order for the strategy to be competitive against the inverse of the normal matrix of the concept projection matrix approach (CPM).

3 Approximate Least Squares Based Strategies

Note that all we want is to find a *sparse* matrix M such that the functional $f(M) = \|A - CM\|_F^2$ is minimized, or more precisely, *approximately minimized*. A similar problem in the preconditioning community is to compute a sparse approximate inverse matrix M for a nonsingular coefficient matrix A by minimizing $f(M) = \min_{M \in \mathcal{G}} \|I - AM\|_F^2$, subject to certain constraints on the sparsity pattern of M. Here the sparsity pattern of M is restricted to a (usually unknown *a priori*) subset $\mathcal{G}$.

So, for a term-document matrix A, we can minimize the functional

$$f(M) = \min_{M \in \mathcal{G}} \|A - CM\|_F^2 \tag{3}$$

with a constraint such that M is sparse. The most important part of this minimization procedure is to determine the sparsity pattern constraint set $\mathcal{G}$, which gives the sparsity pattern of M.

3.1 Static Sparsity Pattern (SSP)

The static sparsity pattern strategy use *a priori* patterns. In preconditioning field, a particularly useful and effective strategy is to use the sparsity pattern of the coefficient matrix C or C^T for M. The difficulty for choosing a static sparsity pattern in information retrieval lies in the fact that the dimensions of the matrices C and M do not match and there is no known study that has been done to find a suitable sparsity pattern, to the best of our knowledge. This research ventures into a non-traditional application of computational numerical linear algebra with approximate decomposition matrix computation in information retrieval.

A strategy based on the sparsity pattern of C and entry values of both C and A is described below.

- This strategy is based on the numerical computation, vector-vector product. That is $c_i m_j = a_{ij}$, where c_i is the ith row of C, m_j is the jth column of M. The sparsity pattern of m_j is given in this way: If a_{ij} is the largest entry in the jth column of A, the sparsity pattern of m_j is the same as that of c_i. Here we use small matrices to illustrate our ideas. Suppose the three matrices are $C_{4\times3}$, $M_{3\times5}$ and $A_{4\times5}$. The pattern of $CM = A$ is depicted by the Eq (4).

$$\begin{pmatrix} x & 0 & x \\ 0 & x & 0 \\ x & x & 0 \\ 0 & x & 0 \end{pmatrix}_{4\times3} \begin{pmatrix} - & - & - & - & - \\ - & - & - & - & - \\ - & - & - & - & - \end{pmatrix}_{3\times5} = \begin{pmatrix} 0 & x & 0 & x & 0 \\ 0 & 0 & 0 & x & x \\ x & x & 0 & 0 & 0 \\ x & 0 & x & 0 & 0 \end{pmatrix}_{4\times5} \tag{4}$$

Here, "x" denotes nonzero entry, "-" denotes undefined pattern. We determine the sparsity pattern of M column by column. First, find the largest entry in each column of A, suppose they are a_{31}, a_{12}, a_{43}, a_{14}, and a_{25} in Eq (4). Then the sparsity pattern of m_1, is the same as that of c_3 and the sparsity pattern of m_2 is the same as that of c_1, m_3 has same sparsity pattern of c_4 and m_4 is the same as that of c_1. Finally we have the sparsity pattern of M like this: $\begin{pmatrix} x & x & 0 & x & 0 \\ x & 0 & x & 0 & x \\ 0 & x & 0 & x & 0 \end{pmatrix}$.

Since there may be more than one largest entries in each column in term-document matrix A, the following rules may be applied to choosing the largest term.

  – Start the above procedure from the column of A that has the smallest number of nonzero entries.
  – Do not use the same row's sparsity pattern of C if possible.

Based on this strategy, the sparsity ratio of M is almost the same as that of C.

3.2 Dynamic Sparsity Pattern (DSP)

The dynamic sparsity pattern strategy first computes an approximate decomposition matrix by solving the least squares problem (3) with respect to an initial sparsity pattern guess. Then this sparsity pattern is updated according to some rules and is used as the new sparsity pattern guess for solving (3). The approximate decomposition matrix computation may be repeated several times until some stopping criteria are satisfied. Different update rules lead to different dynamic sparsity pattern strategies. One useful rule, suggested by Grote and Huckle [6] in computing sparse approximate inverse preconditioners, adds the candidate indices into the sparsity pattern of the current approximation S that can most effectively reduce the residual $g = C(\cdot, S)\bar{m}_j - a_j$. The candidate indices are chosen from the set

$$\beta = \{j \notin S |\ C(\alpha, j) \neq 0\},$$

where $\alpha = \{\ i \mid g(i) \neq 0\}$. This is a one-dimensional minimization problem

$$\min_{u_j} \|g + u_j(Ce_j - a_j)\|_2, \quad j \in \beta,$$

which e_j is the jth unit vector. Denote $l_j = Ce_j - a_j$, the above minimization problem has the solution

$$u_j = -\frac{g^T l_j}{\|l_j\|_2^2}.$$

For each j, we compute the 2-norm of the new residual as

$$\rho_j = \|g\|^2 - \frac{(g^T l_j)^2}{\|l_j\|_2^2}.$$

Then we can choose the most profitable indices j which lead to the smallest new residual norm ρ_j. The procedure to augment the sparsity structure S is as follows.

Algorithm **3.1** Construct a dynamic pattern approximate decomposition matrix.

```
1.  Given the maximum update steps $ns$, fill-ins $\mu > 0$,
    stopping tolerance $\epsilon$, and
    an initial diagonal sparsity pattern $S$
2.  Loop
3.      Compute $\rho_j$ for all indices $j \in \beta$
4.      Compute the mean $\lambda$ of $\{\rho_j\}$
5.      At most $\mu$ indices with $\rho_j < \lambda$
        will be added into $S$
6.  Until $||r||_2 < \epsilon$ or exceed the steps $ns$
```

Barnard *et al.*[1] released an MPI implementation, SPAI_3.0, for computing the sparse approximate inverse preconditioner of a nonsingular matrix. We modified the code for our computation of the sparse approximate decomposition matrix.

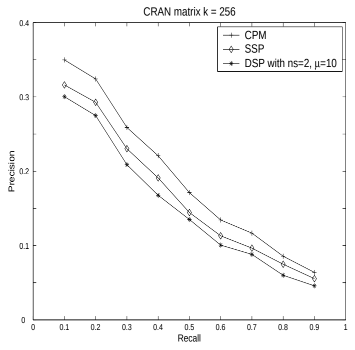
4 Numerical Experiments

To evaluate the performance of the proposed matrix approximation approaches SSP and DSP, we apply them to three popular text databases: CRAN,MED and CISI and compare them with the concept project matrix method (CPM). The information about the three databases are given in Table 1. A standard way to evaluate the performance of an information retrieval system is to compute precision and recall values. We average the precision of all queries at fixed recall values as $10\%, 20\%, \ldots, 90\%$. We also compare their storage costs and CPU time required for query procedure and approximate matrix computation. The precision computation and query time are carried out in UNIX system by using matlab. The sparse matrix computation and matrix inverse are carried out in IBM Power/Intel(Xeon) hybrid system at University of Kentucky by using C.

Table 1: The information of three databases

Database	Matrix size	Number of queries
CISI	5609×1460	112
CRAN	4612×1398	225
MED	5831×1033	30

The number of clusters k is chosen as follows: $k = 256$ and $k = 500$ are used for CISI and CRAN databases, $k = 200$ and $k = 500$ are for MED database. For DSP approach in algorithm 3.1, we choose $\epsilon = 1$ for all tests. ns and μ vary in different tests. Higher values of ns, μ lead to more work, more fill-in, and usually more accurate matrix.



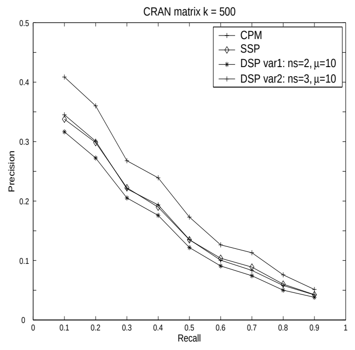


Figure 1: Precision-recall test for CRAN database

The query precision tests for all three databases are given in Fig. 1, Fig. 2, and Fig. 3. We are not surprised that CPM has better query results as CPM matrix is much more dense (we will show this in the later tests) and hence more accurate than that of SSP and DSP.

We then compare their storage costs required in previous tests by listing the number of nonzeros of the approximate matrix M and CPM matrix $(C^TC)^{-1}C^TA$. The test results presented in the left and right panel in Fig. 4 are corresponding to the upper panel and down panel in Fig. 1, Fig. 2, and Fig. 3 respectively. From this test we see that SSP and DSP use much less memory space than CPM. Considering the fact with about 90% less memory costs than CPM, SSP and DSP suffer only 6% precisions lost, they may be more attractive if storage cost is a bottle-neck. We also tried to sparsify the CPM matrix by dropping small entries. By reducing 20% of the memory storages, the precision is lost more than 40% for CISI with $k = 500$ test.

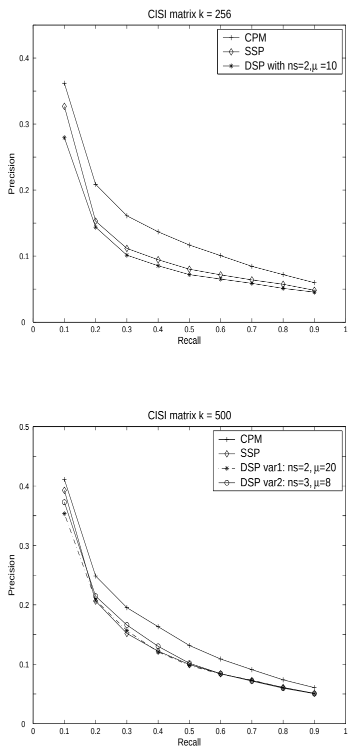


Figure 2: Precision-recall test for CISI database

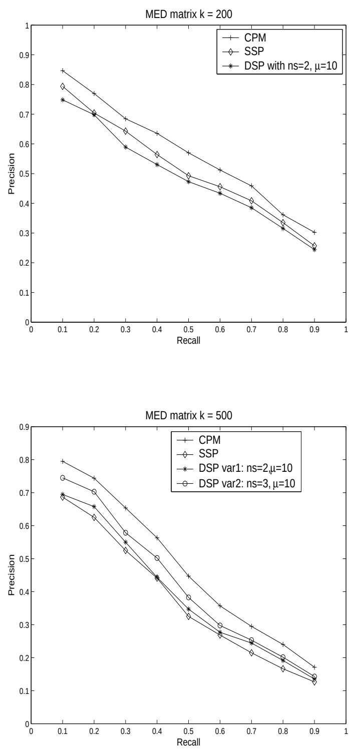


Figure 3: Precision-recall test for MED database

In order to see which sparsity pattern is good, we compare SSP and DSP by increasing the density of the DSP matrix in the tests of all three databases with $k = 500$. In the down panel of the Fig. 1, Fig. 2 and Fig. 3, the precision-recall curves corresponding to DSP var2 are better or very close to that of SSP while still uses smaller memory space compared to SSP. That means, given the same storage constraint, dynamic sparsity pattern strategy more accurately computes sparse approximate matrix than the static sparsity pattern strategy.

Finally we give the query time and total CPU time required for the query procedures and matrix computation. The query time is one of the most important aspects in information retrieval system. It affects the retrieval performance. We see from the upper panel of the Fig. 5 that SSP and DSP greatly reduce the query time required by CPM approach due to their very sparse matrices. If k is large, the new approaches also reduce total CPU time as the inverse of C^TC takes much time for CPM, see the down panel in Fig. 5.

5 Summary

We have developed numerical matrix computation methods based on static sparsity pattern (SSP) and dynamic sparsity pattern (DSP) to compute the approximate matrix to the concept matrix decomposition. We tested and compared with CPM based retrieval schema. In our numerical experiments, the sparsity pattern based approaches SSP and DSP turn out to be more competitive in terms of query precision, computational costs and memory space.

With the comparison of SSP and DSP sparsity pattern strategies, DSP displays some of the following advantages: First is that it computes a more accurate approximate matrix than SSP given the same density constraint. Second is that the accuracy of the approximate decomposition matrix can be controlled easily. By allowing more update steps, we can compute a more accurate approximate decomposition matrix. Third, as the information retrieval database may be updated periodically to accommodate new documents or to remove some out-of-dated

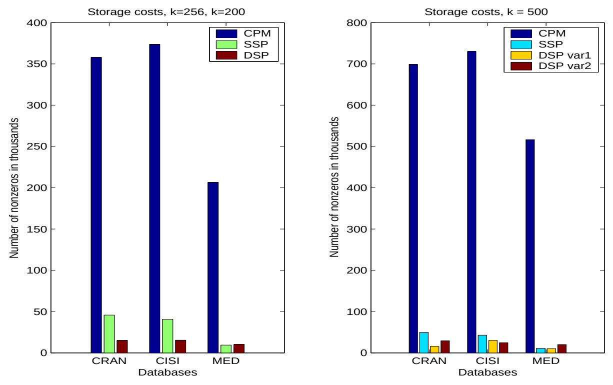


Figure 4: Storage cost comparison of three different retrieval methods.

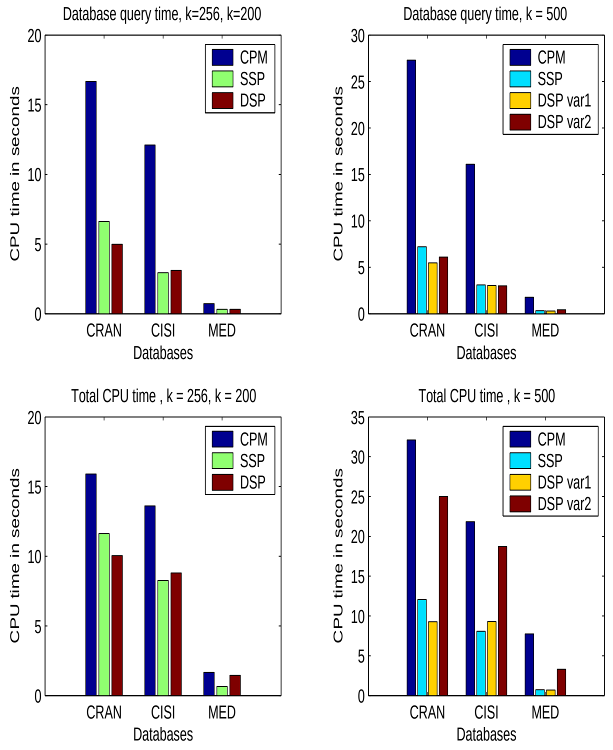


Figure 5: CPU time for query procedure and matrix construction

documents, the computed sparsity pattern may be used as a static sparsity pattern in some intermediate database update. But the dynamic sparsity pattern strategy is more expensive in terms of computational cost. Note that in information retrieval, these computations are called the *pre-processing* computation to prepare the database for the purpose of retrieval. Thus, an expensive one-time cost can be allowed if the prepared database enables more accurate and faster retrieval. If the storage costs and query time are the bottle-neck during the query procedure, the dynamic sparsity pattern strategy looks more attractive.

References

[1] S. T. Barnard, L. M. Bernardo, and H. D. Simon. An MPI implementation of the SPAI preconditioner on the T3E. *Int. J. High Performance Comput. Appl.*, 13:107–128, 1999.

[2] E. Chow. A priori sparsity patterns for parallel sparse approximate inverse preconditioners. *SIAM J. Sci. Comput.*, 21(5):1804–1822, 2000.

[3] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, 2001.

[4] J. Gao and J. Zhang. Text retrieval using sparsified concept decomposition matrix. Technical Report No. 412-04, Department of Computer Science, University of Kentucky, KY, 2004.

[5] J. Gao and J. Zhang. Clustered SVD strategies in latent semantic indexing. in *Information Processing and Management*, 41(5):1051-1063, 2005.

[6] M. Grote and T. Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM J. Sci. Comput.*, 18:838–853, 1997.

[7] J. Hartigan. *Clustering Algorithms*. John Wiley & Sons, New York, NY, 1975.

[8] J. Hartigan and M. Wong. Algorithm AS136: A k-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

[9] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surveys*, 31(3):264–323, 1999.