Multilevel Linear Dimensionality Reduction using Hypergraphs for Data Analysis[*]

Haw-ren Fang
Department of Computer Science and Engineering
University of Minnesota; Minneapolis, MN 55455
hrfang@cs.umn.edu

Yousef Saad
Department of Computer Science and Engineering
University of Minnesota; Minneapolis, MN 55455
saad@cs.umn.edu

ABSTRACT

Classical algorithms used for dimension reduction can be time-consuming when the data set is large. In this paper we consider a method based on hypergraph coarsening to find a smaller set of data representing a given data set, prior to performing the projection into the low-dimensional space. The cost of the dimensionality reduction process is reduced because of this hypergraph-based pre-processing step. In a hypergraph model, each data item is represented as a vertex and related data items are connected by a hyperedge, which is simply a subset of vertices. To coarsen the data, we use a method that merges pairs of vertices. In the multilevel framework, the coarsening is recursively repeated until a coarsened data set of a certain size is reached. Then we project the coarsened data into a lower dimensional space, using a known linear dimensionality reduction method. The same linear mapping from the coarsened data is then applied to the original data set for projecting data into low-dimensional space. As an application of this idea, we consider text mining. Experimental results indicate that the multilevel hypergraph technique proposed in this paper offer a very appealing cost to quality ratio.

Categories and Subject Descriptors

F.2.1 [**Numerical Algorithms and Problems**]: Computations on matrices; G.2.2 [**Graph Theory**]: Hypergraphs; H.3.3 [**Information Search and Retrieval**]: Retrieval models, Relevance feedback

General Terms

algorithms, performance

Keywords

Multilevel hypergraph coarsening, dimensionality reduction, latent semantic indexing

[*]This work was supported by NSF grants DMS 0510131 and DMS 0528492 and by the Minnesota Supercomputing Institute.

1. INTRODUCTION

Dimensionality reduction techniques appear in many fields, including data mining, machine learning, and computer vision. The goal of dimensionality reduction is to map the high dimensional samples into a lower dimensional space so that certain properties are preserved. When the number of data samples is large, existing methods, such as those based on Principal Component Analysis (PCA) can be prohibitively expensive.

A simple idea for reducing the cost of dimension reduction techniques is simply to select a smaller data set that is a good representative of the whole sample. Assume for a moment this can be done. This means we would, for example, replace the original data set $X = [x_1, \cdots, x_n] \in \mathbb{R}^{m\times n}$ by a subset $\hat{X} \in \mathbb{R}^{m\times k}$ of X, which without loss of generality we can assume to consist of the k first items of X, so $\hat{X} = [x_1, \cdots, x_k]$. Then, based on $\hat{X}$, we would find a projector from m-dimensional space to d-dimensional space, with $d \ll m$. The same projector can now be used to project any item in $\mathbb{R}^m$ to d-dimensional space.

The only remaining question is how to find a good representative subset of X. An appealing method when some adjacency graph of the data is available is to perform a succession of coarsening steps [9, 10, 17]. When the graph is not available then a K-nearest-neighbor graph can be built but this process is expensive.

However, there are instances such as in text mining when the original data itself is sparse. For such cases, there is a hypergraph that is canonically associated with the data. A coarsening step can be performed using this hypergraph. Hypergraphs are generalizations of graphs that allow edges, now called hyperedges, to connect more than two vertices.

The hypergraph model combined with a multilevel approach, using coarsening among other tools, has had a remarkable success for partitioning meshes and, generally, sparse data in scientific computing, see, e.g., [2, 3, 5, 8, 11, 16]. This technique has applications in many fields, including parallel sparse-matrix techniques (e.g., [2, 5, 16]), and VLSI design (e.g., [8, 11]). Motivated by this success, we explore dimensionality reduction techniques for data analysis, that exploit the multilevel hypergraph framework.

Formally, a hypergraph $H = (V, E)$ is defined as a set of vertices V and a set of hyperedges E, where each hyperedge is a subset of the vertex set V. The size of a hyperedge is the cardinality of this subset. (A hyperedge is also called a *net*.) A weighted hypergraph has non-negative numeric weights associated with each vertex, each hyperedge, or both. A hypergraph can be represented by a boolean matrix where each column represents a vertex, and each row represents a hyperedge which connects all vertices with a one in the row. When a data matrix is sparse, as is the case for a term-document matrix, the nonzero pattern defines a hypergraph in a canonical way. In this case hyperedges correspond to the rows and vertices correspond to the columns of the data matrix. In the particular example of term-document matrix, a hyperedge represents a relationship between some documents. Thus, a hyperedge which represents row i, is simply the subset of all documents containing term i in the data set under consideration.

We need a coarsening process which will preserve these relationships as best possible. The important underlying assumption here is that the information is very redundant and this redundancy should be exploited. In the simplest case, if two documents have the exact subset of terms, then one is enough to represent both. If a document x has a set of terms which includes the union of the terms of two documents then x will be enough to represent all 3 documents. As can be guessed from these examples, there is some form of dimension reduction taking place in the document space – one that is basic and considers only structure.

We will recursively compute a coarsened version of the original hypergraph, i.e., one with fewer vertices, of the data set using a method called maximal-weight matching, to merges pairs of vertices. (See, e.g., [5].) Then, we can apply any projective method of dimensionality reduction to the coarsened data at the lowest level. The resulting projector can be used to project all the original data or any new test data such as a new query to be processed.

One might argue that a scheme of this type does not necessarily achieve the most important goal of dimensionality reduction which is to remove noise and redundancies from the data. Recall that LSI determines a basis which represents the main features ('latent semantic') of a set of text documents and resolves common issues related to word usage, such as synonymy and polysemy. As was discussed above, hypergraph coarsening should achieve this goal partly – though it works on documents rather than terms which are processed by the more powerful technique of LSI on the resulting subset of documents. The experiments confirm this. In some cases the multilevel scheme even gives slightly better results than LSI.

The rest of this paper is organized as follows. Section 2 gives some background on the hypergraph model. Section 3 presents the multilevel dimensionality reduction methods based on hypergraph coarsening. Applications to text mining (information retrieval) are illustrated in Sections 4. A conclusion is given in Section 5.

2. THE HYPERGRAPH MODEL

A hypergraph $H = (V, E)$, consists of a set of vertices V and a set of hyperedges (nets) E. Each hyperedge is a non-empty subset of V; the size (cardinality) of this subset is called the *degree* of this hyperedge. Likewise, the *degree* of a vertex is the number of hyperedges which include it. Two vertices are called *neighbors* if there is a hyperedge connecting them, i.e., if they belong to at least one common hyperedge. Hypergraphs extend the classical notion of graphs. In a standard graph an edge connects two vertices, i.e., it is a set of two vertices, whereas a hyperedge may connect an arbitrary subset of vertices.

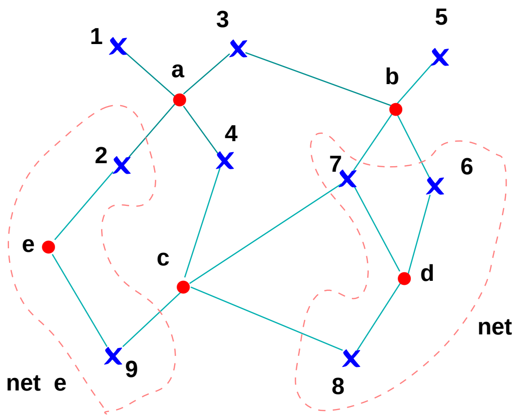

Figure 1: A sample hypergraph.

A hypergraph $H = (V, E)$ can be canonically represented by a boolean matrix A, where the vertices in V and hyperedges (nets) in E are represented by the columns and rows of A, respectively. Each hyperedge, a row of A, connects the vertices whose corresponding entries in that row are non-zero. An example is illustrated in Figure 1, where $V = \{1, \dots, 9\}$ and $E = \{a, \dots, e\}$ with $a = \{1, 2, 3, 4\}$, $b = \{3, 5, 6, 7\}$, $c = \{4, 7, 8, 9\}$, $d = \{6, 7, 8\}$, and $e = \{2, 9\}$. The boolean matrix representation of this hypergraph is

$$A = \begin{array}{ccccccccc|c} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & \\ \hline 1 & 1 & 1 & 1 & & & & & & a \\ & & 1 & & 1 & 1 & 1 & & & b \\ & & & 1 & & & 1 & 1 & 1 & c \\ & & & & & 1 & 1 & 1 & & d \\ & 1 & & & & & & & 1 & e \end{array}$$

For data sets represented by sparse matrices, such as for example, cases of term-document data sets, the adjacency matrix is precisely the matrix representation of the a hypergraph representing the relation "term i is contained in document j". Thus, hyperedge i (represented by row i), consists of all documents (columns/vertices) containing term i. For applications where the data matrix is dense, such as a matrix of vectorized face images, techniques such as wavelets decomposition can be applied to sparsify the data before using multilevel coarsening. This approach is currently under investigation.

2.1 Hypergraph coarsening

Consider a data set of n entries in $\mathbb{R}^m$ represented by a matrix $X \in \mathbb{R}^{m \times n}$, and a hypergraph $H = (V, E)$ with vertex set V corresponding to the columns of X. The hypergraph can be represented by a boolean matrix A, where the columns of A represent the vertices in V, and the rows of A represent the hyperedges in E.

Coarsening a hypergraph $H = (V, E)$ means finding a 'coarse' approximation $\hat{H} = (\hat{V}, \hat{E})$ to H with $|\hat{V}| < |V|$, which is a reduced representation of the original hypergraph H, in that it retains as much of the structure of the original hypergraph as possible. By recursively coarsening we obtain a succession of smaller hypergraphs which approximate the original graph. Several methods exist for coarsening hypergraphs, see [2, 11] for a discussion. The method used in this paper is based on merging pairs of vertices.

In order to select which pairs of vertices to merge in a hypergraph, we consider the *maximum-weight matching* problem (e.g., [2, 5]). Pairing two vertices is termed *matching*. The edge weight between two vertices is the number of hyperedges connecting them. For example, in Figure 1, the weight of the pair (6,7) is 2 because vertices 6 and 7 both belong to the hyperedges b and d and no other common hyperedge. On the other hand the pair $(5, 9)$ has a weight of zero. In the hyperedge-vertex matrix representation (a boolean matrix), the weight of the pair i, j (vertices) is the inner product of the two columns i and j as can be readily verified with the examples just given.

This inner-product weight is adopted as a similarity metric in two software packages for hypergraph partitioning, hMETIS [8] and Mondriaan [16]. The maximum-weight matching problem consists of finding a matching that maximizes the sum of edge weights of the vertex pairs. In practice, it is not necessary to find the optimal set of matching pairs, see e.g., [5], as sub-optimal greedy approaches yield satisfactory results. A greedy Algorithm for maximum-weight matching will be used in the experiments. The vertices can be visited in a random order, or in the order in which the data items are listed. For each unmatched vertex v, all the unmatched neighbor vertices u are considered, and the inner product between v and each u is computed.

The vertex with the highest non-zero inner product is matched with u and the procedure is repeated until all vertices have been matched. The computed matching is a coarse representation of the hypergraph, with the coarse hyperedges inherited from the fine graph. More precisely, the coarse vertex set consists of matched fine vertex pairs. A fine vertex pair is in a coarse hyperedge if any of the two vertices is in the corresponding fine hyperedge. It is convenient to present the hypergraph coarsening procedure in matrix form. The pseudo-code is given in Algorithm 1.

Three remarks on Algorithm 1 must be made. First, if X is a boolean matrix, then the loop (*) results in ip[k] being the inner product of column j and column k of X. Second, it is possible that vertex j does not have any unmatched neighbor and the algorithm will branch to (**). However, this is rare in practice, since a hyperedge can connect multiple vertices and vertex j almost always ends up finding an unmatched neighbor, unless X is too sparse. Third, the columns of $\hat{X}$ are the sums of matched pairs. This property is particularly good for applications with sparse data matrices which are

Algorithm 1 Hypergraph coarsening by maximum-edge matching.

```
{Coarsen a hypergraph represented by the sparse pattern
    of matrix $X$ with $n$ columns. }
{The $n$ vertices are indexed by $1, \dots, n$.}
$S \leftarrow \{1, \dots, n\}$                    ▷ Set of unmatched vertices
$p \leftarrow 0$                                  ▷ Number of vertex pairs
repeat
    $p \leftarrow p + 1$
    Randomly pick $j \in S$; $S \leftarrow S - \{j\}$
    Set ip[$k$] $\leftarrow 0$ for $k = 1, \dots, n$.        ▷ Inner products
    for all $i$ with $a_{ij} \neq 0$ do
        for all $k$ with $a_{ik} \neq 0$ do
            ip[$k$] $\leftarrow$ ip[$k$] + 1                  ▷ (*)
        end for
    end for
    $i \leftarrow$ argmax{ip[$k$] : $k \in S$}
    if ip[$i$] = 0 then              ▷ Vertex $j$ is isolated from
        unmatched vertices.
        $\hat{X}(:, p) \leftarrow X(:, j)$                    ▷ (**)
    else         ▷ Vertex $i$ matches vertex $j$ as its nearest
        unmatched neighbor.
        $\hat{X}(:, p) \leftarrow X(:, i) + X(:, j)$
        $S \leftarrow S - \{i\}$
    end if
until $S = \emptyset$
{The sparsity pattern of $\hat{X}$ corresponds to the coarsened
    graph of $X$.}
```

directly associated with hypergraphs. In such cases we may simply input to Algorithm 1 the sparse data matrix itself.

By recursively coarsening the graph we obtain a sequence of sparse matrix $X_1, X_2, \dots, X_r$, where X_k corresponds to the coarse graph H_k of level k for $k = 1, \dots, r$, and X_r represents the lowest level graph H_r.

3. MULTILEVEL DIMENSIONALITY REDUCTION

The objective of dimensionality reduction is to map the data in the high dimensional space into a low dimensional one such that certain properties are preserved. More precisely, given a matrix $X \in \mathbb{R}^{m \times n}$ whose columns correspond to the vertices V and whose rows correspond to the hyperedges E in a hypergraph $H = (V, E)$, produce $Y \in \mathbb{R}^{d \times n}$ $(d < m)$ such that Y preserves certain features of X.

In the multilevel framework of hypergraph coarsening we apply a linear dimensionality reduction method to the coarsened data matrix $X_r \in \mathbb{R}^{m \times n_r}$ at the lowest level (rth level) and obtain $Y_r \in \mathbb{R}^{d \times n_r}$ $(d < m)$, where n_r is the number of data items at coarse level r $(n_r < n)$. The linear mapping, denoted by P, is then applied to the original data $X \in \mathbb{R}^{m \times n}$ to obtain a reduced representation $Y = PX \in \mathbb{R}^{d \times n}$ $(d < m)$ of the original data set. The procedure is illustrated in Figure 2. The same linear mapping can also be applied to any 'out-of-sample' test data. For example, in LSI, the same projector is applied to a query ('pseudo-document') and to the document set before a comparison is made.

The procedure just described provides a hypergraph which,

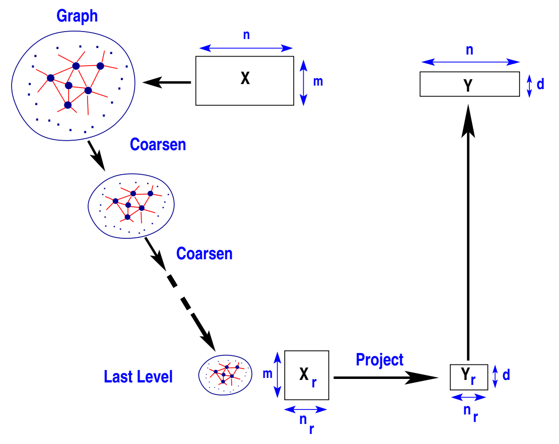

Figure 2: A sketch of the multilevel reduction

it is hoped, is a good representation of the original graph. However, it does not tell us much about the resulting reduced set X_r related to the original data matrix X. In a coarsened hypergraph H_k, each vertex represents a subset of vertices from the original hypergraph H_1. A data item (a column in X_k) corresponding to a vertex in a coarsened hypergraph is the sum of all original data items in X (multiple columns in X) corresponding to this subset of vertices with which it is associated.

4. APPLICATION TO TEXT MINING

Latent Semantic Indexing (LSI) [4] is a well-established framework for conceptual information retrieval [1, 6]. In this section we compare the retrieval performance of LSI and multilevel-LSI. The latter is based on the algorithm just described in this paper.

In the vector space model, a collection of n documents indexed by m terms is represented by a sparse term-document matrix $X \in \mathbb{R}^{m\times n}$. The rows and columns of X are called term vectors and document vectors, respectively. The (i, j) entry of X, denoted by x_{ij}, is the number of occurrences of term i in document j, called *term frequency*. This matrix is canonically associated with a hypergraph $H = (V, E)$, where the vertices V correspond to the term vectors and the hyperedges E correspond to the document vectors.

A term-document matrix $X \in \mathbb{R}^{m\times n}$ is usually scaled before its usage. In the experiments we adopted TF-IDF (term frequency-inverse document frequency) scaling [15]. The *inverse document frequency* is defined by

$$z_i = \log(n/|\{j : x_{ij} > 0\}|), \tag{1}$$

where $|\{j : x_{ij} > 0\}|$ is the number of documents with term i occurring in them. A TF-IDF entry is the multiplication of TF entry and IDF entry, $\tilde{x}_{ij} = x_{ij}z_i$. The TF-IDF scaled matrix $\bar{X}$ is obtained by normalizing the columns to be unit vectors. In other words, the (i, j) entry of $\bar{X}$ is $\bar{x}_{ij} = \tilde{x}_{ij}/\sqrt{\sum_{k=1}^{m}\tilde{x}_{kj}^2}$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$.

Given a query $q \in \mathbb{R}^m$ (an array of term frequencies), *query matching* is the process to find the relevant documents. A query is also called a *pseudo-document* vector. Before the matching process, a query is also TF-IDF scaled. The scaled vector is denoted by $\bar{q} \in \mathbb{R}^m$. Note that however, the inverse document frequencies (IDF) here, defined in (1), are from the term-document matrix X.

The vector space model measures the similarity of two vectors by the cosine distance (i.e., the cosine of the acute angle between them). The full vector space model potentially contains noise and redundancies, that affect the retrieval performance. Instead, LSI approximates a given term-document matrix by its truncated SVD, denoted by $\bar{X} = [\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n] \approx U_d\Sigma_dV_d^T$, where $d \in \mathbb{R}$ is a certain desired rank. With reduced noise, a lower dimensional approximation of X helps discover the underlying latent semantic structure of the data. The columns of $V_d^T = [\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n] \in \mathbb{R}^{d\times n}$ are used as reduced representations of document vectors $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n \in \mathbb{R}^m$. Likewise, the rows of $U_d \in \mathbb{R}^{m\times d}$ are the reduced term vectors. Given a query $\bar{q} \in \mathbb{R}^m$, it is transformed to a reduced representation $\hat{q} = \Sigma_d^{-1}U_d^T\bar{q} \in \mathbb{R}^d$ in d-dimensional space. Document x_i is considered relevant to the query q if the cosine distance between their reduced representations $\langle\hat{q}, \hat{x}_i\rangle/\|\hat{q}\|\|\hat{x}_i\|$ is larger than some pre-defined threshold.

When a relevance vector (a boolean string of size n) is provided, the *precision* and *recall* are defined by

$$\text{Precision: } \frac{D_R}{D_T}, \qquad \text{Recall: } \frac{D_R}{N_R}, \tag{2}$$

where D_R, D_T, and N_R are the number of relevant documents retrieved by the process, the total number of documents in the collection, and the total number of relevant documents in the collection, respectively.

When the term-document matrix X is large, the computation of the SVD factorization can be expensive. A smaller set of document vectors can be obtained by the multilevel techniques described in Section 3. We denote it by $X_r \in \mathbb{R}^{m\times n_r}$, which represents the original $X \in \mathbb{R}^{m\times n}$ ($n_r < n$). The TF-IDF scaling is then applied to X_r, resulting in $\bar{X}_r$. Like the standard LSI, we compute the truncated SVD of $\bar{X}_r \approx U_d\Sigma_dV_d^T$, where d is the rank. We apply the same mapping to X and obtain a reduced representation $\Sigma_d^{-1}U_d^T\bar{X} = [\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n] \in \mathbb{R}^{d\times n}$. Note that we have applied TF-IDF scaling to the term-document matrix X, but here the inverse document frequencies (IDF), defined in (1), use the coarsened matrix X_r. Each query $q \in \mathbb{R}^m$ is also scaled in the same way to be $\bar{q}$, and then transformed to $\hat{q} = \Sigma_d^{-1}U_d^T\bar{q} \in \mathbb{R}^d$. The similarity of q and x_i are measured by the cosine distance between $\hat{q}$ and $\hat{x}_i$ for $i = 1, \ldots, n$. We call the resulting scheme multilevel-LSI.

The precision and recall defined in (2) depend on the tolerance of the similarity scores in the cosine distance measure. The performance evaluation may differ when the tolerance is different. Therefore, we use the average precision [7] to assess the retrieval performance. Sorting the similarity scores of query q to documents $x_1, \ldots, x_n$, we consider for $i = 1, \ldots, n$ the first i documents with the highest scores and obtain the precision and recall

$$P_i = R_i/i, \qquad R_i = R_i/R_n,$$

where R_i is the number of relevant documents among the first i documents. The *average precision* is defined by the mean of the interpolated precision

$$\bar{P} = \frac{1}{n}\sum_{i=0}^{n-1} \hat{P}\left(\frac{i}{n-1}\right), \qquad \hat{P}(x) = \max\{P_i : x \le R_i\}.$$

Three public data sets were used in our experiments: `Medline`, `Cran` and `NPL`[1]. The characteristics of these sets, such as numbers of documents, terms, and queries are listed in Table 1.

Table 1: Characteristics of the test sets.

Data set	`Medline`	`Cran`	`NPL`
# documents	1033	1398	11429
# terms	7014	3763	7491
sparsity (%)	0.74%	1.41%	0.27%
# queries	30	225	93
avg. # rel./query	23.2	8.2	22.4

The experiments were performed in sequential mode on a PC equipped with two Intel(R) Core(TM)2 @ 2.40GHz processors, using our Matlab implementation. In all tests we coarsened the data down to four levels. Compared with LSI, multilevel-LSI requires additional work to process the hypergraph coarsening. However, it saves time when computing the truncated SVD of the coarsened (smaller) term-document matrix. The CPU time used for coarsening `Medline`, `Cran`, `NPL` data sets is shown in the second columns of Tables 2, 3, and 4, respectively. The savings on SVD computation are much more significant.

Note that the average precision depends on the dimension used. We call the dimension that maximizes the average precision being *optimal*.

The experimental result using the `Medline` data set is now discussed. Figure 3 is the resulting plot of average precisions using various dimensions for SVD (ranks of truncated SVD). The number of documents, the optimal dimensions, and the average precision at all levels are displayed in Table 2. Using the optimal dimensions we obtain the precision-recall plot in Figure 4. Figure 5 shows the savings in CPU time gained by multilevel-LSI for computing truncated SVD.

Table 2: Statistics of `Medline` data set.

Level	coarsen. time	# doc.	optimal # dim.	optimal avg. precision
#1	N/A	1033	30	71.6%
#2	0.32	517	28	72.7%
#3	0.13	259	30	71.5%
#4	0.07	130	27	67.5%

Figure 6 is a plot showing the average precisions using various dimensions for SVD (ranks of truncated SVD) for the `Cran` data set. Table 3 lists the number of documents, optimal dimensions, and average precision at all levels. Using the optimal dimensions we obtain the precision-recall plot in Figure 7. The savings in CPU time gained by multilevel-LSI for computing the truncated SVD are shown in Figure 8.

[1]`ftp://ftp.cs.cornell.edu/pub/smart`

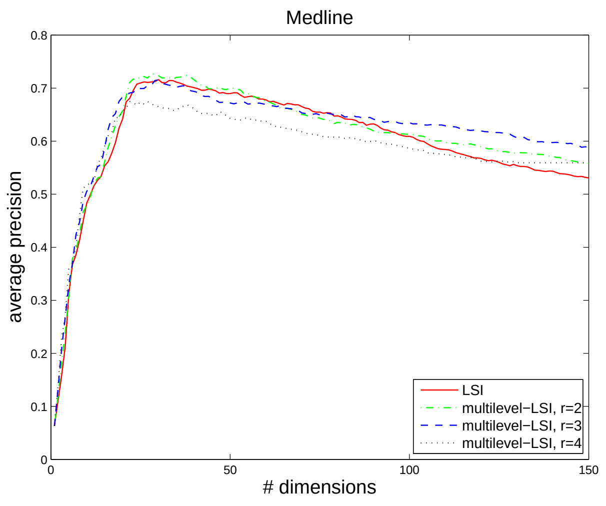

Figure 3: Average precision using the `Medline` set.

Table 3: Statistics of `Cran` data set.

Level	coarsen. time	# doc.	optimal # dim.	optimal avg. precision
#1	N/A	1398	95	39.8%
#2	0.95	699	101	40.6%
#3	0.34	350	92	40.6%
#4	0.18	175	76	37.5%

For the `NPL` data set, rather than $V_d^T \in \mathbb{R}^{d\times n}$, we used columns of $\Sigma_d V_d^T \in \mathbb{R}^{d\times n}$ as the reduced document vectors. Recall that the truncated SVD of the term-document matrix is denoted by $U_d \Sigma_d V_d \in \mathbb{R}^{m\times n}$, where d is the rank. Therefore, the reduced representation of a query $q \in \mathbb{R}^m$ is $\hat{q} = U_d^T q \in \mathbb{R}^d$. This adaption significantly improves the results of LSI and multilevel-LSI using the `NPL` data set.

Figure 9 is the resulting plot of average precisions using various dimensions for SVD (ranks of truncated SVD) for the set `NPL`. In addition to the savings of the computation time, the multilevel-LSI usually outperformed LSI for 800 or less dimensions. The number of documents, optimal dimensions and average precision at all levels are displayed in Table 4. Using the optimal dimensions we obtain the precision-recall plot in Figure 10. The savings in CPU time gained by multilevel-LSI for computing the truncated SVD are shown in Figure 11.

Table 4: Statistics of `NPL` data set.

Level	coarsen. time	# doc.	optimal # dim.	optimal avg. precision
#1	N/A	11429	736	23.5%
#2	3.68	5717	592	23.8%
#3	2.19	2861	516	23.9%
#4	1.50	1434	533	23.3%

The results are summarized as follows. Using the `Cran` data set, multilevel-LSI achieved similar performance of LSI, but reduced the SVD computation. For the `Medline` data set, multilevel-LSI slightly outperformed LSI, in addition to the

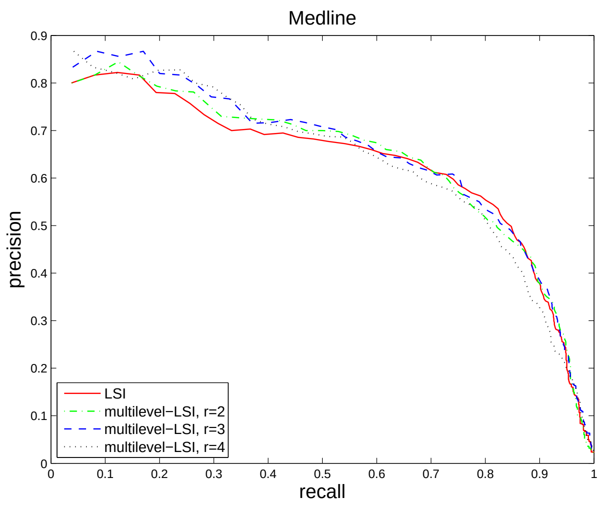

Figure 4: Precision-recall plot using the `Medline` set.

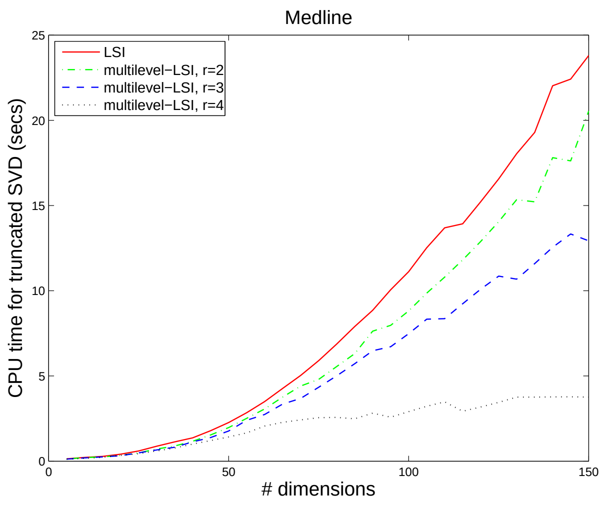

Figure 5: CPU time for truncated SVD using the `Medline` data set.

CPU time savings on SVD. For the `NPL` data set, the recall-precision plots of LSI and multilevel-LSI are comparable, using the optimal dimensions. However, multilevel-LSI performed better than LSI while fewer dimensions were used. This issue is important for large data sets since the SVD computation can be very expensive.

We also compared the hypergraph-based multilevel techniques presented in this paper with the kNN-graph-based multilevel schemes in [14]. Using the `Medline` and `Cran` data sets and the same number of levels, the hypergraph-based method slightly outperformed the kNN-graph-based one. In addition, a hypergraph is canonically associated with a sparse term-document matrix. While using the kNN-graph-based method, additional computation is required to construct a kNN graph. This issue is important for large data sets since the kNN graph construction can be prohibitively expensive. We conclude that the hypergraph-based multilevel techniques are more adequate than the kNN-graph-based multilevel schemes for text information retrieval.

Relevance feedback is a common technique in text information retrieval. The assumption is that we know in advance that some document vectors are related to a query. Then the query is added by the sum of these related documents, followed by a standard text mining procedure. More precisely, a query q is replaced by $q + b^T X$, where b is the boolean column vector indicating which documents are known *a priori* related to query q.

We tested relevance feedback for the multilevel framework. The experiments used the vector b defined above as the relevance vector, assuming that the exact information is available. The resulting average precision plots on `Medline`, `Cran` and `NPL` data sets are given in Figures 12, 13, and 14, respectively. These show that with relevance feedback, the multilevel-LSI still worked nicely, but not as good as that of LSI.

Note that LSI and multilevel-LSI we used in our experiments are SVD-based. However, the multilevel hypergraph framework we have proposed does not rely on the SVD computation. Indeed, we can incorporate other matrix approximation methods, such as semi-discrete decomposition (SDD) [12] and non-negative matrix factorization (NMF) [13], into the multilevel framework, resulting in multilevel SDD-based and NMF-based LSI for text information retrieval.

5. CONCLUSION

A multilevel framework was presented to perform dimensionality reduction in situations when the data sets are sparse. In applications with sparse matrix data sets, the hypergraph model can be directly applied since the pattern of non-zero entries of the sparse matrix yields a hypergraph. To coarsen the data, we use a method, called maximal-weight matching, which merges pairs of vertices. Dimensionality reduction is performed on the data at the lowest (coarsest) level with a linear projection method. The resulting projector is then applied to the original data. The method is illustrated with applications in text mining generally showing a good quality of the results at reduced cost.

Acknowledgments

We would like to thank Sofia Sakellaridi for a literature survey on multilevel hypergraph partitioning, and Jie Chen who gathered and processed the data sets used in this paper.

6. REFERENCES

[1] M. W. Berry and M. Browne. *Understanding Search Engines*. SIAM, 1999.

[2] U. V. Catalyurek and C. Aykanat. Hypergraph-partitioning based decomposition for parallel sparse-matrix vector multiplication. *IEEE Transaction on Parallel and Distributed Systems*, 10(7):673–693, 1999.

[3] U. V. Catalyurek and C. Aykanat. PaToH: a multilevel hypergraph partitioning tool. version 3.0. Technical report, Bilkent University, Department of Computer

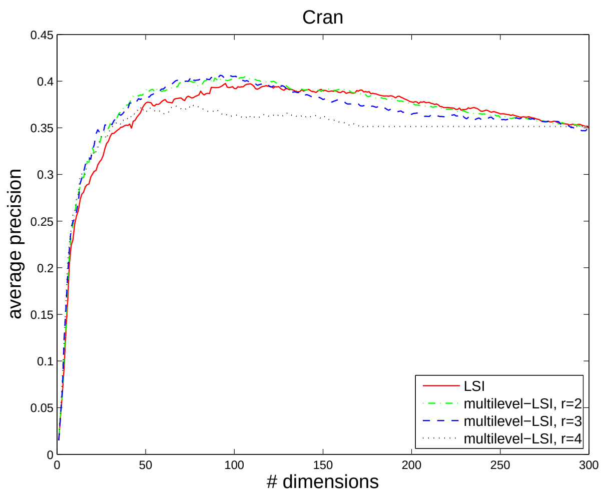

Figure 6: Average precision using the `Cran` set.

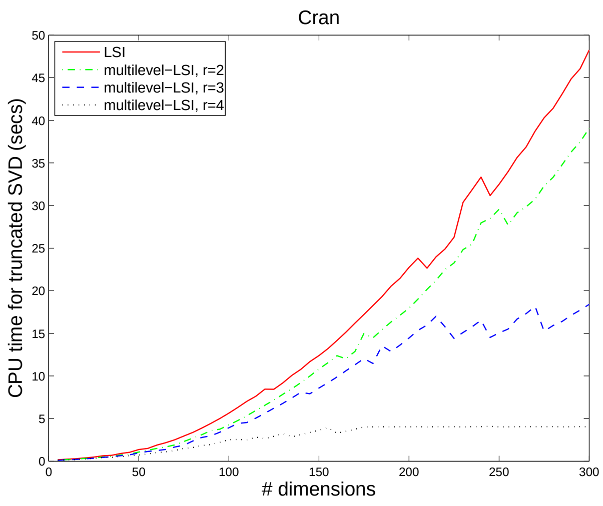

Figure 8: CPU time for truncated SVD using the `Cran` data set.

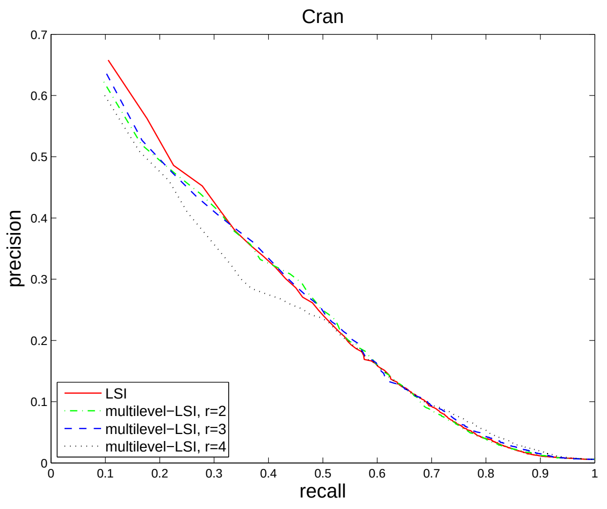

Figure 7: Precision-recall plot using the `Cran` set.

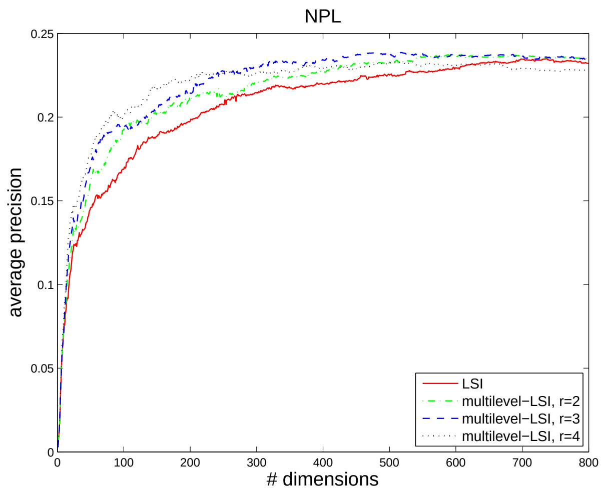

Figure 9: Average precision using the `NPL` set.

Engineering, Ankara, Turkey, 1999. PaToH is available from http://bmi.osu.edu/ umit/software.html.

[4] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *J. Soc. Inf. Sci.*, 41:391–407, 1990.

[5] K. Devine, E. G. Boman, R. Heaphy, R. Bisseling, and U. V. Catalyurek. Parallel hypergraph partitioning for scientific computing. In *20th International Parallel and Distributed Processing Symposium (IPDPS)*, page 10, 2006.

[6] L. Eldén. *Understanding Search Engines*. SIAM, 2007.

[7] D. K. Harman, editor. *The 3rd Text Retrieval Conference (TREC-3)*. NIST Special Publication 500-255, 1995. http://trec.nist.gov/.

[8] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: application in VLSI domain. In *34th Design Automation Conference*, pages 526–529, 1997.

[9] G. Karypis and V. Kumar. Analysis of multilevel graph partitioning. In *Supercomputing '95: Proceedings of the 1995 ACM/IEEE conference on Supercomputing (CDROM)*, New York, NY, USA, 1995. ACM Press. Article No. 29.

[10] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *J. Parallel Distrib. Comput.*, 48(1):96–129, 1998.

[11] G. Karypis and V. Kumar. Multilevel k-way hypergraph partitioning. *VLSI Design*, 11(3):285–300, 2000.

[12] T. G. Kolda and D. P. O'Leary. A semi-discrete matrix decomposition for latent semantic indexing in information retrieval. *ACM Trans. Information Systems*, 16:322–346, 1998.

[13] C.-J. Lin. Projected gradient methods for non-negative matrix factorization. *Neural*

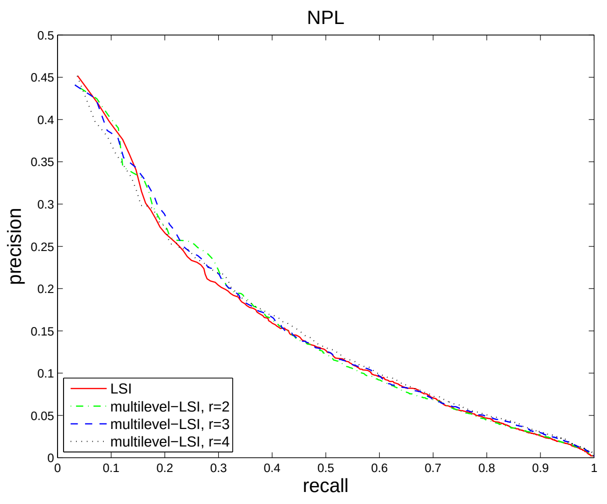

Figure 10: Precision-recall plot using the `NPL` set.

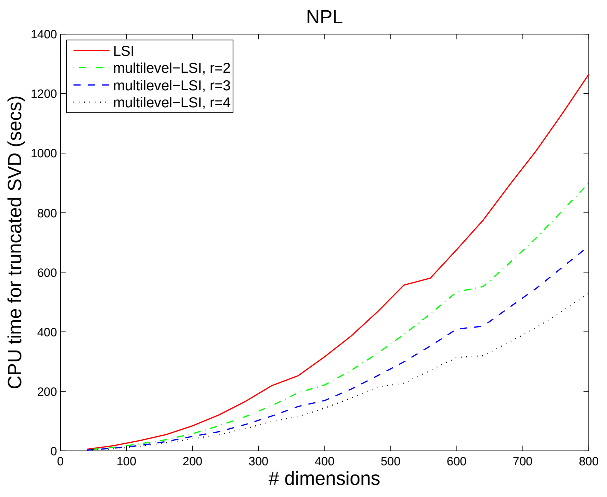

Figure 11: CPU time for truncated SVD using the `NPL` data set.

Computation, 19:2756–2779, 2007.

[14] S. Sakellaridi, H. r. Fang, and Y. Saad. Multilevel linear dimensionality reduction for data analysis using nearest-neighbor graphs. Submitted to KDD2008 conference.

[15] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523, 1988.

[16] B. Vastenhouw and R. H. Bisseling. A two-dimensional data distribution method for parallel sparse matrix-vector multiplication. *SIAM review*, 47(1):67–95, 2005.

[17] F. Wang and C. Zhang. Fast multilevel transduction on graphs. In *The 7th SIAM Conference on Data Mining (SDM)*, 2007.

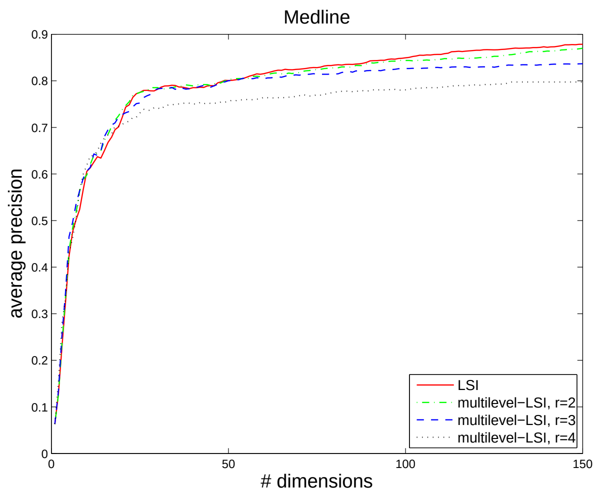

Figure 12: Precision-recall plot with relevance feedback using the `Medline` data set.

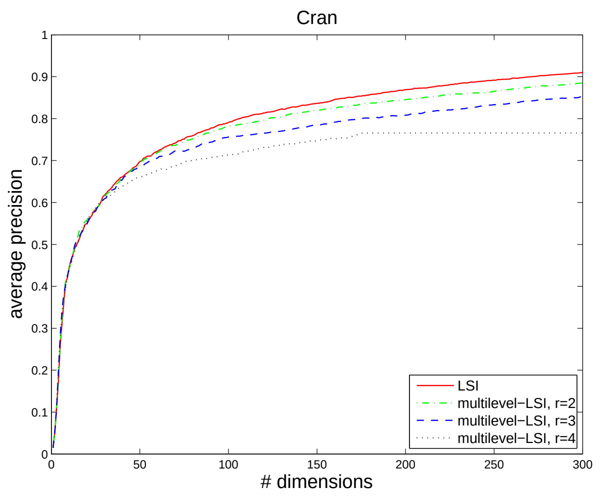

Figure 13: Precision-recall plot with relevance feedback using the `Medline` data set.

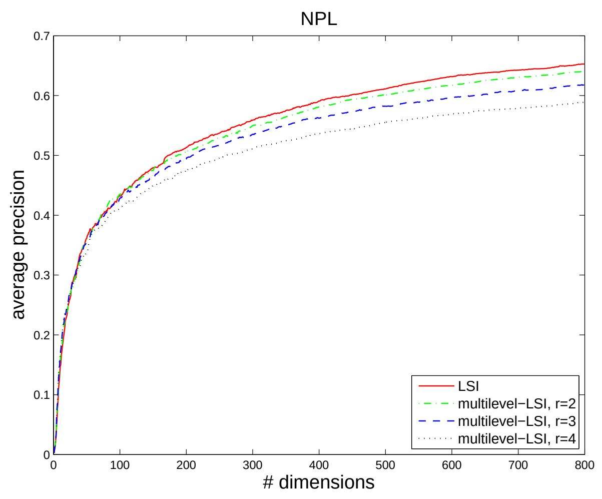

Figure 14: Precision-recall plot with relevance feedback using the `NPL` data set.