Latent Semantic Analysis for Multiple-Type Interrelated Data Objects

Xuanhui Wang[†], Jian-Tao Sun[‡], Zheng Chen[‡], ChengXiang Zhai[†]

[†]Department of Computer Science,
University of Illinois at Urbana-Champaign
{xwang20, czhai}@cs.uiuc.edu

[‡]Microsoft Research Asia, Beijing, P.R.China
{jtsun, zhengc}@microsoft.com

ABSTRACT

Co-occurrence data is quite common in many real applications. Latent Semantic Analysis (LSA) has been successfully used to identify semantic relations in such data. However, LSA can only handle a single co-occurrence relationship between two types of objects. In practical applications, there are many cases where multiple types of objects exist and any pair of these objects could have a pairwise co-occurrence relation. All these co-occurrence relations can be exploited to alleviate data sparseness or to represent objects more meaningfully. In this paper, we propose a novel algorithm, *M-LSA*, which conducts latent semantic analysis by incorporating all pairwise co-occurrences among multiple types of objects. Based on the mutual reinforcement principle, M-LSA identifies the most salient concepts among the co-occurrence data and represents all the objects in a unified semantic space. M-LSA is general and we show that several variants of LSA are special cases of our algorithm. Experiment results show that M-LSA outperforms LSA on multiple applications, including collaborative filtering, text clustering, and text categorization.

Categories and Subject Descriptors: H.3.3 [Information Search and Retrieval:] Indexing methods

General Terms: Algorithms

Keywords: M-LSA, LSA, mutual reinforcement principle, multiple-type

1. INTRODUCTION

Co-occurrence data arises naturally and frequently in a variety of applications such as information retrieval and text mining. In most existing work on analysis of co-occurrence data, only a single pairwise co-occurrence relationship between *two* types of objects is considered. For example, in information retrieval, the co-occurrence information between documents and words is used to rank documents with respect to queries [2]. In collaborative filtering, items are recommended to an active user based on historical co-occurrence data between users and items [14].

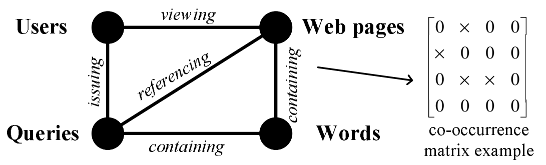
Users
viewing
Web pages
issuing
referencing
containing
Queries
containing
Words
$\begin{bmatrix} 0 & \times & 0 & 0 \\ \times & 0 & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$
co-occurrence matrix example

Figure 1: Example of multiple-type interrelated data objects. Each edge denotes a single co-occurrence relationship.

However, in most applications, there exist multiple types of data objects and each pair of them could have a pairwise co-occurrence relationship. For example, in the Web domain as shown in Figure 1, *users* co-occur with *Web pages* by *viewing*, *queries* co-occur with *Web pages* by *referencing*, *Web pages* co-occur with *words* by *containing*, and so on. With each kind of objects containing thousands of instances, each single co-occurrence relationship could be quite sparse. In the example above, using a single relationship, say, <user, Web page>, to represent users may not be meaningful since there are millions of Web pages and each user may only view a tiny portion of them. Latent Semantic Analysis (LSA) [10] was proposed to alleviate the data sparseness problem by representing objects in a low-dimensional semantic space. In this space, semantically related objects are expected to be near to each other. Such a dimension reduction technique has been shown to improve performance in a variety of applications (e.g., [10, 13, 4]). However, the application of LSA is rather limited since it can only consider the co-occurrence relationship between *two* types of objects. With multiple co-occurrence relationships available, it is beneficial to exploit all of them to identify the semantic relations and alleviate the data sparseness problem. In the example above, we can also exploit other relations such as <user, query> and <query, Web page> to better represent users, since users with similar interests tend to issue similar queries, and similar queries could refer to similar Web pages. All these co-occurrence relations could be complementary, thus it is desirable to incorporate all of them so as to represent each type of objects more meaningfully.

Though promising, exploiting the co-occurrence relations among multiple types of objects is challenging: 1) It is not clear how to effectively utilize all the co-occurrence relations among *multiple* types of objects to overcome data sparseness. 2) There may exist hidden relations between any two types of objects and these relations are complex since the information can also propagate through any co-occurrence path. Take Figure 1 as an example. There is no direct co-occurrence between users and words. But similar words can induce similar queries and Web pages, thus in turn, similar users. The similarity between words can be propagated to users through the path "words $\rightarrow$ queries $\rightarrow$ users" or "words $\rightarrow$ Web pages $\rightarrow$ users". Even more complex propagations are also possible.

To effectively utilize all the information among heterogeneous objects, we propose a novel and unified latent semantic analysis algorithm, M-LSA, to model all the objects in a unified framework and identify the *latent semantic* relations underneath *all* the co-occurrence data. By exploiting all the pairwise co-occurrence data simultaneously, M-LSA identifies the most salient or important concepts among them. These concepts span a unified low-dimensional semantic space, where each object is represented by a vector which reflects the strengths of its association with these concepts.

Specifically, to identify important concepts, a natural belief is that important concepts are related to important objects. Based on this assumption, we utilize the *mutual reinforcement principle*, which is underlying the traditional LSA, to identify the important objects in each type leveraging all the co-occurrence relations quantitatively. This principle leads to an eigenvector problem and the obtained eigenvectors are regarded as the latent concepts. We show that the M-LSA algorithm is a natural generalization of the traditional LSA from two to multiple types of objects.

The rest of this paper is organized as follows. Section 2 is to discuss previous work. We define our problem in Section 3. To solve this problem, we identify the *mutual reinforcement principle* underlying LSA and extend it to multiple types of objects in Section 4. This principle naturally leads to a solution to our problem: M-LSA. Section 5 is to present our experimental results for different applications. Finally, we conclude our paper in Section 6.

2. RELATED WORK

LSA was first introduced to address the *synonym* and *polysemy* problems in information retrieval [10]. Since then, LSA has attracted much attention and several researches analyzed it theoretically [3, 11, 21, 4]. For example, [11] and [21] used probabilistic model to study the effectiveness of LSA. More recently, [4] argued that spectral algorithms such as LSA can expand the documents implicitly to improve retrieval accuracy.

Some variants of LSA have also been proposed recently. Probabilistic LSA (PLSA) [15] applies a probabilistic *aspect model* to the co-occurrence data. Iterative Residual Rescaling (IRR) [1] is proposed to counteract LSA's tendency to ignore the minor-class documents. Unlike LSA, Nonnegative Matrix Factorization (NMF) [19, 25] decomposes the <document, word> matrix into two matrices with no negative values.

Most work above only considers co-occurrence relations between two types of objects. High-order co-occurrence data or high-order tensor is studied in multilinear algebra [18]. In [18], the High-Order Singular Value Decomposition (HOSVD) is proposed to factor high-order tensors; in contrast, we consider the *pairwise* co-occurrence relations between different types of objects in this paper, which is less computationally expensive than HOSVD.

Several recent studies utilize pairwise co-occurrence data for different specific purposes such as object clustering [5] and similarity measuring [24, 16]. In [9], a unified approach is proposed to analyze both link and text information. Compared with these studies, our approach is more general and fundamental in that we provide a general principled method for analyzing any multiple types of objects.

Our work is related to the HITS algorithm [17] and a key-phrase extraction algorithm [27] in the sense of sharing the *mutual reinforcement principle*. HITS uses this principle to find good pages from a Web subgraph and [27] uses it to identify salient key-phrases from a document. In this paper, we use this principle for multiple-type latent semantic analysis.

3. THE PROBLEM

The problem we study is to analyze the co-occurrence relationship among multiple types of objects. Suppose we have N types of objects $\{X_1, X_2, ..., X_N\}$ and each pair of them could have a pairwise co-occurrence relation. Formally, we construct an *undirected* graph $G(V, E)$. V consists of N vertices with each corresponding to a type of objects. If there is a pairwise co-occurrence relation between two types of objects, we have an edge in E which connects the corresponding vertices. We name graph G as "multiple-type graph". In G, each type corresponds to a set of objects and we use $|X_i|$ to denote the number of objects of this type. For each edge $e_{ij} \in E$, we have a $|X_i| \times |X_j|$ co-occurrence matrix M_{ij}. Each edge could have a weight α_{ij} to measure the importance of the co-occurrence relation between X_i and X_j. Note that G is not necessary to be a complete graph. An edge e_{ij} is absent if the corresponding co-occurrence data is unavailable or not meaningful for an application.

For example, in Figure 1, the corresponding graph G contains 4 types of objects: users, queries, Web pages, and words. We have 5 co-occurrence relations in G and each of them is denoted by an edge in Figure 1, thus we have 5 co-occurrence matrices.

Intuitively, based on graph G, objects of any type (e.g., users) can be represented by objects of the other types to which it is *directly* connected (e.g., Web pages and queries). However, this method is not effective to exploit all the information on a multiple-type graph. A more general method is to represent objects of any type by all the types of objects which have paths to them (e.g., representing users by words). However, due to the complex relations among data objects, there may be many paths between two types of objects (e.g., users and words), thus this method is difficult to be implemented directly.

To effectively utilize the information on a multiple-type graph G, our goal is to find the latent semantic representations for each type of objects. Specifically, based on the co-occurrence data of G, we first identify the most salient *concepts* based on the *mutual reinforcement principle*. These concepts span a semantic space. We then represent each object in this unified low-dimensional space.

4. M-LSA

In this section, we first describe the mutual reinforcement principle based on the analysis of the traditional LSA. We then extend it to multiple types of objects and present the M-LSA algorithm. Finally, we show that two variants of the traditional LSA are special cases of M-LSA. In the following, we denote matrices by upper-case letters (e.g., A, B), scalars by lower-case letters (e.g., a, b), and vectors by bold lower-case letters (e.g., $\mathbf{a}$, $\mathbf{b}$).

4.1 The Mutual Reinforcement Principle of LSA

4.1.1 Brief Review of LSA

LSA is based on a mathematical operation, Singular Value Decomposition (SVD), which is akin to factor analysis. Consider the analysis of document-word co-occurrence data, if there are a total of n documents and m words in a document collection, the process starts with the creation of the co-occurrence matrix between the documents and words $A = [a_{ij}]$, with each entry a_{ij} representing the co-occurrence frequency of the i-th word in the j-th document. For the $m \times n$ matrix A, where without loss of generality $m \leq n$ and $rank(A) = r$, the SVD is defined as [12]:

$$A = U\Sigma V^T$$

where $U = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_r]$ is an $m \times r$ column-orthonormal matrix whose columns are called left singular vectors; $\Sigma = diag[\sigma_1, \sigma_2, ..., \sigma_r]$ is an $r \times r$ diagonal matrix whose diagonal elements are positive singular values sorted in descending order. $V = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_r]$ is an $n \times r$ column-orthonormal matrix whose columns are called right singular vectors.

Given an integer k ($k \ll r$), LSA uses the first k singular vectors to represent the documents and words in a k-dimensional space [10]. Each singular vector is regarded as a latent concept which captures a salient recurrent word combination pattern in the document collection; a document has a large index value for a concept if it contains the corresponding word pattern [6]. More precisely, LSA represents each document by a row of $[\sigma_1\mathbf{v}_1, ..., \sigma_k\mathbf{v}_k]$ and each word by a row of $[\sigma_1\mathbf{u}_1, ..., \sigma_k\mathbf{u}_k]$.

4.1.2 *The Mutual Reinforcement Principle*

In LSA, the first k singular vectors of A represent the most important k concepts in the document collection. Alternatively, we can assume that important concepts are related to both important documents and important words. To identify the most important concept from A, we associate *importance* property for documents and words respectively and utilize the mutual reinforcement principle[1],

> An *important* document co-occurs with *important* words;
> an *important* word co-occurs with *important* documents.

Numerically, if we associate an importance value v_i with the i-th document and an importance value u_j with the j-th word, the mutual reinforcement principle is expressed as:

$$v_i = \textstyle\sum_{j:j\sim i} u_j; u_j = \sum_{i:j\sim i} v_i.$$

where $j \sim i$ means that the j-th word co-occurs with the i-th document. If we denote the importance of all documents by a vector $\mathbf{v}$ and the importance of all words by a vector $\mathbf{u}$, we can express the mutual reinforcement principle as:

$$\left\{ \begin{array}{rcl} \mathbf{v} & = & A^T\mathbf{u} \\ \mathbf{u} & = & A\mathbf{v} \end{array} \right. \qquad (1)$$

It is easy to see that $\mathbf{v} = A^TA\mathbf{v}$ and $\mathbf{u} = AA^T\mathbf{u}$. Therefore, $\mathbf{u}$ and $\mathbf{v}$ are the principal left and right singular vectors of A respectively (please refer to [8] for proof).

Based on the co-occurrence data, the mutual reinforcement principle provides a reasonable solution to factor the most important concept out. However, a collection generally contains multiple topics. The most important concept represents well the most salient topic, but not other topics. Fortunately, the mutual reinforcement principle can also be extended to the non-principal singular vectors of A easily [17]. Specifically we can get the first k singular vectors. Each vector is then considered as a latent concept, and the magnitude of the corresponding singular value represents the importance of the concept. LSA projects the documents and words to a low-dimensional semantic space spanned by these concepts.

[1]Similar principles are used in HITS [17] and [27].

4.1.3 *Unified Importance and Concept Vectors*

In LSA, the latent concepts are represented by $\mathbf{v}$ in document space and $\mathbf{u}$ in word space. Conceptually, however, they are associated with the same concept. Thus, our key idea for generalizing LSA to handle multiple co-occurrence matrices is to represent each concept as a single vector. We now describe the unified importance and concept vectors.

Recall in Equation (1), we have $\mathbf{v}$ and $\mathbf{u}$. If we concatenate them as a *unified* importance vector $[\mathbf{u}, \mathbf{v}]^T$, Equation (1) can be rewritten as:

$$[\mathbf{u}, \mathbf{v}]^T = \left[\begin{array}{cc} 0 & A \\ A^T & 0 \end{array} \right] [\mathbf{u}, \mathbf{v}]^T = B \cdot [\mathbf{u}, \mathbf{v}]^T \qquad (2)$$

where B is defined as:

$$B = \left[\begin{array}{cc} 0 & A \\ A^T & 0 \end{array} \right] \qquad (3)$$

It is easy to see that $[\mathbf{u}, \mathbf{v}]^T$ is the eigenvector of B. Mathematically, [8] shows that the eigenvectors of B are closely related to the singular vectors of A: $\mathbf{u}$ and $\mathbf{v}$ in Equation (2) are the left and right singular vectors of A respectively. Thus, computationally the SVD of A is equal to finding the eigenvectors of B, while B gives us a (more preferred) unified view of importance.

With the notion of a unified concept vector, we can now explain LSA in a unified view. Let $\mathbf{c} = [\mathbf{u}, \mathbf{v}]^T$ be the unified concept vector. We denote the first k eigenvectors of B as $\{\mathbf{c}_1, ..., \mathbf{c}_k\}$. Each of them is a unified concept vector and has the corresponding importance precisely represented by the eigenvalue of B, σ_i, which is the same as the singular value of A [8]. Therefore, with this unified view, LSA represents each object by a row of the matrix:

$$[\sigma_1\mathbf{c}_1, ..., \sigma_k\mathbf{c}_k] = \left[\begin{array}{ccc} \sigma_1\mathbf{u}_1, & ..., & \sigma_k\mathbf{u}_k \\ \sigma_1\mathbf{v}_1, & ..., & \sigma_k\mathbf{v}_k \end{array} \right].$$

The upper part of the matrix is for words and the lower part of the matrix is for documents.

4.2 The M-LSA Algorithm

To factor the latent semantic concepts out from multiple co-occurrence relations represented by a multiple-type graph G, we first extend the mutual reinforcement principle of LSA to multiple-type graph.

> On a multiple-type graph G with N vertices and a number of pairwise co-occurrence relationships, *important* objects of a type co-occur with *important* objects of other types.

The mutual reinforcement principle on a multiple-type graph is a natural generalization of that of two types of objects. This principle also provides a reasonable solution to finding the important latent concepts among the multiple co-occurrence data as we will describe in the following.

Formally, recall that we have N types of objects on graph G: $\{X_1, X_2, ..., X_N\}$. For any two types of objects: X_i and X_j, we have the co-occurrence matrix M_{ij} ($M_{ij} = 0$ if the edge e_{ij} is absent on G). It is easy to see that $M_{ij} = M_{ji}^T$. (For brevity, we only consider the co-occurrence data between different types of objects. The co-occurrence relationship within a single type of objects can be incorporated similarly.) Let us associate an importance value with each object. For the i-th type of objects in X_i, we have one weight vector $\mathbf{w}_i$ to denote their importance. The mutual reinforcement principle can be expressed as:

$$\mathbf{w}_i \doteq \sum_{\forall j:j\neq i} M_{ij}\mathbf{w}_j \qquad (4)$$

Taking unified view of latent concepts, we use $\mathbf{w} = [\mathbf{w}_1, ..., \mathbf{w}_N]^T$ as the concatenated importance vector and define

$$R = \begin{bmatrix} 0 & M_{12} & \cdots & M_{1N} \\ M_{21} & 0 & \cdots & M_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ M_{N1} & M_{N2} & \cdots & 0 \end{bmatrix} \quad (5)$$

as the unified co-occurrence matrix. We can rewrite Equation (4) in a matrix format:

$$\mathbf{w} \doteq R \cdot \mathbf{w} \quad (6)$$

It is easy to show that $\mathbf{w}$ will converge to the eigenvector of the co-occurrence matrix R.

Similar to LSA, since important objects will have high weights in $\mathbf{w}$, we regard $\mathbf{w}$ as the most important latent concept vector across all the co-occurrence relations. Each entry in $\mathbf{w}$ corresponds to an object and its value can be regarded as the association weight between the object and this latent concept. Similarly, the first k eigenvectors of R represent the top k important concepts, which span a k-dimensional semantic space to represent all the objects. Specifically, let

$$\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_k$$

be the top k eigenvalues of R and the corresponding vectors are respectively

$$\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_k.$$

The symmetry of matrix R guarantees that all eigenvalues are real numbers and λ_l gives precisely the importance of the corresponding concept vector $\mathbf{c}_l$ $(1 \leq l \leq k)$. Therefore, the i-th object can be represented by

$$[\lambda_1 c_{1i}, \lambda_2 c_{2i}, ..., \lambda_k c_{ki}]$$

where c_{li} is the i-th entry in $\mathbf{c}_l$, i.e., the association weight between the i-th object and the l-th concept. All the objects are represented in the matrix:

$$[\lambda_1 \cdot \mathbf{c}_1, \lambda_2 \cdot \mathbf{c}_2, ..., \lambda_k \cdot \mathbf{c}_k]$$

with each row representing an object in the k-dimensional space.

The above analysis treats all the co-occurrence relationships equally, i.e., G is unweighted. It is not the best choice in many cases. For example, different co-occurrences might not be equally reliable on the same scale; some may contain more noises than others. On the other hand, the entries in different co-occurrence matrices may have different scales, thus need to be normalized. Therefore, in general, we want to give different weights to different co-occurrence matrices, i.e., G is weighted. To incorporate these weights, we can directly replace M_{ij} by $\alpha_{ij} \cdot M_{ij}$ in matrix R in Equation (5) and then conduct semantic analysis on this new matrix. These weights can be used as normalization factors or to reflect the importance of different matrices for a specific application. Since α_{ij}'s only represent the *relative* importance of the matrices and their scales do not change the eigenvectors of matrix R, without loss of generality, we could add a constraint $\sum_{i<j} \alpha_{ij} = 1$ or set a specific $\alpha_{ij} = 1$ and adjust others.

We use the name **M-LSA** to denote our algorithm since its analysis is based on *multiple-type* graphs.

4.3 Relation with LSA

It is trivial to show that the traditional LSA is a special case in our framework. In particular, Since LSA only deals with two types of objects, say, X_1 and X_2, there is only one co-occurrence matrix $\alpha_{12} \cdot M_{12}$. In the M-LSA framework, we thus have

$$R = \begin{bmatrix} 0 & \alpha_{12} M_{12} \\ \alpha_{12} M_{21} & 0 \end{bmatrix} = \alpha_{12} \cdot \begin{bmatrix} 0 & M_{12} \\ M_{12}^T & 0 \end{bmatrix}$$

It is easy to see that the eigenvectors of R is the same as the eigenvectors of B in Equation (3) when $M_{12} = A$. Furthermore, by setting $\alpha_{12} = 1$, the result of M-LSA for R is the same as the result of LSA for A. Therefore, the traditional LSA is a special case in our M-LSA algorithm.

Previous work has also applied LSA to the objects with two types of features. For example, in [20], LSA was applied to images incorporating both keyword features and low-level image features (e.g., colors of images). By concatenating two kinds of features as longer feature vectors, [20] conducted LSA on a larger matrix. In our M-LSA framework, if we use X_1 as images, X_2 as keywords, and X_3 as low-level features, we have,

$$R = \begin{bmatrix} 0 & \alpha_{12} M_{12} & \alpha_{13} M_{13} \\ \alpha_{12} M_{12}^T & 0 & 0 \\ \alpha_{13} M_{13}^T & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & M \\ M^T & 0 \end{bmatrix}$$

where $M = [\alpha_{12} M_{12}, \alpha_{13} M_{13}]$, which concatenates the keyword and low-level features of images together. Thus, this variant of LSA is also a special case in our M-LSA algorithm.

5. EXPERIMENTS

In this section, we evaluate M-LSA on different data sets and for different tasks, including collaborative filtering, text clustering, and text categorization. In these applications, we design the co-occurrence matrix R from the available data and study the effectiveness of our algorithm. In our experiments, we use two benchmark data sets, MovieLens[2] and 20-Newsgroup[3].

5.1 Application I: Collaborative Filtering

Collaborative Filtering (CF) [14] is to recommend items to an active user based on the historical data of like-minded users. Based on the current ratings of the active user, memory-based algorithms find its nearest neighbors and recommend items based on the ratings of the neighbors. In this paper, we use the Pearson method, a popular memory-based method, as one of our baselines. Pearson method uses Pearson correlation coefficient to find the nearest neighbors for an active user v:

$$w_{v,u} = \frac{\sum_{i=1}^{m} (r_{v,i} - \bar{r}_v)(r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i=1}^{m} (r_{v,i} - \bar{r}_v)^2 \cdot \sum_{i=1}^{m} (r_{u,i} - \bar{r}_u)^2}}$$

where m is the number of items, $r_{v,i}$ $(r_{u,i})$ is the rating of user v (u) for item i, and $\bar{r}_v$ $(\bar{r}_u)$ is the average rating of user v (u). $w_{v,u}$ is the similarity score between the two users. We use N_v as the selected set of the nearest neighbors of v, then the prediction of v's rating for an unseen item i is calculated as

$$\hat{r}_{v,i} = \bar{r}_v + \frac{\sum_{u \in N_v} (r_{u,i} - \bar{r}_u) \cdot w_{v,u}}{\sum_{u \in N_v} w_{v,u}} \quad (7)$$

For the memory-based CF algorithm, the prediction accuracy depends on the accuracy of the nearest neighbors. In our experiments, we show that M-LSA can improve the prediction accuracy by representing the users more meaningfully, thus finding more accurate nearest neighbors.

[2]http://www.cs.umn.edu/research/GroupLens

[3]http://people.csail.mit.edu/jrennie/20Newsgroups

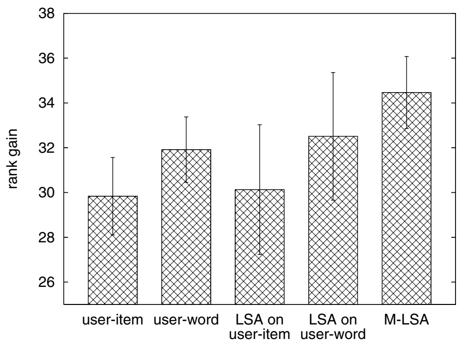

Figure 2: CF result comparison of different methods on MovieLens data

5.1.1 *Experiment Design*

We use the benchmark data set MovieLens, which includes 100,000 ratings (1–5) from 943 users on 1682 movies. Each movie has the title keyword information. Therefore, we have three types of objects: users (X_1), items (X_2) and keywords (X_3). The co-occurrence matrices include user-item (M_{12}), user-word (M_{13}) and item-word (M_{23}). M_{12} is constructed with the ratings of users on movies; M_{23} is constructed with the movie titles. We construct M_{13} as follows: if a keyword appears in a title of any movie rated by a user, this keyword and this user has a co-occurrence. Their co-occurrence frequency is calculated by the user's rating on movies and the term-frequency of this keyword in the corresponding movie titles. We weigh M_{13} and M_{23} by the standard TF-IDF scheme which is commonly used in retrieval retrieval [2]. Finally, we can construct the relation matrix R based on the three matrices as:

$$R = \begin{bmatrix} 0 & \alpha M_{12} & \beta M_{13} \\ \alpha M_{12}^T & 0 & \gamma M_{23} \\ \beta M_{13}^T & \gamma M_{23}^T & 0 \end{bmatrix} \quad (8)$$

where the weight parameters $\alpha, \beta, \gamma \geq 0$, and $\alpha + \beta + \gamma = 1$.

We use two baseline methods. One is the Pearson method which is based on user-item matrix M_{12}. The other is based on user-word matrix M_{13} to calculate the nearest neighbors. We also apply the traditional LSA on these two matrices and then use the low-dimensional representation to calculate the nearest neighbors. Our M-LSA will be compared with all the four methods. For all the five methods, after calculating the nearest neighbors, the predictions are based on Equation (7) and the comparison is based on the half-time utility metric defined in [7]. For a user v, the expected utility of a ranked list of items is:

$$\mathcal{R}_v = \sum_j \frac{\max(r_{v,j} - \bar{r}_v, 0)}{2^{(j-1)/(\tau-1)}}$$

where τ is the half-time parameter ($\tau = 5$ in our experiments) and $r_{v,j}$ is v's rating for the item which is at the j-th position in the current rank list. The final score over all users in the test set is:

$$\mathcal{R} = 100 \frac{\sum_v \mathcal{R}_v}{\sum_v \mathcal{R}_v^{\max}}$$

where $\mathcal{R}_v^{\max}$ is the maximum possible utility obtained where all the test items are ranked at top according to user v's rating. In the following figures, we use "rank gain" to denote $\mathcal{R}$ score.

5.1.2 *Results*

All the results are averaged over a randomly split of 5 folds on the MovieLens data. We fix the number of neighbors to 50 and the number of dimensions for LSA and M-LSA to 20. We set $\alpha = 0.3$, $\beta = 0.5$, and $\gamma = 0.2$ in matrix R. Figure 2 gives the comparison results of different methods. The figure shows that M-LSA achieves the best result. Compared with the other methods, M-LSA achieves a relative improvement of 15.5% over user-item, 8.0% over user-word, 14.4% over LSA(user-item), and 6.0% over LSA(user-word) with respect to the $\mathcal{R}$ score measure. Our results are consistent with that of [22], where the authors found that user-word matrix could get better result because it is less sparse than user-item matrix. LSA based methods could only achieve marginal improvement compared with baselines. Our method can improve the utility over LSA significantly. We also calculate the standard deviation of the five methods over the 5 folds. The results in Figure 2 show that M-LSA is more stable. This confirms that M-LSA can effectively explore all the co-occurrence relations to represent objects more meaningfully.

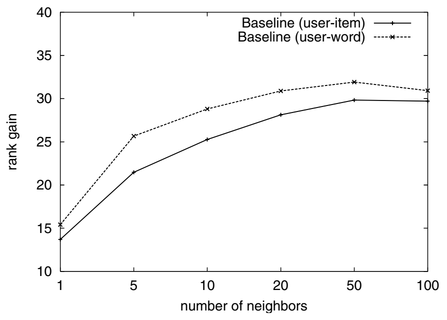

Figure 3: Impact of number of neighbors on baseline methods

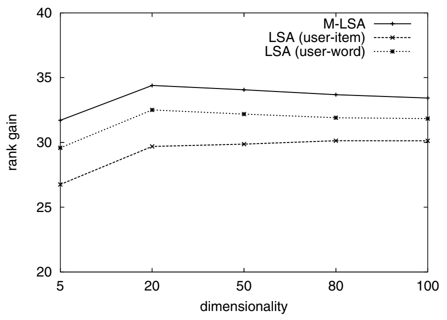

Figure 4: Impact of the dimensionality on LSA and M-LSA

We also study the parameters for different methods. In Figure 3, we plot the results along with the number of neighbors for the two baselines. We can see that the best results are obtained when the number is 50. Thus we fix the number of neighbors as 50 in all the experiments.

In Figure 4, we study the impact of number of dimensions for LSA and M-LSA. For M-LSA, we set $\alpha = 0.3$, $\beta = 0.5$, and $\gamma = 0.2$ in matrix R. It can be seen that M-LSA consistently outperforms LSA based methods and all attain their best results when the dimensionality is 20.

Finally, we study the influence of α, β, and γ in matrix R for M-LSA. Since $\alpha + \beta + \gamma = 1$, we only vary α and β and report our results in Table 1. We vary α (β respectively) from 0.1 to 0.9 by step 0.2 and we retain the top 20 eigenvectors for M-LSA. In this table, we get the best result when $\alpha = 0.3$, $\beta = 0.5$, thus $\gamma = 0.2$. Furthermore, when $\alpha \geq 0.3$, most of the results (bold font) are better than the baselines and LSA based methods, thus M-LSA is effective for a wide range of parameters.

Table 1: Impact of weight parameters of M-LSA on CF results

		β				
		0.1	0.3	0.5	0.7	0.9
α	0.1	24.47	32.2	**33.78**	**33.57**	**33.45**
	0.3	29.32	**33.92**	**34.46***	**34.02**	-
	0.5	29.81	**32.53**	**34.03**	-	-
	0.7	29.08	31.04	-	-	-
	0.9	28.92	-	-	-	-

It is worth noting that when $\gamma = 0$ (i.e., $\alpha + \beta = 1.0$ in Table 1), we only consider the user-item matrix M_{12} and user-word matrix M_{13} but not M_{23}, which means we concatenate both features with appropriate weights to represent the users. However, we obtained the best result when $\alpha = 0.3$, $\beta = 0.5$, and $\gamma = 0.2$. This means all the co-occurrence information is useful and can be incorporated by M-LSA effectively.

5.2 Application II: Text Clustering

Text clustering is one of the fundamental problems and has received much attention recently. When the vector space model (VSM) is used, each document is represented as a term vector and the similarity score between two documents is calculated as the cosine value (or dot product) of corresponding term vectors. In this section, we show experimentally that our M-LSA method can improve the document representation and boost text clustering result significantly. We use k-means, one of the most popular clustering algorithms, to compare different document representation methods.

5.2.1 Experiment Design

To obtain multiple co-occurrence data, we use the 20-Newsgroup data set. In this newsgroup data, different emails or posts may have the same subject. Thus, besides the email-word relation, we have email-subject and subject-word relations. Therefore, the objects we have are: emails (X_1), subjects (X_2), and words (X_3). Each pair of them has a co-occurrence matrix and the obtained matrix R is similar as in Equation (8). We select the five "comp.*" out of the 20 categories as our data set. Each of the five categories has 1000 emails, thus, we have 5000 in total. After preprocessing the subjects by removing "Re:" and "Fwd:", we obtain 2933 subjects in total. We use the F measure defined in [23] as our evaluation metric. For each cluster, we calculate its *Precision* and *Recall* with respect to each given category. The *F measure* is defined by combining the *Precision* and *Recall* together. Specifically, for cluster j and category i:

$$Recall(i,j) = \frac{n_{ij}}{n_i}$$

$$Precision(i,j) = \frac{n_{ij}}{n_j}$$

$$F(i,j) = \frac{2 \times Precision(i,j) \times Recall(i,j)}{Precision(i,j) + Recall(i,j)}$$

where n_{ij} is the number of members of category i in cluster j, n_i is the number of members in category i, n_j is the number of members in cluster j and $F(i,j)$ is the *F measure* of cluster j and category i. The F measure of the whole clustering result is defined as a weighted sum over all categories as follows:

$$F = \sum_i \frac{n_i}{n} \max_j \{F(i,j)\}$$

where the $\max$ is taken over all clusters.

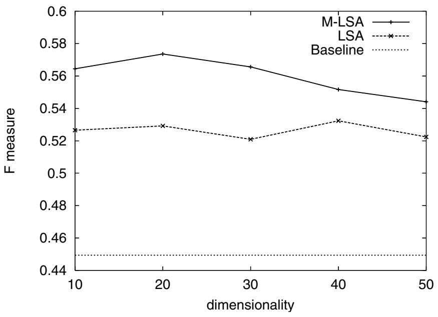

Figure 5: Results on text clustering. We compare the result on different dimensions

In our experiments, we first represent each email (body + subject) using the TF-IDF method. This gives us the baseline result. LSA is applied to this TF-IDF representation and our M-LSA method is based on the matrix R defined above. K-means is then run on the low-dimensional representations for LSA and M-LSA. We randomly select the initial centroids 20 times and have 20 runs. The results for each method are averaged over these 20 runs.

5.2.2 Results

We set $\alpha = 0.3$, $\beta = 0.5$, and $\gamma = 0.2$ for M-LSA. Figure 5 is the primary results for clustering. In this figure, we vary the number of dimensions for both LSA and M-LSA. It is clear that M-LSA and LSA outperform the baseline method (0.449) substantially. By comparing M-LSA with LSA, we can see that M-LSA always achieves better result. For example, when we set the number of dimensions to 20, the F measures of M-LSA and LSA are 0.574 and 0.532 respectively. Thus, M-LSA achieves 8.0% relative improvement over LSA. The t-test over the 20 runs indicates the improvement is statistically significant (p-value=0.0001).

For M-LSA, we also study the parameters α, β, and γ in a similar way as in collaborative filtering. We set the number of dimensions to 20 and the results are reported in Table 3. The best result (0.5736) is obtained when $\alpha = 0.3$, $\beta = 0.5$, and $\gamma = 0.2$. A difference here is that when α is set to a large value, the F measure is lowered a lot. This is because α is the weight for the (email, subject) co-occurrence matrix M_{12} and each email has only one subject. On average, 1.70 emails share a subject. Thus, the co-occurrence between email and subject is extremely sparse and only two emails that have the same subject will have a nonzero similarity score if we set $\alpha = 1$. Thus, the clustering result is biased when α is set too large. However, given an appropriate weight to this co-occurrence matrix, M-LSA can improve the clustering performance substantially. This again confirms the effectiveness of M-LSA to incorporate the meaningful co-occurrence information.

In Table 2, we show the first five eigenvectors of R when we set $\alpha = 0.3$, $\beta = 0.5$, and $\gamma = 0.2$. The most important words and their weights associated with each eigenvector are given in this table. It is clear that each eigenvector is a focused concept. For example, eigenvector 2 is about "operating system", eigenvector 3 is about "hardware", and eigenvector 4 is about "monitor". This shows the effectiveness of M-LSA to identify latent semantic concepts by the mutual reinforcement principle.

5.3 Application III: Text Categorization

Finally, we test our algorithm on text categorization problem. This experiment is to show that M-LSA is much flexible to model

Table 2: The first five concepts of on comp.* newsgroup data

Eigenvector 1	Eigenvector 2	Eigenvector 3	Eigenvector 4	Eigenvector 5
window 0.099589	microsoft 0.101049	scsi 0.191321	card 0.191498	drive 0.117997
card 0.078050	os 0.065222	drive 0.155479	monitor 0.145576	appl 0.109939
drive 0.077698	ms 0.049975	id 0.138539	video 0.142046	mac 0.108689
file 0.073806	challeng 0.048302	controll 0.101291	driver 0.103758	power 0.078488
system 0.068470	window 0.043423	mb 0.098239	ati 0.088274	disk 0.068229
do 0.067048	product 0.041857	bu 0.091817	dx 0.076899	price 0.063807
run 0.066707	duke 0.040493	hard 0.073117	diamond 0.069758	monitor 0.062536
edu 0.066448	innov 0.034644	isa 0.072097	simm 0.067689	simm 0.057811
com 0.065669	market 0.033406	card 0.068160	ultra 0.064259	hard 0.054318
help 0.063963	do 0.033213	disk 0.055418	mhz 0.061410	centri 0.050969

Table 4: The representative words for comp.* categories

os.ms-windows.misc		graphics		sys.mac.hardware		sys.ibm.pc.hardware		windows.x	
Centroid	M-LSA	Centroid	M-LSA	Centroid	M-LSA	Centroid	M-LSA	Centroid	M-LSA
window	window	graphic	imag	mac	mac	drive	drive	window	window
file	file	imag	graphic	appl	appl	card	card	motif	server
driver	do	file	file	drive	monitor	controll	scsi	server	applic
do	driver	program	format	monitor	drive	scsi	controll	widget	displai
win	run	look	gif	simm	simm	bu	bu	displai	run
edu	ms	format	convert	edu	price	id	id	applic	motif
program	win	gif	program	price	edu	pc	mb	sun	widget
run	program	convert	look	quadra	comput	dx	system	run	sun
ms	edu	thank	ftp	centri	power	com	disk	com	program
card	card	bit	color	lc	centri	mb	hard	program	set

Table 3: Impact of weight parameters for M-LSA on clustering results

		β				
		0.1	0.3	0.5	0.7	0.9
α	0.1	0.4617	0.4612	**0.5458**	**0.5371**	**0.5332**
	0.3	0.4653	0.5217	**0.5736***	**0.5420**	-
	0.5	0.4199	0.4477	0.5312	-	-
	0.7	0.3703	0.3927	-	-	-
	0.9	0.3818	-	-	-	-

a variety of co-occurrence relations. In particular, we show that M-LSA can incorporate category information.

5.3.1 Experiment Design

The data set we used in this experiment is the same as above: 5 categories of "comp.*" newsgroup data. For each category, we randomly select 800 documents as training and use the remaining 200 as test data.

To incorporate the category information, in this experiment, we use a different co-occurrence relation matrix. The objects are: emails (X_1), categories (X_2), and words (X_3). M_{13} is based on the TF-IDF weighting of email-word matrix. Since each training example has a word vector in M_{13}, for each category, we calculate the centroid vector of the corresponding training examples. The category-word matrix (M_{23}) is composed of all these centroid vectors. We construct the binary email-category matrix M_{12} as follows: for each training example, it has a co-occurrence with its labeled category. The test data is not used in the construction of the co-occurrence matrix.

We apply the M-LSA algorithm to the above co-occurrence matrix and obtain the first k eigenvectors. We then project both the training and test examples along these k eigenvectors, thus represent all the examples in a k-dimensional space. Classification experiments are conducted in this space.

Since M_{13} and M_{23} are standard TF-IDF matrices, we set their weights to 1.0. We thus only vary the weight α for matrix M_{12} and study its impact. The co-occurrence matrix is:

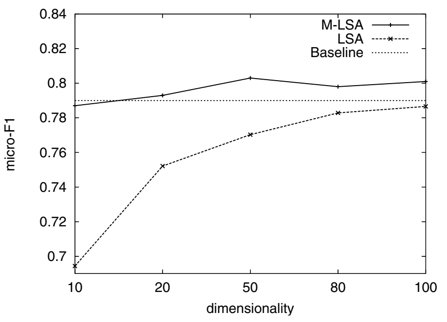

Figure 6: Results on text categorization. We compare the results on different dimensions

$$R = \begin{bmatrix} 0 & \alpha M_{12} & M_{13} \\ \alpha M_{12}^T & 0 & M_{23} \\ M_{13}^T & M_{23}^T & 0 \end{bmatrix} \tag{9}$$

For the classifier, we use the SVM^{light} software[4] and all the results reported below are based on the micro-averaging F1 (micro-F1) measure defined in [26]. F1 measure is a tradeoff between precision and recall. Another commonly-used F1 measure is macro-averaging F1 (macro-F1). Macro-F1 is the arithmetic average of F1 measure over all the categories and micro-F1 is the weighted average that emphasizes on categories with more examples. Since all the categories in our data set have the same size, macro-F1 is similar to micro-F1 measure. Thus we only report micro-F1.

5.3.2 Results

We compare our result with the standard SVM on the email-word matrix and LSA based method. We set $\alpha = 0.3$ and Figure 6 shows the results along with different dimensions. It is clear that M-LSA can outperform the baseline, while LSA can not. When we set the number of dimensions to 50, M-LSA achieves 0.803 on micro-F1,

[4]http://svmlight.joachims.org

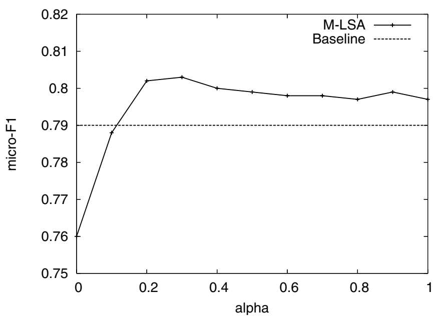

Figure 7: Impact of weight for text categorization

which is higher than the baseline 0.790 and the best result of LSA 0.789. Since M-LSA incorporates the category supervised information into the co-occurrence matrix, it can achieve better result even when the dimension is set to a smaller value (e.g., 20), as shown in Figure 6.

Figure 7 shows the impact of the weight α for the matrix M_{12}. We set the number of dimensions to 50 and vary α from 0.0 to 1.0 with step 0.1. When $\alpha = 0.0$, M-LSA does not consider the category information thus the result is the same as LSA. From this figure, we can see that M-LSA is insensitive to the change of α when $\alpha \geq 0.1$ and its micro-F1 is always higher than the baseline. We obtain the best result when the weight is 0.3. The result is really encouraging since it indicates: although M-LSA is unsupervised in spirit, it can also incorporate the supervised information by appropriately introducing the co-occurrence relationship.

Finally, we show that M-LSA can associate meaningful words to categories. In Table 4, we show the most salient words in the centroid vector of each category (denoted by Centroid). For M-LSA, we do not use category-word matrix M_{13} in Equation (9) and set the weight of email-category M_{12} to $\alpha = 0.3$. We use the first 50 eigenvectors for M-LSA. In M-LSA, all the categories and words are projected into a unified semantic space, we thus calculate the similarities between words and categories by their dot products in this space. For each category, the most similar words are reported in Table 4 (denoted by M-LSA). It is clear that there is a big overlap between Centroid and M-LSA based methods. This again confirms the effectiveness of M-LSA in identifying the latent concepts by utilizing all the co-occurrence information and representing each type of objects meaningfully in a unified semantic space.

6. CONCLUSIONS

It is important to exploit the co-occurrence relations among different types of objects in many applications. In this paper, we modelled all the pairwise co-occurrence relations with a multiple-type graph and proposed a general algorithm, M-LSA, which conducts latent semantic analysis by incorporating all pairwise co-occurrences among multiple types of objects. Based on the mutual reinforcement principle as used in the traditional LSA, M-LSA identifies the most salient concepts among all the co-occurrence data and represents each object in a unified semantic space. M-LSA is general and covers several variants of LSA as special cases. We evaluated M-LSA on three applications and obtained very encouraging results. All the experiments showed that M-LSA is effective in utilizing all the information on a multiple-type graph. M-LSA can be applied to any co-occurrence data involving multiple types of objects, thus has potentially many applications in multiple domains.

7. REFERENCES

[1] R. K. Ando. Latent semantic-space: iterative scaling improves precision of inter-document similarity measurement. In *Proceedings of the 23th SIGIR*, pages 216–223, 2000.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.

[3] B. T. Bartell, G. W. Cottrell, and R. K. Belew. Latent semantic indexing is an optimal special case of multidimensional scaling. In *SIGIR*, pages 161–167, 1992.

[4] H. Bast and D. Majumdar. Why spectral retrieval works. In *SIGIR*, pages 11–18, 2005.

[5] R. Bekkerman, R. El-Yaniv, and A. McCallum. Multi-way distributional clustering via pairwise interactions. In *ICML*, 2005.

[6] M. Berry, S. Dumais, and G. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595, 1995.

[7] J. S. Breese, D. Heckerman, and C. M. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, pages 43–52, 1998.

[8] J. K. Cullum and R. A. Willoughby. *Lanczos Algorithms for Large Symmetric Eigenvalue Computation, Volumn 1 Theory.* Birkhäuser, Boston, 1985.

[9] B. D. Davison. Toward a unification of text and link analysis. In *SIGIR*, pages 367–368, 2003.

[10] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science and Technology (JASIS)*, 41(6):391–407, 1990.

[11] C. H. Q. Ding. A probabilistic model for latent semantic indexing. *JASIST*, 56(6):597–608, 2005.

[12] G. H. Golub and C. F. V. Loan. *Matrix Computations, third edition*. The Johns Hopkins University Press, 1996.

[13] Y. Gong and X. Liu. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th SIGIR*, pages 19–25, 2001.

[14] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR*, pages 230–237, 1999.

[15] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, pages 50–57, 1999.

[16] G. Jeh and J. Widom. Simrank: A measure of structural-context similarity. In *KDD*, 2002.

[17] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[18] L. D. Lathauwer, B. D. Moor, and J. Wandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[19] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[20] F. Monay and D. Gatica-Perez. On image auto-annotation with latent space models. In *ACM Multimedia*, pages 275–278, 2003.

[21] C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. Latent semantic indexing: A probabilistic analysis. *J. Comput. Syst. Sci.*, 61(2):217–235, 2000.

[22] A. Popescul, L. H. Ungar, D. M. Pennock, and S. Lawrence. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *UAI*, pages 437–444, 2001.

[23] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *Technical Report 00-034. Department of Computer Science and Engineering, University of Minnesota*, 2000.

[24] W. Xi, E. A. Fox, W. Fan, B. Zhang, Z. Chen, J. Yan, and D. Zhuang. Simfusion: measuring similarity using unified relationship matrix. In *SIGIR*, pages 130–137, 2005.

[25] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *SIGIR*, pages 267–273, 2003.

[26] Y. Yang and X. Liu. A re-examination of text categorization methods. In *SIGIR*, pages 42–49, 1999.

[27] H. Zha. Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. In *SIGIR*, pages 113–120, 2002.