A Unified Framework for Providing Recommendations in Social Tagging Systems Based on Ternary Semantic Analysis

Panagiotis Symeonidis, Alexandros Nanopoulos, and Yannis Manolopoulos

Abstract—Social Tagging is the process by which many users add metadata in the form of keywords, to annotate and categorize items (songs, pictures, web links, products, etc.). Social tagging systems (STSs) can provide three different types of recommendations: They can recommend 1) tags to users, based on what tags other users have used for the same items, 2) items to users, based on tags they have in common with other similar users, and 3) users with common social interest, based on common tags on similar items. However, users may have different interests for an item, and items may have multiple facets. In contrast to the current recommendation algorithms, our approach develops a unified framework to model the three types of entities that exist in a social tagging system: users, items, and tags. These data are modeled by a 3-order tensor, on which multiway latent semantic analysis and dimensionality reduction is performed using both the Higher Order Singular Value Decomposition (HOSVD) method and the Kernel-SVD smoothing technique. We perform experimental comparison of the proposed method against state-of-the-art recommendation algorithms with two real data sets (Last.fm and BibSonomy). Our results show significant improvements in terms of effectiveness measured through recall/precision.

Index Terms—Social tags, recommender systems, tensors, HOSVD.

1 Introduction

Social tagging is the process by which many users add metadata in the form of keywords, to annotate and categorize songs, pictures, products, etc. Social tagging is associated to the "Web 2.0" technologies and has already become an important source of information for recommender systems. For example, music recommender systems such as Last.fm and MyStrands allow users to tag artist, songs, or albums. In e-commerce sites such as Amazon, users tag products to easily discover common interests with other users. Moreover, social media sites, such as Flickr and YouTube use tags for annotating their content. All these systems can further exploit these social tags to improve the search mechanisms and personalized recommendations. Social tags carry useful information not only about the items they label, but also about the users who tagged. Thus, social tags are a powerful mechanism that reveal three-dimensional correlations between users, tags, and items.

Several social tagging systems (STSs), e.g., Last.fm, Amazon, YouTube, etc., recommend items to users, based on tags they have in common with other similar users. Traditional recommender systems use techniques such as Collaborative Filtering (CF) [5], [15], [16], [20], which apply to two-dimensional data, i.e., users and items. Thus, such systems do not capture the multimodal use of tags. To alleviate this problem, Tso-Sutter et al. [36] propose a generic method that allows tags to be incorporated to standard CF algorithms, by reducing the three-dimensional correlations to three 2D correlations and then applying a fusion method to reassociate these correlations.

Another type of recommendation in STSs, e.g., Facebook, Amazon, etc., is to recommend tags to users, based on what tags other users have provided for the same items. Tag recommendations can expose different facets of an information item and relieve users from the obnoxious task to come up with a good set of tags. Thus, tag recommendation can reduce the problem of data sparsity in STSs, which results by the unwillingness of users to provide an adequate number of tags. Recently, several algorithms have been proposed for tag recommendation [17], [39], which project the three-dimensional correlations to three 2D correlations. Then, the two-dimensional correlations are used to build conceptual structures similar to hyperlink structures that are used by Web search engines.

A third type of recommendation that can be provided by STSs is to recommend *interesting* users to a target user, opting in connecting people with common interests and encouraging people to contribute and share more content. With the term *interesting* users, we mean those users who have similar profile with the target user. If a set of tags is frequently used by many users, then these users spontaneously form a group of users with common interests, even though they may not have any physical or online connections. The tags represent the commonly interested web contents to this user group. For example, Amazon recommends to a user who used a specific tag, other new users

Q1
- *P. Symeonidis and Y. Manolopoulos are with the Department of Informatics, Aristotle University, Thessaloniki 54124, Greece. E-mail: {symeon, manolopo}@csd.auth.gr.*
- *A. Nanopoulos is with the Information Systems and Machine Learning Lab, Marienburger Platz 22, University of Hildesheim, 31141 Hildesheim, Germany. E-mail: nanopoulos@ismll.de.*

Manuscript received 29 Apr. 2008; revised 24 Nov. 2008; accepted 25 Mar. 2009; published online 31 Mar. 2009.
Recommended for acceptance by Q. Yang.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2008-04-0228.
Digital Object Identifier no. 10.1109/TKDE.2009.85.

considering them as *interesting* ones. Amazon ranks them based on how frequent they used the specific tag.

1.1 Motivation

The three types of recommendations in STSs (i.e., item, tag, and user recommendations) have been so far addressed separately by various approaches, which differ significantly to each other and have, in general, an ad hoc nature. Since in STSs all three types of recommendations are important, what is missing is a unified framework that can provide all recommendation types with a single method.

Moreover, existing algorithms do not consider the three dimensions of the problem. In contrast, they split the three-dimensional space into pair relations {user, item}, {user, tag}, and {tag, item}, that are two-dimensional, in order to apply already existing techniques like CF, link mining, etc. Therefore, they miss a part of the total interaction between the three dimensions. What is required is a method that is able to capture the three dimensions all together without reducing them into lower dimensions.

Finally, the existing approaches fail to reveal the latent associations between tags, users, and items. Latent associations exist due to three reasons: 1) users have different interests for an item, 2) items have multiple facets, and 3) tags have different meanings for different users. As an example, assume two users in an STSs for web bookmarks (e.g., Del.icio.us, Bibsonomy). The first user is a car fan and tags a site about cars, whereas the other tags a site about wild cats. Both use the tag "jaguar." When they provide the tag "jaguar" to retrieve relevant sites, they will receive both sites (cars and wild cats). Therefore, what is required is a method that can discover the semantics that are carried by such latent associations, which in the previous example can help to understand the different meanings of the tag "jaguar."

1.2 Contribution

In this paper, we develop a unified framework that models the three dimensions, i.e., items, tags, and users. The three-dimensional data are represented by three-dimensional matricies, which are called *3-order tensors*. We avoid splitting the three-dimensional correlations and we handle all dimensions equally. To reveal latent semantics, we perform 3-mode analysis, using the Higher Order Singular Value Decomposition (HOSVD) [24]. Our method reveals latent relations among objects of the same type, as well among objects of different types.

Moreover, the proposed method addresses the problem that three-dimensional data are highly sparse. Sparsity stems from the fact that users tend to tag only a very small portion of items. Recommender systems are susceptible to data sparsity, which affects their performance. SVD has been proved useful to address the data sparseness problem for traditional CF algorithms (i.e., for two-dimensional rating data) [34]. However, sparsity is more severe in three-dimensional data, and handling sparsity in this case is still an open problem. In our approach, to address the sparseness problem, we combine kernel-SVD [8], [10] with HOSVD. This Kernel-SVD smoothing substantially improves the accuracy of item and tag recommendations.

The contributions of our approach are summarized as follows:

- For the first time to our knowledge, we provide a unified framework for providing all three types of recommendations in STSs: item, tag, and user recommendations.
- We use a 3-order tensor to model the three types of entities (user, item, and tag) that exist in social sites.
- We apply dimensionality reduction (HOSVD) in 3-order tensors, to reveal the latent semantic associations between users, items, and tags. We also apply a smoothing technique based on Kernel-SVD to address the sparseness of data.
- We perform extensive experimental comparison of the proposed method against state-of-the-art recommendation algorithms, using Last.fm and BibSonomy data sets.
- Our method substantially improves accuracy of item and tag recommendations. Moreover, we study a problem of how to provide user recommendations, which can have significant applications in real systems but which have not been studied in depth so far in related research.

The rest of this paper is organized as follows: Section 2 summarizes the related work, whereas Section 3 briefly reviews background techniques employed in our approach. A motivating example and the proposed approach are described in Section 4. Experimental results are given in Section 5. Finally, Section 6 concludes this paper.

2 Related Work

In this section, we briefly present some of the research literature related to Social Tagging. We also present related work in tag, item, and users recommendation algorithms. Finally, we present works that applied HOSVD in various research domains.

Social Tagging is the process by which many users add metadata in the form of keywords to share content. So far, the literature has studied the strengths and the weaknesses of STSs. In particular, Golder and Huberman [13] analyzed the structure of collaborative tagging systems as well as their dynamical aspects. Moreover, Halpin et al. [14] produced a generative model of collaborative tagging in order to understand the dynamics behind it. They claimed that there are three main entities in any tagging system: users, items, and tags.

In the area of item recommendations, many recommender systems already use CF to recommend items based on preferences of similar users, by exploiting a two-way relation of users and items [5]. In 2001, Item-based algorithm was proposed, which is based on the items' similarities for a neighborhood generation [29]. However, because of the ternary relational nature of Social Tagging, two-way CF cannot be applied directly, unless the ternary relation is reduced to a lower dimensional space. Jaschke et al. [19], in order to apply CF in Social Tagging, considered for the ternary relation of users, items, and tags two alternative two-dimensional projections. These projections preserve the user information, and lead to log-based like recommender systems based on occurrence or nonoccurrence of items, or tags, respectively, with the users. Another recently proposed state-of-the-art item recommendation algorithm is tag-aware Fusion [36]. They propose a generic method that allows tags

to be incorporated to standard CF algorithms, by reducing the three-dimensional correlations to three 2D correlations and then applying a fusion method to reassociate these correlations.

In the area of tag recommendation, there are algorithms which are based on conceptual structures similar to the hyperlink structures used in Search Engines. For example, Collaborative Tag Suggestions algorithm [39], also known as Penalty-Reward algorithm (PR), uses an authority score for each user. The authority score measures how well each user has tagged in the past. This authority score can be computed via an iterative algorithm similar to HITS [22]. Moreover, the PR algorithm "rewards" the high correlation among tags, whereas it "penalizes" the overlap of concepts among the recommended tags to allow high coverage of multiple facets for an item. Another state-of-the-art tag recommendation algorithm is FolkRank [17]. FolkRank exploits the conceptual structures created by people inside the STSs. Their method is inspired by the seminal PageRank [28] algorithm, which reflects the idea that a web page is important, if there are many pages linking to it, and if those pages are important themselves. FolkRank employs the same underlying principle for Web Search and Ranking in Social Tagging. The key idea of FolkRank algorithm is that an item which is tagged with important tags by important users becomes important itself. The same holds for tags and users; thus, we have a tripartite graph of vertices which mutually reinforcing each other by spreading their weights. FolkRank is like the Personalized PageRank, which is a modification of global PageRank, and was first proposed for personalized Web search in [28]. Finally, Xu et al. [38] proposed a method that recommends tags by using HOSVD. However, their method does not cover all three types of recommendations in STSs and misses the comparison with state-of-the-art algorithms. In contrast, our approach proposes a unified framework for all recommendation types in STSs. We also combine HOSVD with Kernel-SVD to handle data sparsity, attaining significant improvements in the accuracy of recommendations in comparison with simple HOSVD, as will be shown experimentally.

In the area of discovering shared interests in social networks there are two kinds of existing approaches [25]. One is user-centric, which focuses on detecting social interests based on the online connections among users; the other is object-centric, which detects common interests based on the common objects fetched by users in a social community. In the user-centric approach, recently Ali-Hasan and Adamic [2] analyzed user's online connections to discover users with particular interests for a given user. Different from this kind of approach, we aim to find the people who share the same interest no matter whether they are connected by a social graph or not. In the object-centric approach, Sripanidkulchai et al. [31] explored the common interests among users based on the common items they fetched in peer-to-peer networks. However, they cannot differentiate the various social interests on the same items, due to the fact that users may have different interests for an information item and an item may have multiple facets. In contrast, our approach focuses on directly detecting social interests and recommending users by taking advantage of social tagging, by utilizing users' tags.

Differently from existing approaches, our method develops a unified framework to concurrently model all three dimensions. Usage data are modeled by a 3-order tensor, on which latent semantic analysis is performed using the HOSVD [24]. Moreover, to address the sparseness problem, we propose the combination of Kernel-SVD [8], [10] with HOSVD, which substantially improves the accuracy of item and tag recommendations.

HOSVD is a generalization of singular value decomposition (SVD) and has been successfully applied in several areas. In particular, Wang and Ahuja [37] present a novel multilinear algebra-based approach to reduced dimensionality representation of multidimensional data, such as image ensembles, video sequences, and volume data. In the area of Data Clustering, Chen et al. [7] used also a high-order tensor. However, they transform the initial tensor (through Clique Expansion algorithm) into lower dimensional spaces, so that clustering algorithms (such as k-means) can be applied. Finally, in the area of Personalized Web Search, Sun et al. proposed CubeSVD [32] to improve Web Search. They claimed that as the competition of Web Search increases, there is a high demand for personalized Web search. Therefore based on their CubeSVD analysis, Web Search activities can be carried out more efficiently.

3 Preliminaries—Tensors and HOSVD

In this section, we summarize the HOSVD procedure. In the following, we denote tensors by calligraphic uppercase letters (e.g., $\mathcal{A}, \mathcal{B}$), matrices by uppercase letters (e.g., A, B), scalars by lowercase letters (e.g., a, b), and vectors by bold lowercase letters (e.g., $\mathbf{a}, \mathbf{b}$).

SVD and Latent Semantic Indexing. The SVD [3] of a matrix $F_{I_1 \times I_2}$ can be written as a product of three matrices, as shown in (1):

$$F_{I_1 \times I_2} = U_{I_1 \times I_1} \cdot S_{I_1 \times I_2} \cdot V^T_{I_2 \times I_2}, \tag{1}$$

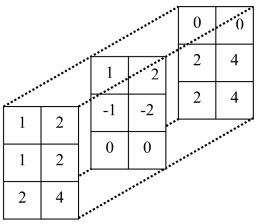
where U is the matrix with the left singular vectors of F, V^T is the transpose of the matrix V with the right singular vectors of F, and S is the diagonal matrix of (ordered) singular values of F.

By preserving only the largest $c < \min\{I_1, I_2\}$ singular values of S, SVD results to matrix $\hat{F}$, which is an approximation of F. In Information Retrieval, this technique is used by Latent Semantic Indexing (LSI) [12], to deal with the latent semantic associations of terms in texts and to reveal the major trends in F.

Tensors. A *tensor* is a multidimensional matrix. An N-order tensor $\mathcal{A}$ is denoted as $\mathcal{A} \in R^{I_1 \ldots I_N}$, with elements $a_{i_1,\ldots,i_N}$. In this paper, for the purposes of our approach, we only use 3-order tensors.

HOSVD. The high-order singular value decomposition [24] generalizes the SVD computation to multidimensional matrices. To apply HOSVD on a 3-order tensor $\mathcal{A}$, three *matrix unfolding* operations are defined as follows [24]:

$$A_1 \in R^{I_1 \times I_2 I_3}, \quad A_2 \in R^{I_2 \times I_1 I_3}, \quad A_3 \in R^{I_1 I_2 \times I_3},$$



$$\begin{pmatrix} 1 & 2 & 1 & 2 & 0 & 0 \\ 1 & 2 & -1 & -2 & 2 & 4 \\ 2 & 4 & 0 & 0 & 2 & 4 \end{pmatrix}$$

Fig. 1. An example tensor $\mathcal{A}$ and its 1-mode matrix unfolding A_1.

where A_1, A_2, and A_3 are called the 1-mode, 2-mode, and 3-mode matrix unfoldings of $\mathcal{A}$, respectively. Each A_n, $1 \leq n \leq 3$, is called the n-mode matrix unfolding of $\mathcal{A}$ and is computed by arranging the corresponding fibers of $\mathcal{A}$ as columns of A_n. The left part of Fig. 1 depicts an example tensor, whereas the right part its 1-mode matrix unfolding $A_1 \in R^{I_1 \times I_2 I_3}$, where the columns (1-mode fibers) of $\mathcal{A}$ are being arranged as columns of A_1.

Next, we define the n-mode product of an N-order tensor $\mathcal{A} \in R^{I_1 \times \cdots \times I_N}$ by a matrix $U \in R^{J_n \times I_n}$, which is denoted as $\mathcal{A} \times_n U$. The result of the n-mode product is an $(I_1 \times I_2 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N)$-tensor, the entries of which are defined as follows:

$$(\mathcal{A} \times_n U)_{i_1 i_2 \ldots i_{n-1} j_n i_{n+1} \ldots i_N} = \sum_{i_n} a_{i_1 i_2 \ldots i_{n-1} i_n i_{n+1} \ldots i_N} u_{j_n i_n}. \quad (2)$$

Since we focus on 3-order tensors, $n \in \{1, 2, 3\}$, we use 1-mode, 2-mode, and 3-mode products.

In terms of n-mode products, SVD on a regular two-dimensional matrix (i.e., 2-order tensor), can be rewritten as follows [24]:

$$F = S \times_1 U^{(1)} \times_2 U^{(2)}, \quad (3)$$

where $U^{(1)} = (u_1^{(1)} u_2^{(1)} \ldots u_{I_1}^{(1)})$ is a *unitary* $(I_1 \times I_1)$-matrix, $U^{(2)} = (u_1^{(2)} u_2^{(2)} \ldots u_{I_1}^{(2)})$ is a *unitary* $(I_2 \times I_2)$-matrix, and S is a $(I_1 \times I_2)$-matrix with the properties of:

1. pseudodiagonality: $S = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_{\min\{I_1, I_2\}})$ and
2. ordering: $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min\{I_1, I_2\}} \geq 0$.

By extending this form of SVD, HOSVD of 3-order tensor $\mathcal{A}$ can be written as follows [24]:

$$\mathcal{A} = \mathcal{S} \times_1 U^{(1)} \times_2 U^{(2)} \times_3 U^{(3)}, \quad (4)$$

where $U^{(1)}$, $U^{(2)}$, and $U^{(3)}$ contain the orthonormal vectors (called the 1-mode, 2-mode, and 3-mode singular vectors, respectively) spanning the column space of the A_1, A_2, and A_3 matrix unfoldings. S is the core tensor and has the property of all orthogonality.

4 The Proposed Approach

We first provide the outline of our approach, which we name Tensor Reduction, through a motivating example. Next, we analyze the steps of the proposed algorithm. Finally, we apply a smoothing scheme in our approach.

4.1 Outline

In this section, we elaborate on how HOSVD is applied on tensors and on how the recommendation of items is performed according to the detected latent associations. Note that a similar approach is followed for the tag and user recommendations.

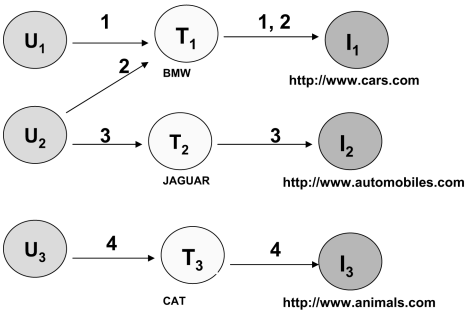


Fig. 2. Usage data of the running example.

When using a social tagging system, to be able to retrieve information items easily, a user u tags an item i with a tag t. After some time of usage, the tagging system accumulates a collection of usage data, which can be represented by a set of triplets $\{u, i, t\}$.

Our Tensor Reduction approach applies HOSVD on the 3-order tensor constructed from these usage data. In accordance with the HOSVD technique introduced in Section 3, the Tensor Reduction algorithm uses as input the usage data of $\mathcal{A}$ and outputs the reconstructed tensor $\hat{\mathcal{A}}$. $\hat{\mathcal{A}}$ measures the associations among the users, items, and tags. Each element of $\hat{\mathcal{A}}$ can be represented by a quadruplet $\{u, i, t, p\}$, where p measures the likeliness that user u will tag item i with tag t. Therefore, items can be recommended to u according to their weights associated with $\{u, t\}$ pair.

In this section, in order to illustrate how our approach works, we apply the Tensor Reduction algorithm to a running example. As illustrated in Fig. 2, three users tagged three different items (weblinks). In Fig. 2, the part of an arrow line (sequence of arrows with the same annotation) between a user and an item represents that the user tagged the corresponding item, and the part between an item and a tag indicates that the user tagged this item with the corresponding tag. Thus, the annotated numbers on the arrow lines gives the correspondence between the three types of objects. For example, user U_1 tagged item I_1 with tag "BMW," denoted as T_1. The remaining tags are "Jaguar," denoted as T_2, and "CAT," denoted as T_3.

From Fig. 2, we can see that users U_1 and U_2 have common interests on cars, while user U_3 is interested in cats. A 3-order tensor $\mathcal{A} \in R^{3 \times 3 \times 3}$, can be constructed from the usage data. We use the co-occurrence frequency (denoted as weight) of each triplet user, item, and tag as the elements of tensor $\mathcal{A}$, which are given in Table 1. Note that all associated weights are initialized to 1.

TABLE 1
The Elements of the Example Tensor

Arrow Line	User	Item	Tag	Weight
1	U_1	I_1	T_1	1
2	U_2	I_1	T_1	1
3	U_2	I_2	T_2	1
4	U_3	I_3	T_3	1

TABLE 2
The Elements of the Reconstructed Tensor

Arrow Line	User	Item	Tag	Weight
1	U_1	I_1	T_1	0.72
2	U_2	I_1	T_1	1.17
3	U_2	I_2	T_2	0.72
4	U_3	I_3	T_3	1
5	$\mathbf{U_1}$	$\mathbf{I_2}$	$\mathbf{T_2}$	**0.44**

After performing the Tensor Reduction analysis (details of how to do this are given in the following section), we can get the reconstructed tensor of $\hat{\mathcal{A}}$, which is presented in Table 2, whereas Fig. 3 depicts the contents of $\hat{\mathcal{A}}$ graphically (the weights are omitted). As shown in Table 2 and Fig. 3, the output of the Tensor Reduction algorithm for the running example is interesting, because a new association among these objects is revealed. The new association is between U_1, I_2, and T_2. This association is represented with the last (bold faced) row in Table 2 and with the dashed arrow line in Fig. 3).

If we have to recommend to U_1 an item for tag T_2, then there is no direct indication for this task in the original tensor $\mathcal{A}$. However, we see that in Table 2 the element of $\hat{\mathcal{A}}$ associated with $\{U_1, T_2, I_2\}$ is 0.44, whereas for U_1 there is no other element associating other tags with I_2. Thus, we recommend item I_2 to user U_1, who used tag T_2.

The resulting recommendation is reasonable, because U_1 is interested in cars rather than cats. That is, the Tensor Reduction approach is able to capture the latent associations among the multitype data objects: user, item, and tags. The associations can then be used to improve the item recommendation procedure, as will be verified by our experimental results.

Moreover, for purposes of tag recommendations, we can view our tensor from a different perspective. In particular, our tensor equivalently represents a quadruplet $\{u, i, t, p\}$ where p is the likeliness that user u will tag item i with tag t. Therefore, tags can be recommended to u according to their weights associated with $\{u, i\}$ pair. In our running example, if user U_1 is about to tag I_2, he will be recommended tag T_2.

Finally, for recommending users, our tensor can be viewed as a quadruplet $\{t, i, u, p\}$, where p is the likeliness that tag t will be used to label item i by the user u. Therefore, new users can be recommended for a tag t, according to their total weight, which results by aggregating all items, which are labeled with the same tag by the target user. In our running example, if user U_1 tagged item I_2 with tag T_2, he would receive user U_2 as user recommendation.

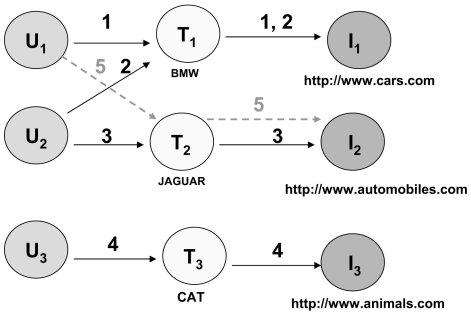


Fig. 3. Illustration of the Tensor Reduction Algorithm output for the running example.

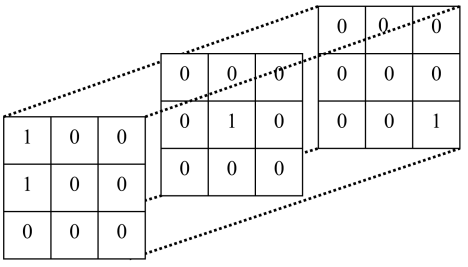


Fig. 4. The tensor construction of our running example.

4.2 The Tensor Reduction Algorithm

The Tensor Reduction algorithm initially constructs a tensor, based on usage data triplets $\{u, t, i\}$ of users, tags, and items. The motivation is to use all three entities that interact inside a social tagging system. Consequently, we proceed to the unfolding of $\mathcal{A}$, where we build three new matrices. Then, we apply SVD in each new matrix. Finally, we build the core tensor $\mathcal{S}$ and the resulting tensor $\hat{\mathcal{A}}$. All these can be summarized in six steps, as follows.

4.2.1 The Initial Construction of Tensor $\mathcal{A}$

From the usage data triplets (user, tag, and item), we construct an initial 3-order tensor $\mathcal{A} \in R^{u \times t \times i}$, where u, t, and i are the numbers of users, tags, and items, respectively. Each tensor element measures the preference of a (user u, tag t) pair on an item i. Fig. 4 presents the tensor construction of our running example.

4.2.2 Matrix Unfolding of Tensor $\mathcal{A}$

As described in Section 3, a tensor $\mathcal{A}$ can be matricized, i.e., to build matrix representations in which all the column (row) vectors are stacked one after the other. In our approach, the initial tensor $\mathcal{A}$ is matricized in all three modes. Thus, after the unfolding of tensor $\mathcal{A}$ for all three modes, we create three new matrices A_1, A_2, and A_3. In Fig. 5, we present the matrix unfoldings of our running example.

4.2.3 Application of SVD on Each Mode

We apply SVD on the three matrix unfoldings A_1, A_2, and A_3:

$$A_n = U^{(n)} \cdot S^{(n)} \cdot \left(V^{(n)}\right)^T, \quad 1 \le n \le 3. \tag{5}$$

For the running example, Figs. 6, 7, and 8 present these matrixes with the left singular vectors and the matrixes with the singular values for the decomposition in each mode (to

$$A_1 \in R^{I_u \times I_t I_i}, \qquad A_2 \in R^{I_t \times I_u I_i}, \qquad A_3 \in R^{I_u I_t \times I_i}$$

$$\mathrm{A_1} = \begin{bmatrix} 1&0&0&0&0&0&0&0&0 \\ 1&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1 \end{bmatrix}, \mathrm{A_2} = \begin{bmatrix} 1&1&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1 \end{bmatrix}, \mathrm{A_3} = \begin{bmatrix} 1&1&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1 \end{bmatrix}$$

Fig. 5. The tensor 1-mode, 2-mode, and 3-mode matrix unfoldings of our running example.

-0.53	0	-0.85
-0.85	0	0.53
0	1	0

(a)

1.62	0	0
0	1	0
0	0	0.62

(b)

Fig. 6. Example of (a) $U^{(1)}$ and (b) $S^{(1)}$.

1	0	0
0	1	0
0	0	1

(a)

1.41	0	0
0	1	0
0	0	1

(b)

Fig. 7. Example of (a) $U^{(2)}$ and (b) $S^{(2)}$.

1	0	0
0	1	0
0	0	1

(a)

1.41	0	0
0	1	0
0	0	1

(b)

Fig. 8. Example of (a) $U^{(3)}$ and (b) $S^{(3)}$.

ease presentation, we omit the corresponding matrixes with the right singular vectors).

4.2.4 Computing the Low-Rank Approximations

In matrix dimensionality reduction, low-rank approximation is used to filter out the small singular values that introduce "noise." Thus, SVD is truncated to the first c higher singular values and the corresponding singular vectors. This operation is called *thin-SVD* and is used in LSI [12]. The resulting matrix is denoted as rank-c approximation and SVD is optimal in the sense that it computes the rank-c approximation with the minimum *Frobenious norm*.

In the case of tensor dimensionality reduction, we have to compute a rank-c_1, c_2, c_3 approximation tensor, where c_i is the number of dimensions maintained for i-mode. To compute the rank-c_1, c_2, c_3 approximation, we retain c_i singular values and the corresponding left singular vectors from $U^{(i)}$, when applying SVD on the unfolded matrix A_i of i-mode. The selection of c_1, c_2, and c_3 determines the final dimensionality of the core tensor $\mathcal{S}$. Since each of the three diagonal singular matrices $S^{(1)}$, $S^{(2)}$, and $S^{(3)}$ are calculated by applying SVD on matrices A_1, A_2, and A_3, respectively, we use a different c_i value for each matrix $U^{(i)}$ $(1 \leq i \leq 3)$. This results to $(U^{(i)}_{c_i})$ matrixes, which denote the c_i-dimensionally reduced $U^{(i)}$ matrix $(1 \leq i \leq 3)$.

Determining the c_1, c_2, and c_3 parameters in Tucker models (like HOSVD) is a tedious task [1]. A practical option is to use ranks indicated by SVD on unfolded data in each mode. This way, c_1, c_2, and c_3 are chosen by preserving a percentage of information of the original $S^{(1)}$, $S^{(2)}$, and $S^{(3)}$ matrices after appropriate tuning. Our experimental results indicate that a 70 percent of the original diagonal of $S^{(1)}$, $S^{(2)}$, and $S^{(3)}$ matrices can give good approximations of A_1, A_2, and A_3 matrices. Notice that a percentage of the original diagonal can be obtain by summing the singular values of S matrix and then by keeping those singular values of S matrix that give us the wanted percentage. In our running example, the diagonal of $S^{(1)}$ matrix (see Fig. 6) sums to 3.24 $(1.62 + 1 + 0.62)$. Thus, by setting c_1 parameter equal to 2, we keep 80 percent (2.62/3.24) of the original diagonal of matrix $S^{(1)}$. Due to its simplicity and its efficiency, this approach has been followed in several related works that use tensor decompositions [23], [33]. For this reason we follow this approach too. In our running example, c_1 is equal to 2, c_2 is equal to 3, and c_3 is equal to 3. Fig. 9 presents the transposes of the dimensionally reduced $U^{(i)}$ matrixes.

-0.53	-0.85	0
0	0	1

$(U^{(1)}_{c_1})^T$

1	0	0
0	1	0
0	0	1

$(U^{(2)}_{c_2})^T$

1	0	0
0	1	0
0	0	1

$(U^{(3)}_{c_3})^T$

Fig. 9. The transposes of the dimensionally reduced $U^{(i)}$ matrices.

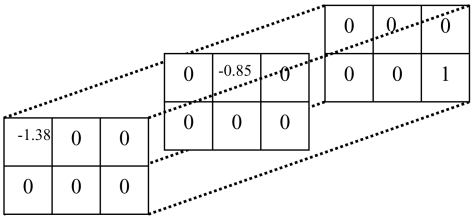


Fig. 10. The resulting core tensor $\mathcal{S}$ for the running example.

Nevertheless, it is worth of describing more systematic methods for determining these parameters. Cichocki et al. [9] proposed flexible cost functions to derive an optimal choice of a free parameter (parameter a) for the automatic rank approximation of a tensor. This method is proposed for dense tensors and signal processing applications (like image analysis), where the optimal choice of a depends [9] on the statistical distribution of data and the additive noise. For example, the optimal choice of the a parameter is different for the normal distribution $(a = 1)$ and the Poisson distribution $(a \to 0)$. However, this method cannot be directly used in STSs, because STSs's data are boolean and sparse and do not follow widely used statistical distributions. DIFFIT [35] is another approach, which enumerates all possible models and uses the differences between model fits to determine the c_1, c_2, and c_3 parameters. However, DIFFIT requires high computational complexity, a fact that makes it infeasible for most practical applications. Improvements of DIFFIT have been proposed [21], [6], which compare approximate model fit values rather than exact model fits.

Despite the existence of approximate methods, there is no straightforward way to find the optimal values for c_1, c_2, and c_3 [6] and, thus, several diagnostics should be used to have a true understanding of the structure of a multiway data set. For all the aforementioned reasons, we consider such an examination to comprise an open research problem of its own that is outside the scope of this manuscript.

4.2.5 The Core Tensor $\mathcal{S}$ Construction

The core tensor $\mathcal{S}$ governs the interactions among user, item and tag entities. From the initial tensor $\mathcal{A}$ we proceed to the construction of the core tensor S, as follows:

$$\mathcal{S} = \mathcal{A} \times_1 \left(U^{(1)}_{c_1}\right)^T \times_2 \left(U^{(2)}_{c_2}\right)^T \times_3 \left(U^{(3)}_{c_3}\right)^T. \quad (6)$$

Fig. 10 presents the core tensor $\mathcal{S}$ for the running example.

4.2.6 The Tensor $\hat{\mathcal{A}}$ Construction

Finally, tensor $\hat{\mathcal{A}}$ is built by the product of the core tensor $\mathcal{S}$ and the mode products of the three matrices $U^{(1)}_{c_1}$, $U^{(2)}_{c_2}$, and $U^{(3)}_{c_3}$ as follows:

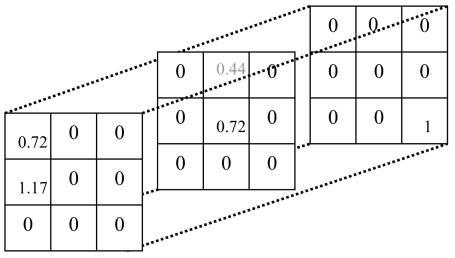


Fig. 11. The resulting $\hat{\mathcal{A}}$ tensor for the running example.

$$\hat{\mathcal{A}} = \mathcal{S} \times_1 U^{(1)}_{c_1} \times_2 U^{(2)}_{c_2} \times_3 U^{(3)}_{c_3}. \quad (7)$$

For the current example, the resulting $\hat{\mathcal{A}}$ tensor is presented in Fig. 11.

4.2.7 The Generation of the Recommendations

The reconstructed tensor $\hat{\mathcal{A}}$ measures the associations among the users, tags, and items, so that each element of $\hat{\mathcal{A}}$ represents a quadruplet $\{u, t, i, p\}$, where p is the likeliness that user u will tag item i with tag t.

On this basis, items can be recommended to u according to their weights associated with $\{u, t\}$ pair. However, we see that in Fig. 11, the element of $\hat{\mathcal{A}}$ associated with $\{U_1, I_2, T_2\}$ is 0.44, whereas for U_1 there is no other element associating other tags with I_2. Thus, in our running example, we recommend to user U_1 item I_2 for tag T_2. Analogous approach can be applied for the recommendation of tags or users, as already described in Section 4.1.

4.2.8 Execution Requirements for Tensor Reduction

To provide recommendations, our approach contains an offline part, which computes the HOSVD, and an online part, for retrieving weights from the reconstructed tensor and forming the recommendations. The latter part, which affects the actual experience of users in real applications, can be done in real time.

Regarding the offline part, in real applications (with large number of users, items, and tags), a key characteristic is sparsity, meaning that most of entries in the tensor are zeros. Therefore, the computation of HOSVD problem boils down to: 1) calculating the leading singular vectors of large, sparse mode unfolding matrixes, and 2) the computation of product of a sparse tensor times a series of dense matrices (for the reconstructed tensor).

The bottleneck of memory overflow during this procedure has been addressed by Kolda and Sun [23], who proposed an implementation framework, called MET, which maximizes the computation speed while optimally utilizing the available memory. This way, very large tensors can be stored using only moderate hardware. Based on the results reported by Kolda and Sun [23], we validated that for very large, 100K-by-100K-by-100K random tensors with 1M nonzero elements, the computation of HOSVD requires less than 200 sec and 4 MB of RAM. For the real data sets we examined, since their size is moderate, execution times were less than half minute. For real-world applications, the use of parallel architectures can further reduce execution times [23].

4.3 Smoothing with Kernel SVD

In Section 4.2.3, we described the application of SVD on the three matrix unfoldings A_1, A_2, and A_3, which results to the three matrixes $U^{(1)}$, $U^{(2)}$, and $U^{(3)}$ that contain the orthonormal vectors (left singular vectors) for each mode. As already mentioned, sparsity is a severe problem in three-dimensional data and it can affect the outcome of SVD. To address this problem, instead of SVD we can apply kernel-SVD [8], [10] in the three unfolded matrices. Kernel-SVD is the application of SVD in the Kernel-defined feature space.

For each unfolding A_i $(1 \leq i \leq 3)$ we have to nonlinearly map its contents to a higher dimensional space using a mapping function ϕ. Therefore, from each A_i matrix we can derive an F_i matrix, where each element a_{xy} of A_i is mapped to the corresponding element f_{xy} of F_i, i.e., $f_{xy} = \phi(a_{xy})$. Next, we can apply SVD and decompose each F_i as follows:

$$F_i = U^{(i)} S^{(i)} \left(V^{(i)}\right)^T. \quad (8)$$

The resulting $U^{(i)}$ matrixes are then used to construct the core tensor, that is, the procedure continues as described in Section 4.2.4.

Nevertheless, to avoid the explicit computation of F_i, all computations must be done in the form of inner products. In particular, as we are interested to compute only the matrixes with the left singular vectors, for each mode i we can define a matrix B_i as follows:

$$B_i = F_i F_i^T. \quad (9)$$

As B_i is computed using inner products from F_i, we can substitute the computation of inner products with the results of a kernel function. This technique is called the "kernel trick" [10] and avoids the explicit (and expensive) computation of F_i. As each $U^{(i)}$ and $V^{(i)}$ are orthogonal and each $S^{(i)}$ is diagonal, it easily follows from (8) and (9) that [26]

$$B_i = \left(U^{(i)} S^{(i)} \left(V^{(1)}\right)^T\right)\left(U^{(i)} S^{(i)} \left(V^{(i)}\right)^T\right)^T = U^{(i)} \left(S^{(i)}\right)^2 \left(U^{(i)}\right)^T. \quad (10)$$

Therefore, each required $U^{(i)}$ matrix can be computed by diagonalizing each B_i matrix (which is square) and taking its eigen-vectors.

Regarding the kernel function, in our experiments we use the Gaussian kernel $K(x, y) = e^{-\frac{\|x-y\|^2}{c}}$, which is commonly used in many applications of kernel SVD. As Gaussian Kernel parameter c, we use the estimate for standard deviation in each matrix unfolding.

4.4 Inserting New Users, Tags, or Items

As new users, tags, or items are being introduced to the system, the $\hat{\mathcal{A}}$ tensor, which provides the recommendations, has to be updated. The most demanding operation for this task is the updating of the SVD of the corresponding mode in (5)-(7). We can avoid the costly batch recomputation of the corresponding SVD, by considering incremental solutions [30], [4]. Depending on the size of the update (i.e., number of new users, tags, or items), different techniques have been followed in related research. For small update sizes we can consider the *folding-in* technique [12], [30], whereas for larger update sizes we can consider Incremental SVD techniques [4]. Both techniques are described in the following. (Notice that recently Sun et al. [33] described an incremental procedure, which however applies when

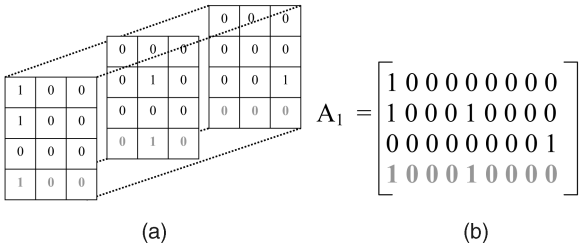


Fig. 12. Example of folding in a new user. (a) The insertion of a new user in the tensor. (b) The new 1-mode unfolded matrix A_1.

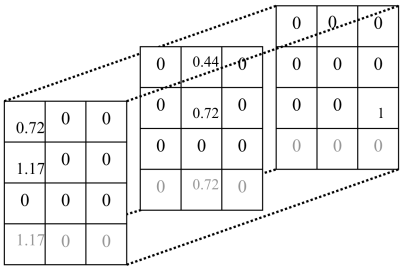


Fig. 14. The resulting $\hat{\mathcal{A}}$ tensor of the running example after the insertion of the new user.

new tensors arrive as time passes, not for new users, items, or tags within a single tensor.)

4.4.1 Update by Folding-In

Given a new user, we first compute the new 1-mode matrix unfolding A_1. It is easy to see that the entries of the new user result to the appending of a new row in A_1. This is exemplified in Fig. 12. Fig. 12a shows the insertion of a new user in the tensor of the current example (the new values are presented with red color). Notice that to ease the presentation, the new user's tags and items are identical with those of user U_2.

Let $\mathbf{u}$ denote the new row that is appended to A_1. Fig. 12b presents the new A_1, i.e., the 1-mode unfolded matrix, where it is shown that the contents of $\mathbf{u}$ (highlighted with red color) have been appended as a new row in the end of A_1.

Since A_1 changed, we have to compute its SVD, as given by (5). To avoid batch SVD recomputation, we can use the existing basis $U_{c1}^{(1)}$ of left singular vectors, to project the $\mathbf{u}$ row onto the reduced $c1$-dimensional space of users in the A_1 matrix. This projection is called folding-in and is computed by using the following equation [12]:

$$\mathbf{u_{new}} = \mathbf{u} \cdot V_{c1}^{(1)} \cdot \left(S_{c1}^{(1)}\right)^{-1}. \tag{11}$$

In (11), $\mathbf{u_{new}}$ denotes the mapped row, which will be appended to $U_{c1}^{(1)}$, whereas $V_{c1}^{(1)}$ and $(S_{c1}^{(1)})^{-1}$ are the dimensionally reduced matrixes derived when SVD was originally applied to A_1, i.e., before the insertion of the new user. In the current example, the computation of $\mathbf{u_{new}}$ is described in Fig. 13.

The $\mathbf{u_{new}}$ vector should be appended in the end of the $U_{c1}^{(1)}$ matrix. For the current example, appending should be done to the previously $U_{c1}^{(1)}$ matrix, whose transpose is shown in Fig. 10. Notice that in this example, $\mathbf{u_{new}}$ is identical with the second column of the transpose of $U_{c1}^{(1)}$. The reason is that the new user has identical tags and items with user U_2 and we mapped them on the same space (recall that the folding-in technique maintains the same space computed originally by SVD).

Finally, to update the $\hat{\mathcal{A}}$ tensor, we have to perform the products given in (9). Notice that only $U^{(1)_{c1}}$ has been modified in this equation. Thus, to optimize the insertion of new users, as mode products are interchangeable, we can perform this product as $[\mathcal{S} \times_2 U_{c_2}^{(2)} \times_3 U_{c_3}^{(3)}] \times_1 U_{c_1}^{(1)}$, where the left factor (inside the brackets), which is unchanged, can be prestored so as to avoid its recomputation. For the current example, the resulting $\hat{\mathcal{A}}$ tensor is shown in Fig. 14.

Analogous insertion procedure can be followed for the insertion of a new item or tag. For a new item insertion, we have to apply (11) on the 2-mode matrix unfolding (A_2) of tensor $\mathcal{A}$, while for a new tag we apply (11) on the 3-mode matrix unfolding (A_3) of tensor $\mathcal{A}$.

4.4.2 Update by Incremental SVD

Folding-in incrementally updates SVD, but the resulting model is not a perfect SVD model, because the space is not orthogonal [30]. When the update size is not big, loss of orthogonality may not be a severe problem in practice. Nevertheless, for larger update sizes, the loss of orthogonality may result into an inaccurate SVD model. In this case, we need to incrementally update SVD so as to ensure orthogonality. This can be attained in several ways. Next, we describe how to use the approach proposed by Brand [4].

Let $M_{p\times q}$ be a matrix, upon which we apply SVD and maintain the first r singular values, i.e.,

$$M_{p\times q} = U_{p\times r} S_{r\times r} V_{r\times q}^T.$$

Assume that each column of matrix $C_{p\times c}$ contains the additional elements. Let $L = U \backslash C = U^T C$ be the projection of C onto the orthogonal basis of U. Let also $H = (I - UU^T)C = C - UL$ be the component of C orthogonal to the subspace spanned by U (I is the identity matrix).

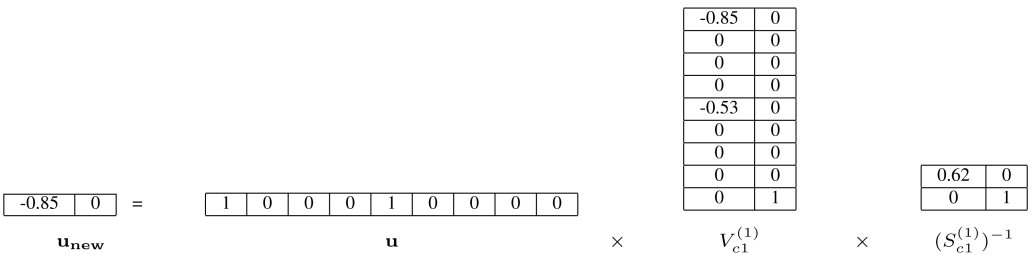


Fig. 13. The result of folding-in for the current example.

Finally, let J be an orthogonal basis of H and let $K = J\backslash H = J^T H$ be the projection of C onto the subspace orthogonal to U. Consider the following identity:

$$\begin{aligned}[U\ J]&\begin{bmatrix} S & L \\ 0 & K \end{bmatrix}\begin{bmatrix} V & 0 \\ 0 & I \end{bmatrix}^T \\ &= [U(I - UU^T)C/K]\begin{bmatrix} S & U^TC \\ 0 & K \end{bmatrix}\begin{bmatrix} V & 0 \\ 0 & I \end{bmatrix}^T \\ &= [USV^T\ C] = [M\ C].\end{aligned}$$

Like an SVD, the left and right matrixes in the product are unitary and orthogonal. The middle matrix, denoted as Q, is diagonal. To incrementally update the SVD, Q must be diagonalized. If we apply SVD on Q we get

$$Q = U'S'(V')^T.$$

Additionally, define U'', S'', and V'' as follows:

$$U'' = [U\ J]U', \quad S'' = S', \quad V'' = \begin{bmatrix} V & 0 \\ 0 & I \end{bmatrix}V'.$$

Then, the updated SVD of matrix $[M\ C]$ is

$$[M\ C] = [USV^T\ C] = U''S''(V'')^T.$$

This incremental update procedure takes $O((p+q)r^2 + pc^2)$ time [4].

Returning to the application of incremental update for new users, items, or tags, as described in Section 4.4.1, in each case, we result with a number of new rows that are appended in the end of the unfolded matrix of the corresponding mode. Therefore, we need an incremental SVD procedure in the case where we add new rows, whereas the aforementioned method works in the case where we add new columns. In this case, we simply swap U for V and U'' for V''.

5 Experimental Evaluation

In this section, in the area of item recommendations, we compare experimentally our approach with state-of-the-art item recommendation algorithms. Henceforth, our proposed approach is denoted as Tensor Reduction. We use in the comparison the tag-aware Fusion algorithm [36] and the Item-based CF algorithm [29], denoted as Fusion and Item based, respectively. We also include in the comparison a CF algorithm based on Latent Semantic Indexing [34], denoted as Matrix SVD. Moreover, in the area of tag recommendations, we compare our approach with state-of-the-art tag recommendation algorithms. We use in the comparison the Folkrank [17] and the Collaborative Tag Suggestions [39] (known as Penalty-Reward algorithm) algorithms, denoted as FolkRank and PR, respectively. Finally, in the area of user recommendations, we compare our approach with a baseline algorithm similar to Amazon.com's user recommendation method, denoted as BaseLine algorithm (BL).

Our experiments were performed on a 3 GHz Pentium IV, with 1 GB of memory, running Windows XP. The tensor construction and processing is implemented in Matlab. All algorithms were implemented in C++ and their parameters were tuned according to the original papers.

To evaluate the examined algorithms, we have chosen real data sets from two different STSs: BibSonomy and Last.fm, which have been used as benchmarks in past works [17].

BibSonomy. We used a snapshot of all users, items (both publication references and bookmarks) and tags publicly available on April 30, 2007. From the snapshot, there are excluded the posts from the DBLP computer science bibliography since they are automatically inserted and all owned by one user and all tagged with the same tag (dblp). The number of users, items and tags is 1,037, 28,648, and 86,563, respectively.

Last.fm. The data for Last.fm were gathered during October 2007, partly through the web services API (collecting user nicknames), partly crawling the Last.fm site. Here, the items correspond to artist names, which are already normalized by the system. There are 12,773 triplets in the form user-artist-tag. To these triplets correspond 4,442 users, 1,620 artists, and 2,939 tags.

Following the approach of [17] to get more dense data, we adapt the notion of a p-core to tripartite hypergraphs. The p-core of level k has the property, that each user, tag, and item has/occurs in at least k posts. For both data sets, we used $\mathrm{k} = 5$. Thus, for the Bibsonomy data set there are 105 users, 246 items, and 591 tags, whereas for the Last.fm data set there are 112 users, 234 items, and 567 tags.

5.1 Experimental Protocol and Evaluation Metrics

For the item and tag recommendations, all algorithms had the task to predict the items/tags of the users' postings in the test set. We performed fourfold cross validation, thus, each time we divide the data set into a training set and a test set with sizes 75 and 25 percent of the original set, respectively.

Based on the approach of [18], [16], a more realistic evaluation of recommendation should consider the division of tags/items of each test user into two sets: 1) the *past* tags/items of the test user, and 2) the *future* tags/items of the test user. Therefore, for a test user we generate the recommendations based only on the tags/items in his past set. The default sizes of the past and future sets are 50 and 50 percent, respectively, of the number of tags posted by each test user.

As performance measures for item and tag recommendations, we use the classic metrics of precision and recall. For a test user that receives a list of N recommended tags (top-N list), precision and recall are defined as follows:

- *Precision* is the ratio of the number of relevant tags in the top-N list (i.e., those in the top-N list that belong in the future set of tags posted by the test user) to N.
- *Recall* is the ratio of the number of relevant tags in the top-N list to the total number of relevant tags (all tags in the future set posted by the test user).

For the user recommendations, we do not use precision-recall metrics because there is no information in the data sets about which users are similar with who. That is, our data sets do not include explicitly for a target user his similar users. Thus, we cannot verify our recommendation results with metrics such as precision and recall.

To evaluate the effectiveness of Tensor Reduction and BL algorithms in recommending *interesting* users, we compute

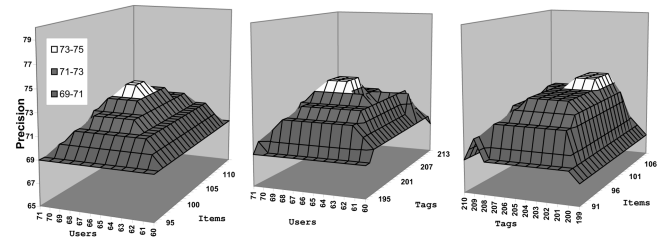


Fig. 15. Precision of Tensor Reduction as dimensions of core tensor vary for BibSonomy data set.

the item similarity within the recommended users [25]. This evaluation is based on the fact that users with shared interests are very likely to tag similar items.

A metric to evaluate this characteristic of each Neighborhood N of recommended users is to compute the average cosine similarity (ACS) of all item pairs inside the Neighborhood of users with common social interest [25]:

$$ACS_N = \frac{\sum_{u,v \in N} \sum_{i \in I(u), j \in I(v)} sim(i,j)}{\sum_{u,v \in N} |I(u)||I(v)|},$$

where for a user $u, I(u)$ denotes the items tagged by u. ACS_N evaluates the tightness or looseness of each Neighborhood or recommended users.

5.2 Influence of the Core Tensor Dimensions and the Smoothing Scheme

We first conduct experiments to study the influence of the core tensor dimensions on the performance of our Tensor Reduction algorithm. If one dimension of the core tensor is fixed, we can find the recommendation accuracy varies as the other two dimensions change, as shown in Fig. 15. The vertical axes denote the precision and the other two axes denote the corresponding dimensions. For the leftmost figure, the tag dimension is fixed at 200 and the other two dimensions change. For the middle figure, the item dimension is fixed at 105. For the rightmost figure, the user dimension is fixed at 66.

Our experimental results indicate that a 70 percent of the original diagonal of $S^{(1)}$, $S^{(2)}$, and $S^{(3)}$ matrices can give good approximations of A_1, A_2, and A_3 matrices. Thus, the numbers c_1, c_2, and c_3 of left singular vectors of matrices $U^{(1)}$, $U^{(2)}$, and $U^{(3)}$ after appropriate tuning are set to 66, 105, and 200 for the BibSonomy data set, whereas are set to 40, 80, and 190 for the Last.fm data set.

Next, we study the influence of the proposed Kernel smoothing scheme on the recommendation accuracy of our Tensor Reduction algorithm in terms of precision. We present our experimental results in Figs. 16a and 16b, for both the BibSonomy and Last.fm data sets. As shown, our smoothing Kernel method can improve the performance accuracy. The results are consistent in both data sets.

5.3 Item Recommendations

5.3.1 Algorithms' Settings

For each of the algorithms of our evaluation we will now describe briefly the specific settings used to run them:

- **Fusion algorithm:** We have varied the λ parameter from 0 to 1 by an interval of 0.1 and the neighborhood k parameter from 10 to 150 by an interval of 10. We have found the best λ to be 0.3 and k to be 20.
- **Matrix SVD algorithm:** Regarding the application of SVD on user-item matrix, we preserved, each time, a different fraction of principal components of the SVD model. More specifically, we preserve 90, 70, and 50 percent of the total information of initial user-item matrix. Our results show that 50 percent is adequate for producing a good approximation of the original matrix. Then, in the reduced model, we apply the user-based CF algorithm.
- **Item-based algorithm:** We have varied the neighborhood k parameter from 10 to 300 by an interval of 10. We found the best k to be 40.
- **Tensor Reduction algorithm:** Our tensor reduction algorithm is modified appropriately to recommend items to a target user. In particular, our tensor represents a quadruplet $\{u, t, i, p\}$ where p is the likeliness that user u will tag item i with tag t.

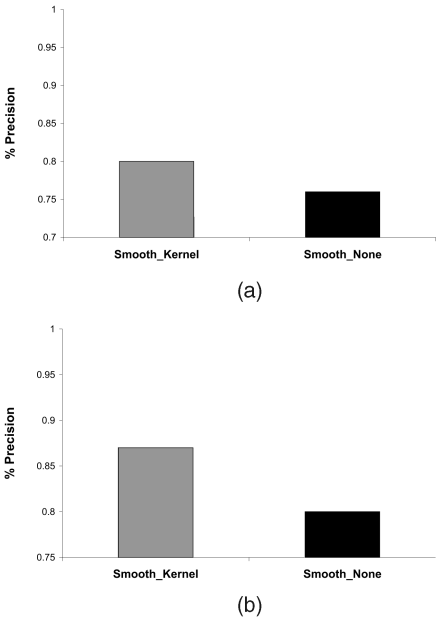


Fig. 16. Precision of Tensor Reduction with and without smooth scheme for (a) BibSonomy and (b) Last.fm.

5.3.2 Results

In this section, we proceed with the comparison of Tensor Reduction with Fusion, Matrix SVD, and Item based, in terms of precision and recall. This reveals the robustness of each algorithm in attaining high recall with minimal losses in terms of precision. We examine the top-N ranked list, which is recommended to a test user, starting from the top item. In this situation, the recall and precision vary as we proceed with the examination of the top-N list.

For the BibSonomy data set (N is between [1..5]), in Fig. 17a, we plot a precision versus recall curve for all three algorithms. As shown, all algorithms' precision falls as N increases. In contrast, as N increases, recall for all five algorithms increases too. Tensor Reduction algorithm attains 80 percent precision, when we recommend a top-1 list of tags. In contrast, Fusion gets a precision of almost 60 percent. Moreover, Tensor Reduction is more effective than Fusion

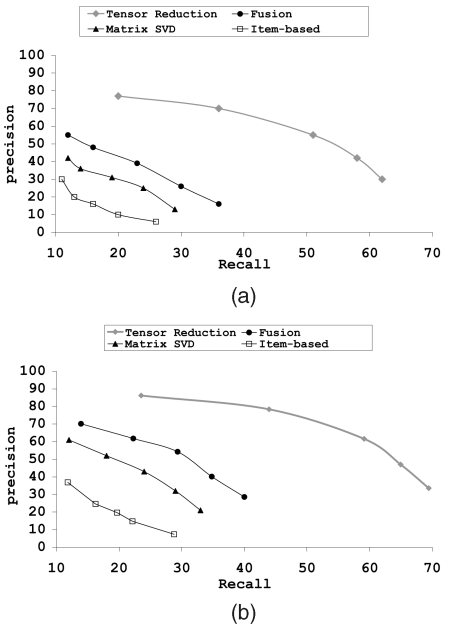


Fig. 17. Comparison of Tensor Reduction, Fusion, Matrix SVD, and Item-based algorithms for the (a) BibSonomy data set and (b) Last.fm data set.

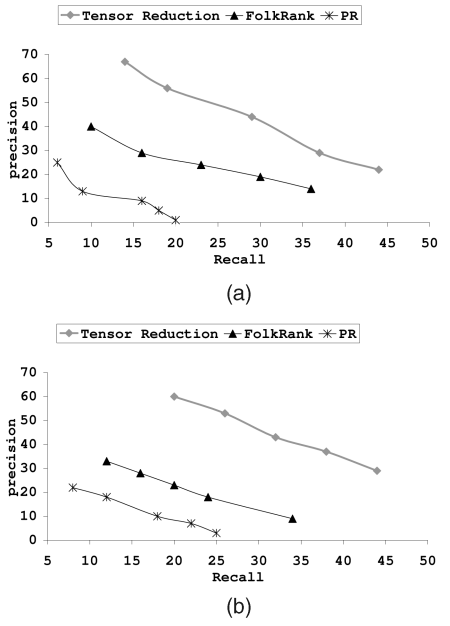


Fig. 18. Comparison of Tensor Reduction, Folkrank, and PR algorithms for the (a) BibSonomy data set and (b) Last.fm data set.

getting a maximum recall of 64 percent, while the latter's is 36 percent. This experiment shows that Tensor Reduction is more robust in finding relevant tags for the test user. The reason is that Tensor Reduction exploits all information that concerns the three objects (users, items, and tags), and through HOSVD, it addressed sparsity and finds latent associations. Item-based and Matrix SVD algorithms present the worst results, because they do not exploit all the existing information (they are applied in two-dimensional data).

For the Last.fm data set (N is between [1..5]), in Fig. 17b, we plot also a precision versus recall curve for all three algorithms. Tensor Reduction algorithm again attains the best performance. Despite the different nature of the two data sets (the one is for bibliographic data and the other for musical data), we observe similar behavior of algorithms for both data sets. It is important that Tensor Reduction provides more accurate recommendations in both cases.

5.4 Tag Recommendations

5.4.1 Algorithms' Settings

For each of the algorithms of our evaluation, we will now describe briefly the specific settings used to run them:

- **FolkRank algorithm:** We set the damping factor $d = 0.7$ and stopped computation after 100 iterations or when the distance between two consecutive weight vectors was less than 10^{-6}. For the preference vector p, we gave higher weights to the user and the item from the post which was chosen. While each user, tag, and resource got a preference weight of 1, the user and resource from that particular post got a preference weight of $1 + |U|$ and $1 + |I|$, respectively.
- **PR algorithm:** Initially, we set the uniform authority score for each user equal to 1.0. The authority score $a(u)$ is computed via an iterative algorithm similar to HITS.
- **Tensor Reduction algorithm:** Our tensor reduction algorithm is modified appropriately to recommend tags to a target user. In particular, our tensor represents a quadruplet $\{u, i, t, p\}$, where p is the likeliness that user u will tag item i with tag t.

5.4.2 Results

In this section, we proceed with the comparison of Tensor Reduction with FolkRank, and PR, in terms of precision and recall.

For the BibSonomy data set (N is between [1..5]), in Fig. 18a, we plot a precision versus recall curve for all three algorithms. Tensor Reduction algorithm attains 68 percent precision, when we recommend a top-1 list of tags. In contrast, FolkRank gets a precision of 42 percent. Moreover, Tensor Reduction is more effective than FolkRank getting a maximum recall of 44 percent, while the latter's is 36 percent. The reason is that Tensor Reduction exploits all information that concerns the three objects (users, items, and tags), and through HOSVD, it addressed sparsity and finds latent associations.

For the Last.fm data set (N is between [1..5]), in Fig. 18b, we plot also a precision versus recall curve for all three algorithms. Tensor Reduction algorithm again attains the best performance.

5.5 User Recommendations

5.5.1 Algorithms' Settings

For each of the algorithms of our evaluation, we will now describe briefly the specific settings used to run them:

- **Baseline algorithm (BL):** BL algorithm is quite similar to Amazon.com's method to recommend *interesting* users to a target user. BL logic is as follows: if a user uses a specific tag for item search, then he is recommended (except of recommended items) also *interesting* users, whose profiles are considered similar to him. These recommended users must have

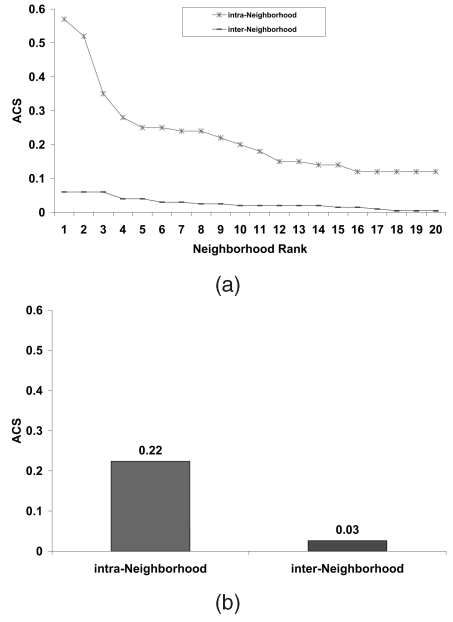


Fig. 19. Comparison of intra-Neighborhood and inter-Neighborhood similarity of Tensor Reduction Algorithm for the BibSonomy data set.

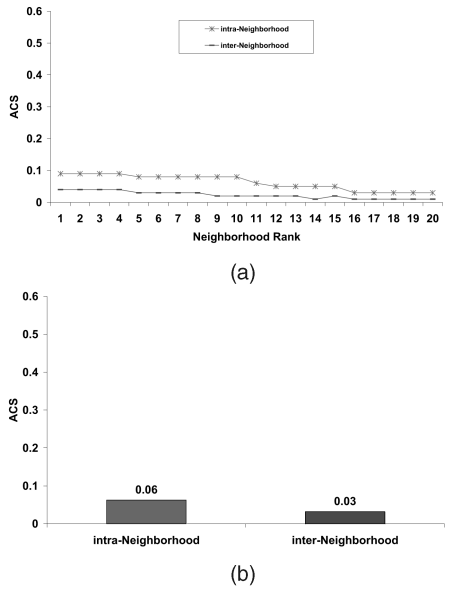


Fig. 20. Comparison of intra-Neighborhood and inter-Neighborhood similarity of BL Algorithm for the BibSonomy data set.

used the specific tag and are ranked based on how many times they used it. The basic idea behind this simple algorithm is that a tag corresponds in a topic of common interest. Thus, users that use the same tag could be interested in a common topic, forming a community of common interest.

- **Tensor Reduction algorithm:** Our tensor reduction algorithm is modified appropriately to recommend Neighborhoods of users to a target user. In particular, our tensor represents a quadruplet $\{t, i, u, p\}$ where p is the likeliness that tag t will be used to label item i by the user u.

5.5.2 Results

In this section, we evaluate the effectiveness of Tensor Reduction and BL algorithms in recommending *interesting* users. We compute the item similarity within the recommended neighborhoods of users [25]. This evaluation is based on the fact that users with shared interests are very likely to tag similar items. Note that, some of the recommended neighborhoods can be consisted of users that are quite related, while others are consisted of users that are less related.

We focus only on the BibSonomy data set, because in this data set users tag web pages, for which we can apply a commonly used similarity measure. Specifically, we crawled for each web site the first page and preprocess it to create a vector of terms. Preprocessing involved the removal of stop words, stemming, and TF/IDF. Then, we find correlation between two web sites based on the keyword terms they include. We compute the similarity between two web sites with the inner product, i.e., the cosine similarity of their TF/IDF keyword term vectors [25].

For each user's neighborhood, we compute the ACS of all web site pairs inside the neighborhood (20 nearest neighbors), called intra-Neighborhood similarity. We also randomly select 20 neighborhood pairs among the 105 user neighborhoods and compute the average pairwise web site similarity between every two neighborhoods, called inter-Neighborhood similarity.

Fig. 19a shows the comparison between the intra-Neighborhood and the inter-Neighborhood similarity of our Tensor Reduction Algorithm. In this figure, x axis is the rank of neighborhoods similarity, sorted by the descending order of their intra-Neighborhood similarities. y-axis shows the intra-Neighborhood similarity of each neighborhood and the corresponding average inter-Neighborhood similarity of this neighborhood with other 20 randomly selected neighborhoods. As we can see, for all users' neighborhoods, the intra-Neighborhood similarity is consistently higher than the average inter-Neighborhood. As also shown in Fig. 19b, the average intra-Neighborhood similarity across all neighborhoods is 0.22 with standard deviation equal to 0.091, while the average of inter-Neighborhood similarities is only 0.03 with standard deviation equal to 0.017.

Corresponding to Figs. 19a and 19b, we show for the BL algorithm the comparison of intra- and inter-Neighborhood similarity for each neighborhood, and the average intra- and inter-Neighborhood similarity for all neighborhoods in Figs. 20a and 20b, respectively. As we can see, BL's intra- and inter-Neighborhood similarity values are very close. This means, that BL fails to recommend coherent and related neighborhoods of users. In addition, our Tensor Reduction algorithm attains at least three times higher ACS than BL. That is, our approach recommends neighborhoods of users that are more related, while BL recommends users which are less relevant.

5.6 Discussion

In real-world social tagging systems, the size of the resulting 3-order tensor, despite its sparsity, may be potentially huge. By analyzing the Tensor Reduction algorithm, we can find that most time is consumed when SVD is performed on the three unfolded matrices. If the tensor scale is very large, this step is time consuming. However, this computation can be performed offline in advance. In cases of large scale tensors, it is also interesting to examine randomized tensor

factorization algorithms, like the ones proposed by Drineas and Mahoney [11], which can handle very large tensors with a guaranteed quality-of-approximation bound. More recently, Tensor-CUR decomposition has been proposed [27] that expresses the original tensor in terms of a basis consisting of underlying subtensors. This method can comprise an alternative solution to tensor factorization, which can handle tensors with large size.

Matrix (i.e., 2-order tensor) factorization has been applied to recommender systems [30], [34]. The results in Section 5.3.2 (see Figs. 17a and 17b) indicate the superiority of tensor factorization over matrix factorization in terms of accuracy, as the former exploits the ternary relation of data and captures the latent associations among the multitype objects. This result is in accordance to existing results in other application domains of tensor factorization [32], [38], which show its superiority against matrix factorization methods (LSI or SVD).

Finally, regarding the online incremental updating methods that are described in Section 4.4, the folding-in method performs fast updating but may result in loss of orthogonality. Existing results in [30] study the implications of this issue for the case of matrix (user-item) based recommendations, showing that accuracy is not affected much. Analogous conclusions are expected for the case of tensors, but due to lack of space, we do not elaborate further on this case. However, we have to note that the incremental SVD method is also fast and preserves the orthogonality. Therefore, it has guaranteed accuracy while offering time efficiency.

6 Conclusions

Social tagging systems provide recommendations to users based on what tags other users have used on items. In this paper, we developed a unified framework to model the three types of entities that exist in a social tagging system: users, items, and tags. We examined multiway analysis on data modeled as 3-order tensor, to reveal the latent semantic associations between users, items, and tags. The multiway latent semantic analysis and dimensionality reduction is performed by combining the HOSVD method with the Kernel-SVD smoothing technique. Our approach improves recommendations by capturing users multimodal perception of item/tag/user. Moreover, we study a problem of how to provide user recommendations, which can have significant applications in real systems but which have not been studied in depth so far in related research. We also performed experimental comparison of the proposed method against state-of-the-art recommendations algorithms, with two real data sets (Last.fm and BibSonomy). Our results show significant improvements in terms of effectiveness measured through recall/precision. As future work, we intend to examine different methods for extending SVD to high-order tensors such as the Parallel Factor Analysis. We also indent to apply different weighting methods for the initial construction of a tensor. A different weighting policy for the tensor's initial values could improve the overall performance of our approach.

Acknowledgments

The authors thank Mr. Tat-Jun Chin for providing his implementation of Kernel SVD method. The second author gratefully acknowledge the partial cofunding of his work through the European Commission FP7 project MyMedia (www.mymediaproject.org) under the grant agreement no. 215006.

References

[1] E. Acar and B. Yener, "Unsupervised Multiway Data Analysis: A Literature Survey," *IEEE Trans. Knowledge and Data Eng.,* vol. 21, no. 1, pp. 6-20, Jan. 2009.

[2] N. Ali-Hasan and A. Adamic, "Expressing Social Relationships on the Blog through Links and Comments," *Proc. Int'l Conf. Weblogs and Social Media (ICWSM),* 2007. Q3

[3] M. Berry, S. Dumais, and G. O'Brien, "Using Linear Algebra for Intelligent Information Retrieval," *SIAM Rev.,* vol. 37, no. 4, pp. 573-595, 1994.

[4] M. Brand, "Incremental Singular Value Decomposition of Uncertain Data with Missing Values," *Proc. European Conf. Computer Vision (ECCV '02),* 2002.

[5] J. Breese, D. Heckerman, and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," *Proc. Conf. Uncertainty in Artificial Intelligence,* pp. 43-52, 1998.

[6] E. Ceulemans and H.A.L. Kiers, "Selecting among Three-Mode Principal Component Models of Different Types and Complexities: A Numerical Convex-Hull Based Method," *British J. Math. and Statistical Psychology,* vol. 59, no. 1, pp. 133-150, 2006.

[7] S. Chen, F. Wang, and C. Zhang, "Simultaneous Heterogeneous Data Clustering Based on Higher Order Relationships," *Proc. Workshop Mining Graphs and Complex Structures (MGCS '07), in conjunction with IEEE Int'l Conf. Data Mining (ICDM '07),* pp. 387-392, 2007.

[8] T. Chin, K. Schindler, and D. Suter, "Incremental Kernel SVD for Face Recognition with Image Sets," *Proc. Int'l Conf. Automatic Face and Gesture Recognition (FGR),* pp. 461-466, 2006.

[9] A. Cichocki, R. Zdunek, S. Choi, R. Plemmons, and S. Amari, "Non-Negative Tensor Factorization Using Alpha and Beta Divergences," *Proc. IEEE Int'l Conf. Acoustics, Speech and Signal Processing (ICCASP '07),* 2007.

[10] N. Cristianini and J. Shawe-Taylor, *Kernel Methods for Pattern Analysis.* Cambridge Univ. Press, 2004.

[11] P. Drineas and M.W. Mahoney, "A Randomized Algorithm for a Tensor-Based Generalization of the Singular Value Decomposition," Technical Report YALEU/DCS/TR-1327, 2005.

[12] G. Furnas, S. Deerwester, and S. Dumais, "Information Retrieval Using a Singular Value Decomposition Model of Latent Semantic Structure," *Proc. ACM SIGIR Conf.,* pp. 465-480, 1988.

[13] S. Golder and B. Huberman, "The Structure of Collaborative Tagging Systems," technical report, 2005.

[14] H. Halpin, V. Robu, and H. Shepherd, "The Complex Dynamics of Collaborative Tagging," *Proc. 16th Int'l Conf. World Wide Web (WWW '07),* pp. 211-220, 2007.

[15] J. Herlocker, J. Konstan, and J. Riedl, "An Empirical Analysis of Design Choices in Neighborhood-Based Collaborative Filtering Algorithms," *Information Retrieval,* vol. 5, no. 4, pp. 287-310, 2002.

[16] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl, "Evaluating Collaborative Filtering Recommender Systems," *ACM Trans. Information Systems,* vol. 22, no. 1, pp. 5-53, 2004.

[17] A. Hotho, R. Jaschke, C. Schmitz, and G. Stumme, "Information Retrieval in Folksonomies: Search and Ranking," *The Semantic Web: Research and Applications,* pp. 411-426, Springer, 2006. Q4

[18] Z. Huang, H. Chen, and D. Zeng, "Applying Associative Retrieval Techniques to Alleviate the Sparsity Problem in Collaborative Filtering," *ACM Trans. Information Systems,* vol. 22, no. 1, pp. 116-142, 2004.

[19] R. Jaschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme, "Tag Recommendations in Folksonomies," *Proc. Knowledge Discovery in Databases (PKDD '07),* pp. 506-514.

[20] G. Karypis, "Evaluation of Item-Based Top-N Recommendation Algorithms," *Proc. ACM Conf. Information and Knowledge Management (CIKM),* pp. 247-254, 2001.

[21] H.A.L. Kiers and A.D. Kinderen, "A Fast Method for Choosing the Numbers of Components in Tucker3 Analysis," *British J. Math. and Statistical Psychology,* vol. 56, no. 1, pp. 119-125, 2003.

[22] J. Kleinberg, "Authoritative Sources in a Hyperlinked Environment," *J. ACM,* vol. 46, no. 5, pp. 604-632, 1999.

[23] T. Kolda and J. Sun, "Scalable Tensor Decompositions for Multi-Aspect Data Mining," *Proc. IEEE Int'l Conf. Data Mining (ICDM '08)*, 2008.
[24] L.D. Lathauwer, B.D. Moor, and J. Vandewalle, "A Multilinear Singular Value Decomposition," *SIAM J. Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253-1278, 2000.
[25] X. Li, L. Guo, and Y. Zhao, "Tag-Based Social Interest Discovery," *Proc. ACM World Wide Web (WWW) Conf.*, 2008.
[26] Y. Li, Y. Du, and X. Lin, "Kernel-Based Multifactor Analysis for Image Synthesis and Recognition," *Proc. IEEE Int'l Conf. Computer Vision*, 2005.
[27] M.W. Mahoney, M. Maggioni, and P. Drineas, "Tensor-Cur Decompositions for Tensor-Based Data," *Proc. ACM Conf. Knowledge Discovery and Data Mining (KDD '06)*, pp. 327-336, 2006.
[28] L. Page, S. Brin, R. Motwani, and T. Winograd, "The Pagerank Citation Ranking: Bringing Order to the Web," technical report, 1998.
[29] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-Based Collaborative Filtering Recommendation Algorithms," *Proc. World Wide Web (WWW) Conf.*, pp. 285-295, 2001.
[30] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Incremental Singular Value Decomposition Algorithms for Highly Scalable Recommender Systems," *Proc. Int'l Conf. Computer and Information Science*, 2002.
[31] K. Sripanidkulchai, B. Maggs, and H. Zhang, "Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems," *Proc. IEEE INFOCOM*, 2003.
[32] J. Sun, D. Shen, H. Zeng, Q. Yang, Y. Lu, and Z. Chen, "Cubesvd: A Novel Approach to Personalized Web Search," *Proc. World Wide Web Conf.*, pp. 382-390, 2005.
[33] J. Sun, D. Tao, and C. Faloutsos, "Beyond Streams and Graphs: Dynamic Tensor Analysis," *Proc. ACM Conf. Knowledge Discovery and Data Mining (KDD)*, pp. 374-383, 2006.
[34] P. Symeonidis, A. Nanopoulos, A. Papadopoulos, and Y. Manolopoulos, "Scalable Collaborative Filtering Based on Latent Semantic Indexing," *Proc. 21st Assoc. for Advancement of Artificial Intelligence (AAAI) Workshop Intelligent Techniques for Web Personalization (ITWP '06)*, pp. 1-9, 2006.
[35] M.E. Timmerman and H.A.L. Kiers, "Three Mode Principal Components Analysis: Choosing the Numbers of Components and Sensitivity to Local Optima," *J. Math. and Statistical Psychology*, vol. 53, no. 1, pp. 1-16, 2000.
[36] K. Tso-Sutter, B. Marinho, and L. Schmidt-Thieme, "Tag-Aware Recommender Systems by Fusion of Collaborative Filtering Algorithms," *Proc. ACM Symp. Applied Computing (SAC) Conf.*, 2008.
[37] H. Wang and N. Ahuja, "A Tensor Approximation Approach to Dimensionality Reduction," *Int'l J. Computer Vision*, vol. 76, no. 3, pp. 217-229, 2008.
[38] Y. Xu, L. Zhang, and W. Liu, "Cubic Analysis of Social Bookmarking for Personalized Recommendation," *Frontiers of WWW Research and Development—APWeb '06*, pp. 733-738, Springer, 2006.
[39] Z. Xu, Y. Fu, J. Mao, and D. Su, "Towards the Semantic Web: Collaborative Tag Suggestions," *Proc. Collaborative Web Tagging Workshop at World Wide Web (WWW '06)*, 2006.

Panagiotis Symeonidis received the bachelor's degree in applied informatics in 1996, and the MSc degree in information systems in 2004, from Macedonia University, Greece. He received the PhD degree in web mining from Aristotle University of Thessaloniki, Greece, in 2008. Currently, he is working as a postdoctoral researcher at Aristotle University of Thessaloniki, Greece. He is the coauthor of more than 20 articles in international journals and conference proceedings. His articles have received more than 40 citations from other scientific publications. His articles have received more than 600 citations from other scientific publications. He teaches courses on databases in the University of Western Macedonia and courses on data mining and data warehousing in a postgraduate program in Aristotle University of Thessaloniki. His main research interests include data mining and machine learning with applications in databases and information retrieval. His other research interests include web mining, information retrieval, recommender systems, and social tagging systems.

Alexandros Nanopoulos received the BSc and PhD degrees from the Department of Informatics of Aristotle University of Thessaloniki, Greece, where he taught as a lecturer from 2004 to 2008 courses on data mining and databases. From 2005 to 2008, he taught courses on databases in the Hellenic Open University. His main research interests include data mining and machine learning with applications in databases and information retrieval. He is the coauthor of more than 60 articles in international journals and conference proceedings. His articles have received more than 600 citations from other scientific publications. He has also coauthored the monographs *Advanced Signature Indexing for Multimedia and Web Applications* and *R-Trees: Theory and Applications*, both published by Springer Verlag. He has also coedited the volume *Wireless Information Highways*, published by Idea Group, Inc. In 2008, he has served as a cochair of the European Conference of Artificial Intelligence (ECAI) Workshop on Mining Social Data and, in 2006 and 2007, as a cochair of the Advances in Databases and Information Systems (ADBIS) Workshops on Data Mining and Knowledge Discovery. He has also served as a program committee member of several international conferences on data mining and databases.

Yannis Manolopoulos received the BEng degree (1981) in electrical engineering and the PhD degree (1986) in computer engineering, from Aristotle University of Thessaloniki. Currently, he is a professor in the Department of Informatics at Aristotle University of Thessaloniki. He has been with the Department of Computer Science at the University of Toronto, the Department of Computer Science at the University of Maryland at College Park, and the Department of Computer Science at the University of Cyprus. He has published about 200 papers in refereed scientific journals and conference proceedings. He is the coauthor of the following books: *Advanced Database Indexing* and *Advanced Signature Indexing for Multimedia and Web Applications* published by Kluwer, as well as *Nearest Neighbor Search: A Database Perspective* and *R-Trees: Theory and Applications* published by Springer. His published work has received more than 1,700 citations from more than 450 institutional groups. He served/serves as a general/PC chair/cochair of the Eighth National Computer Conference (2001), the Sixth ADBIS Conference (2002), the Fifth WDAS Workshop (2003), the Eighth SSTD Symposium (2003), the First Balkan Conference in Informatics (2003), the 16th SSDBM Conference (2004), the Eighth ICEIS Conference (2006), and the 10th ADBIS Conference (2006). His research interests include databases, data mining, web and geographical information systems, bibliometrics/webometrics, and performance evaluation of storage subsystems.

Queries to the Author

Q1. Please check and confirm that the affiliations are OK as typeset.
Q2. There are some references to color in Fig. 12. Kindly rephrase the specific mentions if you would like the figure to be published in black and white.
Q3. The acronyms in conference titles have been set in full in References [2], [7]-[9], [19], [20], [25], [29], [33], [34], [36], and [39]. Please check and confirm that they are correct.
Q4. References [17] and [38] have been set as book-type references. Please check and confirm that they are set correctly.
Q5. The name of the second author and the year information in Reference [30] have been included as per the published data available on the Internet. Please check and confirm that they are correct.
Q6. The bibliographic details in Reference [37] have been set as per the published data available on the Internet. Please check and confirm that they are correct.