Spectral Clustering in Social-Tagging Systems[⋆]

Alexandros Nanopoulos[1], Hans-Henning Gabriel[2], and Myra Spiliopoulou[2]

[1] Institute of Informatics, Hildesheim University, Germany
nanopoulos@ismll.de

[2] Faculty of Computer Science, Otto-von-Guericke-University Magdeburg, Germany
{hgabriel,myra}@iti.cs.uni-magdeburg.de

Abstract. Social tagging is an increasingly popular phenomenon with substantial impact on the way we perceive and understand the Web. For the many Web resources that are not self-descriptive, such as images, tagging is the sole way of associating them with concepts explicitly expressed in text. Consequently, users are encouraged to assign tags to Web resources, and tag recommenders are being developed to stimulate the re-use of existing tags in a consistent way. However, a tag still and inevitably expresses the personal perspective of each user upon the tagged resource. This personal perspective should be taken into account when assessing the similarity of resources with help of tags. In this paper, we focus on similarity-based clustering of tagged items, which can support several applications in social-tagging systems, like information retrieval, providing recommendations, or the establishment of user profiles and the discovery of topics. We show that it is necessary to capture and exploit the *multiple values of similarity* reflected in the tags assigned to the same item by different users. We model the items, the tags on them and the users who assigned the tags in a multigraph structure. To discover clusters of similar items, we extend spectral clustering, an approach successfully used for the clustering of complex data, into a method that captures multiple values of similarity between any two items. Our experiments with two real social-tagging data sets show that our new method is superior to conventional spectral clustering that ignores the existence of multiple values of similarity among the items.

1 Introduction

Social tagging is the process of saving bookmarks to a public Web site and tagging them with free-text keywords. With social tagging, a user expresses the own perspective on *items*, i.e. Web resources like images, videos, scientific papers, thus allowing other like-minded users to find the same information. The success of social tagging resulted to the proliferation of sites like Delicious, Citeulike, Digg, or Flickr. Such sites contain large amounts of tagged data that can be clustered on similarity and then used in information retrieval in social-tagging systems, for the formulation of recommendations in them [8,11], or for the establishment

[⋆] The first author gratefully acknowledges the partial co-funding of his work through the European Commission FP7 project MyMedia (www.mymediaproject.org) under the grant agreement no. 215006.

of user profiles and the discovery of topics, among other applications. However, similarity-based clustering of socially tagged items calls for methods that take account of the personalized perspectives posed by the users upon the items they tag. In this study, we deal with this challenge by proposing an innovative model for socially tagged items, in which we allow for multiple similarities per item. For this model, we use spectral clustering with tensor factorization. The clusters consist of items that are deemed similar by multiple users, even if each one considers them similar for different reasons!

The need for a new model emerges from the very nature of social tagging: data in a social-tagging system have three dimensions - the items, the users annotating them and the tags used for annotations. There is a 3-way relationship among these three dimensions; in classic database terminology, this means that the data cannot be brought into third normal form. Since conventional clustering algorithms model data in two-dimensional arrays (the rows are the items, the columns stand for the features), the 3-way relationship is solved by projecting away the third dimension: clustering is performed over items-users or items-tags arrays. Obviously, this incurs the loss of valuable information contained in the 3-way relationships.

Nevertheless, even when projecting one dimension away, the clustering of socially tagged items is challenging. The main reasons are the size of the feature space (large number of users or tags) and the complex cluster shapes. Spectral clustering algorithms lend themselves for such data. They capture similarity by spanning a graph structure, in which each item is connected to the k items most similar to it. By concentrating on the k nearest neighbors of each item, spectral clustering methods effectively project the data into a smaller, transformed feature space, in which they can detect complex clusters and suppress noise [13]. We exploit these properties by extending, however, spectral clustering to deal with all three dimensions of socially-tagged data without projecting any dimension.

In particular, we propose the computation of *multiple similarity values* for each pair of items to account for the fact that when all three dimensions are considered, the similarity between two items depends both on the users who tagged them and on the tags they used. To perform spectral clustering with multiple similarity values, we generalize the idea of a similarity graph into a similarity *multigraph* that has multiple edges between any two nodes. Similarity values on a graph would have been recorded on a conventional similarity matrix; for a multigraph, we must use a *tensor*, i.e. a multi-dimensional matrix. Spectral clustering on a matrix corresponds to matrix factorization with Singular Value Decomposition (SVD); accordingly, we perform tensor factorization.

The rest of the paper is organized as follows. Section 2 reviews related work. In Section 3 we give an overview of the proposed approach. Then, in Section 4 we describe our data model in detail and in Section 5 we present our clustering algorithm. In Section 6 we report on our comparison against a conventional spectral clustering method that only captures two-way relationships between items and users and between items and tags. We conclude in Section 7.

2 Related Work

The problem of clustering social data has recently started to attract attention. Giannakidou et al. [3] propose a co-clustering scheme that exploits joint groups of related tags and social data sources, in which both social and semantic aspects of tags are considered simultaneously. Their objective is to improve the retrieval of resources by exploiting their relation to tags. We also exploit the relation of items to tags, but not only: we capture also the relation of tags to users. Instead of co-clustering items with tags, we build solely clusters of items. For building these clusters, both the relation of items to tags and of tags to users are exploited to *induce* multiple similarities between items.

The need to cluster data with multiple similarity measures has been recognized only recently, as applications with very complex data structures started to proliferate. Seele et al have studied the problem of clustering bibliographic data with multiple similarity values [9]. They examined six different similarities on a collection of journal articles by considering, among others, similarities between words in abstracts, between names of authors co-citations etc. These similarities are predefined. In contrast, we *induce* multiple similarities from the tags, i.e. make no a priori assumptions as to their nature and number.

Shashua et al. [10] proposed the use of tensor factorization to cluster data that exhibit n-wise similarities, i.e. the definition of similarity involves more than two objects. Our problem specification is different: we consider *pairwise* similarity between objects, i.e. $n = 2$, but we take account of many pairwise similarities between any two objects.

Banerjee et al. [1] propose a method for multi-way clustering on tensors, thus extending co-clustering from matrices to tensors. However, their objective is to cluster different types of entities that are connected with relation graphs, rather than clustering items with multiple similarities.

3 Overview of Proposed Approach

Existing spectral clustering algorithms [13] first compute the k-NN similarity graph, which connects every item with its k-NN. Next, the Laplacian graph of the k-NN similarity graph is used instead, because of the benefits it offers, i.e., it is always positive-semidefinite (allowing its eigenvector decomposition) and the number of times 0 appears as its eigenvalue is the number of connected components in the k-NN similarity graph. Due to these convenient properties, if c clusters are required to be found, spectral clustering algorithms proceed by computing the c eigenvectors that correspond to the c smallest eigenvalues, and represent each original item with as a c-dimensional vector whose coordinates are the corresponding values within the c eigenvectors. With this representation, they can finally cluster the c-dimensional vectors using simple algorithms, like k-means or hierarchical agglomerative.

As described in Introduction, differently from conventional spectral clustering algorithms, our proposed approach considers multiple similarity values between each pair of items. In particular, let U be the set of all users. For a given tag t, let $U_1 \subseteq U$ be the set of users that tagged an item i_1 with t, whereas $U_2 \subseteq U$ be the set of users that tagged an item i_2 with t too. We can define a similarity value between i_1 and i_2 as follows. We form two vectors v_1 and v_2, both with $|U|$ elements that are set to 1 at positions that correspond to the users contained U_1 and U_2, respectively, whereas all rest positions are set to 0. Therefore, the similarity between i_1 and i_2 is given by the cosine measure between the two vectors v_1 and v_2. Since the above process can be repeated for all tags, the result is several similarity values between each pair of items i_1 and i_2. The set of all multiple similarity values are tag-aware and reflect the personalized aspect of similarity perceived by the users (e.g., two users may tag the same item but using entirely different tags).

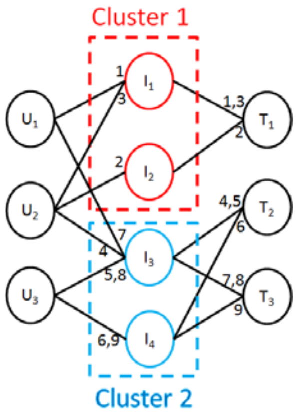
To account for the various similarity values between each pair of items, we extend (Section 4) the k-NN similarity graph to a k-NN multidigraph that is the union of multiple simple k-NN graphs, one for each distinct tag. The adjacency matrix of a k-NN multidigraph forms a tensor, i.e., a multidimensional array. In order to attain the aforementioned advantages of the Laplacian, we propose a method (Section 5.1) to extend towards the construction of the Laplacian multidigraph, whose adjacency matrix is again represented as a tensor. To map each item to a feature space comprised from spectral information extracted from the Laplacian tensor, we describe (Section 5.2) how to use tensor factorization that extends SVD to multidimensional arrays. Finally, based on the computed features, we describe (Section 5.3) how the clustering is performed. To help comprehension, throughout the rest of the article we use a running example with the following data.

Example 1 (Data representation). We assume 3 users, U_1, U_2, and U_3, who assign tags to 4 items (henceforth 'items' for simplicity), $I_1, \ldots, I_4$, from a tag-set with 3 tags, $T_1, \ldots, T_3$. Each assignment comprises a triple of the form (user, item, tag). The 9 triples of the example are given in Figure 1a, whereas we additionally denote (in the first column) the ID of the triple. The corresponding view of the data as tripartite graph is depicted in Figure 1b. In this figure, the numbered labels on the edges correspond to the triple IDs in Figure 1a. For instance, the first triple (ID = 1) is: U_1 tagged I_1 with T_1. In Figure 1b this corresponds to the path consisting of all edges labelled as 1. To avoid cluttering the figure, parallel edges (i.e., edges between the same two nodes) with different labels are depicted as one with different labels separated by comma. In this example, we assume that items I_1 and I_2 form one cluster, whereas items I_3 and I_4 form a second cluster. This follows by observing in Figure 1b that, although users tag items from both clusters, they assign different tags to the first cluster than the second. Therefore, the relationships between items-users alone are not able to determine a clustering structure among the items. In contrast, when considering the 3-way relationships between items-users-tags, we are able to better detect the clustering of items.[1] $\square$

[1] Although this example focuses on the comparison between items-users-tags and items-users relationships, we have to note that we have also verified experimentally that the former are preferable against items-tags relationships, as well.

ID	User	Item	Tag
1	U_1	I_1	T_1
2	U_2	I_2	T_1
3	U_2	I_1	T_1
4	U_2	I_3	T_2
5	U_3	I_3	T_2
6	U_3	I_4	T_2
7	U_1	I_3	T_3
8	U_3	I_3	T_3
9	U_3	I_4	T_3

(a)



(b)

Fig. 1. Running example: (a)Input data. (b) Illustration of the tripartite graph.

4 Modelling the Multiple Similarity Values

In this section, we describe the modelling of multiple similarity values with a k-nearest-neighbor multidigraph. A multidigraph is a directed graph permitted to have multiple directed edges (henceforth, simply called edges), i.e., edges with the same source and target nodes.

The input tripartite graph (like in the example of Figure 1b) can be partitioned according to the tags. For each tag t, we get the corresponding underlying subgraph B_t, by keeping users and items that participate in triples with this tag.

Example 2 (Partitioning of tripartite graph). For the example of Figure 1, the partitioning results to 3 (due to the existence of 3 tags) bipartite subgraphs, which are depicted in Figure 2: Figures 2a, b, and c correspond to the subgraphs $B_{T_1}, B_{T_2}, B_{T_3}$, for the tags T_1, T_2, and T_3, respectively. □

Each bipartite subgraph is represented with its adjacency matrix B_t ($1 \leq t \leq |T|$), whose size is $|I| \times |U|$; that is, its rows correspond to items and its columns to users. (Henceforth, wherever there is no ambiguity, we use interchangeably the same symbol for a graph and its adjacency matrix.) Each element $B_t(i, u)$ is equal to 1, if there is an edge between the item i and user u, or 0 otherwise. Therefore, from each adjacency matrix B_t we can compute between every pair of items i, j ($1 \leq i, j \leq |I|$), a similarity measure according to the values in their corresponding rows $B_t(i, :)$ and $B_t(j, :)$. Following the widely used approach for 2 dimensional matrices (like document-term in information retrieval or user-item in CF), we consider the cosine similarity measure between every pair of items.

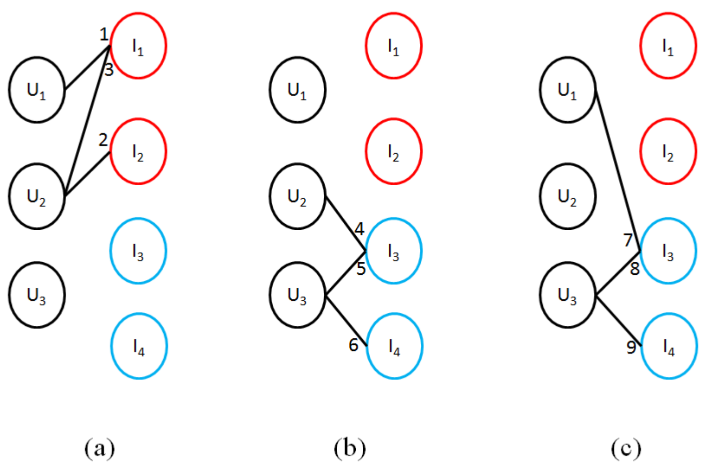


Fig. 2. Partitioning of the tripartite graph of the running example

Having defined a similarity measure, from each subgraph B_t ($1 \leq t \leq |T|$), we can compute the corresponding k-nearest neighbor (k-NN) graph, N_t, which is a labelled and directed graph (digraph). The node set of each N_t corresponds to the set of items (i.e., each item has a corresponding node). The edge set consists of ordered pair of nodes. There is an edge between items i and j ($1 \leq i, j \leq |I|$), if j is among the k nearest neighbors of i. Each edge is labelled with the corresponding similarity value.

By considering all k-NN digraphs together, we form the k-NN labelled multidigraph, $\mathcal{N}$. The node set of $\mathcal{N}$ corresponds to the set of items (i.e., each item has a corresponding node). The labelled edges of $\mathcal{N}$ is a multiset resulting from the union of the labelled edges of all N_t for $1 \leq t \leq |T|$. $\mathcal{N}$ summarizes the information about multiple similarities, according to the different tags between all items.

Example 3 (k-NN multidigraph). For the 3 subgraphs in Figure 2, the resulting k-NN multidigraph $\mathcal{N}$, for $k = 1$, is depicted in Figure 3a. The multiple edges between the nodes of $\mathcal{N}$ denote the different similarities between the items, according to the different tags. In Figure 3a, the edges representing similarities according to tag T_i ($1 \leq i \leq 3$) are annotated with T_i and then follows the corresponding similarity value.[2] Notice that $\mathcal{N}$ correctly captures the clustering structure: edges exist only between items of the same cluster, i.e., between I_1, I_2 for the first cluster and between I_3, I_4 for the second. Conversely, in Figure 3b, which depicts the k-NN digraph (not a multidigraph) when only user-item relationships are considered, the separation of clusters is not clear. □

[2] In this small example, to avoid numerical problems, we assign similarity equal to 0 when at least one item has no edge at all in the corresponding bipartite graphs. Moreover, to avoid cluttering the graph, only the non-zero similarities are depicted.

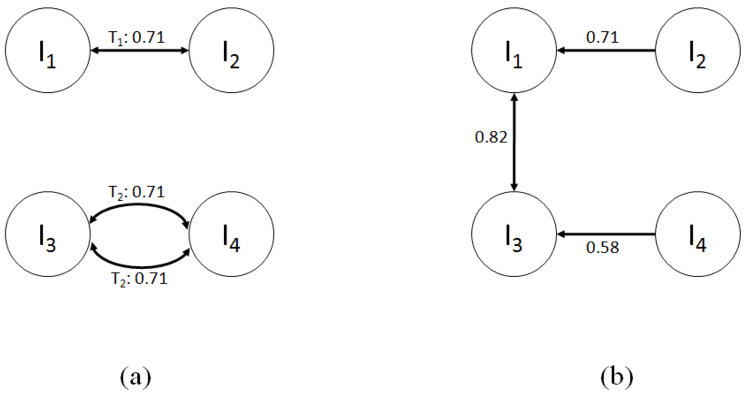


Fig. 3. The k-NN multidigraph for the running example

5 The Proposed Clustering Algorithm

5.1 Constructing the Laplacian Tensor

For each k-NN digraph N_t ($1 \leq t \leq |T|$) of $\mathcal{N}$, compute D_t as a diagonal matrix whose diagonal elements are defined as follows:

$$D_t(i,i) = \sum_{j=1}^{|I|} N_t(i,j) \tag{1}$$

The Laplacian matrix, L_t, of each N_t is computed as follows [7]:

$$L_t = \mathbb{1} - D_t^{-1/2} N_t D_t^{-1/2} \tag{2}$$

where $\mathbb{1}$ is the identity matrix.

The Laplacian tensor of $\mathcal{N}$ is, therefore defined as $\mathcal{L} \in \mathbb{R}^{|I|\times|I|\times|T|}$, whose elements are given as follows:

$$\mathcal{L}(i,j,t) = L_t(i,j) \tag{3}$$

Thus, each matrix L_t, for $1 \leq t \leq |T|$, comprises a frontal slice in $\mathcal{L}$.

Example 4 (Laplacian tensor). For the k-NN multidigraph of Figure 3, the resulting 3-mode Laplacian tensor is depicted in Figure 4, having as frontal slices the 3 L_t matrices ($1 \leq t \leq 3$). □

The Laplacian tensor $\mathcal{L}$ has 3 *modes* (illustrated with red arrows in Figure 4): the first mode corresponds to the items, the second mode to the neighboring items, and the third mode to the tags. To perform spectral clustering, we are interested in extracting the spectrum of $\mathcal{L}$ for the first mode. This procecure is explained in the following.

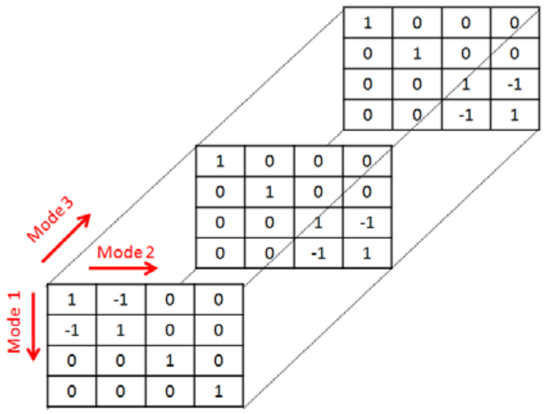


Fig. 4. The Laplacian tensor of the running example

5.2 Factorizing the Laplacian Tensor

In this subsection, we summarize the factorization of the Laplacian tensor using Tucker decomposition [5], which is the high-order analogue of the Singular Value Decomposition (SVD) for tensors. The factorization of the Laplacian tensor will produce the required spectrum of its first (corresponding to items) mode.

First, we have to define the n-mode product $\mathcal{T} \times_n M$ between a general N-order tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ and a matrix $M \in \mathbb{R}^{J_n \times I_n}$. The result is an $(I_1 \times I_2 \times \ldots \times I_{n-1} \times J_n \times I_{n+1} \times \ldots \times I_N)$-tensor, whose entries are defined as follows (elements are denoted through their subscript indexes):

$$(\mathcal{T} \times_n M)_{i_1 i_2 \ldots i_{n-1} j_n i_{n+1} \ldots i_N} = \sum_{i_n} T_{i_1 i_2 \ldots i_{n-1} i_n i_{n+1} \ldots i_N} M_{j_n i_n} \tag{4}$$

Since $\mathcal{L}$ is a 3-order tensor, we henceforth focus only on 1-mode, 2-mode, and 3-mode products.

The Tucker decomposition of the 3-order tensor $\mathcal{L}$ can be written as follows [6]:

$$\mathcal{L} \approx \mathcal{C} \times_1 P_1 \times_2 P_2 \times_3 P_3 \tag{5}$$

The $P_1 \in \mathbb{R}^{|I| \times |I|}, P_2 \in \mathbb{R}^{|I| \times |I|}, P_3 \in \mathbb{R}^{|T| \times |T|}$ are called the mode-1 (items), mode-2 (neighboring items), and mode-3 (tags) projection matrices, respectively. The 3 projection matrices contain the orthonormal vectors for each mode, called the mode-1, mode-2 and mode-3 singular vectors, respectively. $\mathcal{C}$ is called the core tensor and has the property of all orthogonality. Nevertheless, unlike SVD for matrices, $\mathcal{C}$ is not diagonal. Recently, several algorithms have been proposed to efficiently compute the components of the Tucker decomposition. Due to lack of space, more details about the algorithms and their complexity can be found in a recent survey on tensor factorization [5].

Having performed the Tucker decomposition of the Laplacian tensor $\mathcal{L}$, we are interested in the mode-1 singular vectors that are stored in P_1. A frequently followed approach in spectral clustering, when c clusters are required, is to select the c eigenvectors associated to the c smallest eigenvalues [13]. Similarly, we select the c mode-1 singular vectors in P_1 associated to the smallest singular values in the core tensor $\mathcal{C}$.

Example 5 (Selection of the mode-1 singular vectors). By performing the Tucker decomposition of the Laplacian tensor of the running example (Figure 4), the two selected mode-1 singular vectors from P_1 (recall that in the running example we have two clusters of items) are the following:

$$[0, 0, 0.71, 0.71]^T \text{ and } [0.71, 0.71, 0, 0]^T \qquad \square$$

5.3 Performing the Spectral Clustering

To find c clusters of items using the c mode-1 singular vectors that where computed and selected during the factorization of the Laplacian tensor, we apply the following steps: (1) Normalize the c selected mode-1 singular vectors to have norm equal to 1. (2) Form a matrix $X \in \mathbb{R}^{|I| \times k}$, whose columns are the normalized c selected mode-1 singular vectors. (3) Associate each item i to a point x_i whose coordinates are the contents of the i-th row of X. (4) Choose a distance metric for the $(x_i)_{i=1,\ldots,|I|}$ points. (5) Cluster the points $(x_i)_{i=1,\ldots,|I|}$ into c clusters using a clustering algorithm, according to the chosen distance metric. (6) Assign each item to the cluster of its associated point.

Due to the properties of the Laplacian tensor, in practice (and similarly to conventional spectral clustering on Laplacian graphs), the points in X can be easily clustered (Step 5) using simple and well known algorithms. In the sequel we consider hierarchical agglomerative algorithms for this purpose based on Euclidean distance (Step 4).

Example 6 (Clustering of items). After normalizing the vectors selected in Example 5, we get the X matrix depicted in Figure 5a. A simple hierarchical clustering algorithm, based on Euclidean distance, can easily detect two clusters, the first consisting of the first two points, whereas the second of the latter two points. This result is in accordance to the clusters assumed in the running example, i.e., the first one with the items I_1, I_2 and the second with the items I_3, I_4. To exemplify the effectiveness of the proposed representation, we can contrast the aforementioned result with the one obtained when performing spectral clustering without taking into account the information of tags. In this case, the corresponding X matrix is computed by taking the Laplacian matrix based only on items-users relationships, that is, originally we have a matrix where a user-item combination is set to 1 when the user tagged at least once the item. In this case, the resulting X matrix is depicted in Figure 5b. Evidently, in the latter case a clustering algorithm is not able to correctly detect the two clusters,

$$X = \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{pmatrix} \qquad\qquad X = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$$

(a) (b)

Fig. 5. The mapping when considering (a) items-users-tags (b) items-users

because items I_1 and I_4 will be incorrectly assigned to the same cluster (due to their identical coordinates), whereas either I_2 or I_3 will join the cluster of I_1, I_4, as their distance from each other is higher than their distance from I_1, I_4. □

Therefore, the proposed approach can better detect the clustering, because it fully exploits all items-users-tags relationships. This is verified with the experimental results in the following section.

6 Experimental Evaluation

6.1 Experimental Configuration

We experimentally tested the proposed method, denoted as Tensor-based Spectral Clustering (TSC). The baseline method is the Spectral Clustering (denoted as SC), which applies spectral clustering on the item-user 2 dimensional matrix with elements set to 1 when the corresponding item has been tagged at least once (no matter the tag) by the corresponding user.[3] Both TSC and SC have been implemented in Matlab using the same components. Tensor factorization was computed using the Tensor toolbox[4].

We consider two real social-tagging data sets. The first one is Movielens (downloaded from www.grouplens.org/node/73), which contains tags provided by users on movies. Associated information is available for movies. In our experiments we selected the genre (e.g., comedy, drama, etc.), where notice that each movie can belong to more than 1 out of 18 total genres. The second data set is Bibsonomy (provided by the authors of the paper [4]), which contains tags provided by users on Web resources (we excluded the tags of this data set that were given to scientific articles).

Social-tagging data present problems like tag polysemy and sparsity. To address them, we applied the widely used technique of Latent Semantic Indexing (LSI) [2] and reduced the number of dimensions in the modes of users and tags, by maintaining a percentage of them. This reduction was performed by modelling the original triples as a 3-mode tensor and applying Tucker decomposition [5].

[3] We have to note that we performed the same comparison against spectral clustering on a item-tag 2 dimensional matrix and found that it is outperformed by TSC as well. We omit the presentation of these results due to lack of space.

[4] http://csmr.ca.sandia.gov/∼tgkolda/TensorToolbox/

The item mode is left unchanged, whereas the number of maintained users and tags after this process is expressed as a percentage (default value 30%) of the original number of users and tags (for simplicity we use the same percentage for both). SC also utilize this technique by maintaining the same percentage for users or tags.

For the fifth step of the spectral clustering algorithm, we examined the Unweighted Pair Group Method with Arithmetic mean (UPGMA) hierarchical algorithm that defines the distance between two clusters as the average of the distances of all object pairs, selecting one object per cluster. We considered further hierarchical algorithms, but found that UPGMA performed best.

To measure the quality of the clustering results, we have used the measures of *entropy* (the lower the better) and *Jaccard coefficient* (the higher the better) to evaluate a clustering against the explicit class labels of Movielens. To measure the quality of one clustering (both Movielens and Bibsonomy) we used the *silhouette coefficient* (the higher the better). These measures are defined as follows [12].

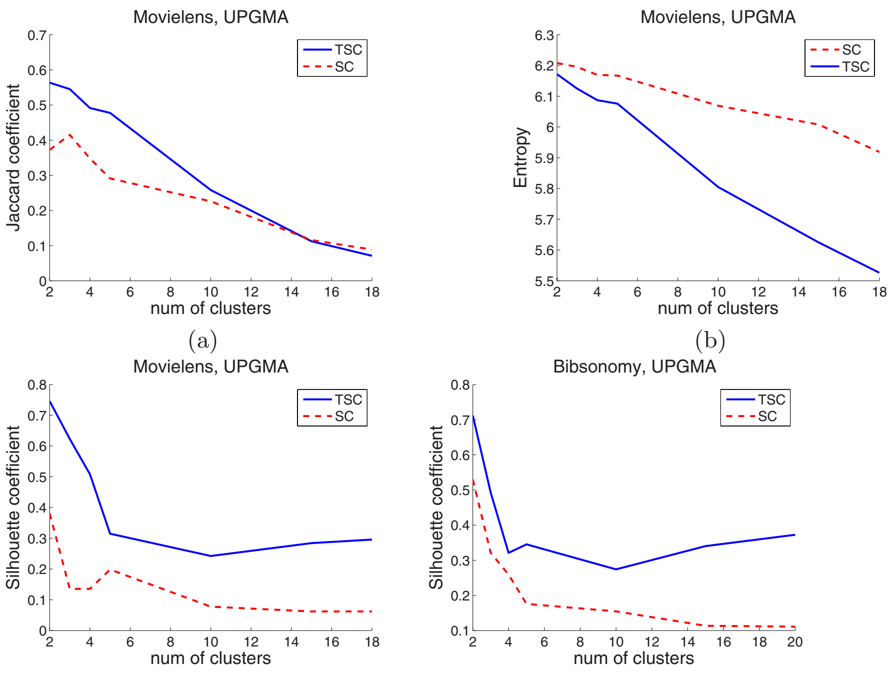
Let ζ be a clustering and ξ be the set of classes. $JaccardCoeff(\zeta, \xi) = \frac{f11}{f11+f01+f10}$, where $f11$ is the number of records of the same class that were put in the same cluster of ζ, $f10$ is the number of records that were put in the same cluster of ζ but belong to distinct classes, and $f01$ is the number of records that belong to the same class but appear in different clusters. The entropy measures the degree to which a cluster contains tuples belonging to a single class: $entropy(\zeta, \xi) = \frac{\sum_{C_u \in \zeta} |C_u| e(C_u)}{|\cup_{C_v \in \zeta} C_v|}$, where the probability that a tuple in C belongs to L_v is $p_{uv} = \frac{|C_u \cap L_v|}{|C_u|}$ and The entropy of C_u is $e(C_u, \xi) = \sum_{L_v \in \xi} p_{uv} log_2 p_{uv}$.

To compute the silhouette coefficient of a tuple x in cluster $C \in \zeta$, we calculate its average distance a_x from all other tuples in C and from the t uples in the clusters of $\zeta \setminus \{C\}$, say b_x. Then $s(x) = \frac{(b_x - a_x)}{\max(a_x, b_x)}$. The silhouette of C is the average silhouette of its members. The silhouette for the clustering $silhouette(\zeta)$ is the average over the cluster silhouettes, weighted with cluster cardinalities.

6.2 Experimental Results

We experimentally compare TSC and SC and examine sensitivity against the following parameters: the number of neighbors, k (default value 10) and the percentage of maintained users/items (default value 10%). We examine a varying number of clusters c that, following the approach of conventional spectral clustering algorithms [13], we considered it as a user-defined parameter. Therefore, the number of clusters vary up to 18 for Movielens (which is the number of distinct genres) and up to 20 for Bibsonomy.

Figures 6a–c present the results for the Movielens data set against varying number of clusters. The measures are Jaccard coefficient, Entropy, and Silhouette coefficient. In all cases, TSC performs favorably against SC. Analogous conclusion is drawn for the Bibsonomy data set, the results for which (only Silhouette coefficient is applicable) are presented in Figures 6d.



(a) (b)

(c) (d)

Fig. 6. Experimental results for Movielens and Bibsonomy data set

We tested the sensitivity against the number of nearest neighbors, k, that is used during the creation of the k-NN multidigraph. Figures 7a and b present the results for Movielens and Bibsonomy, respectively. In both cases we require the maximum number of clusters that were examined in the previous experiments (for brevity, for Movielens we present only Silhouette coefficient). When k is very low, the performance of TSC can be negatively affected, because not enough similarity information is captured. For k values in the range between 10 and 20, best performance is attained. Then, as k increases, performance deteriorates, because noise incurs (TSC considers items that are not truly neighbors). Analogous results hold for SC, although deterioration is less pronounced. Nevertheless, in all cases TSC is superior to SC.

Next, we measured the impact of the percentage of maintained users/tags. Figures 8 a and b present the results for Movielens and Bibsonomy, respectively. Again, we required the maximum number of clusters, whereas k is set to 10. As expected, for both TSC and SC, as the percentage of maintained users/tags increases, performance is reduced due to the problems described in Section 6.1. However, TSC is able to attain much better performance than SC even for higher percentage of maintained users/tags, which means that TSC can better cope with these problems.

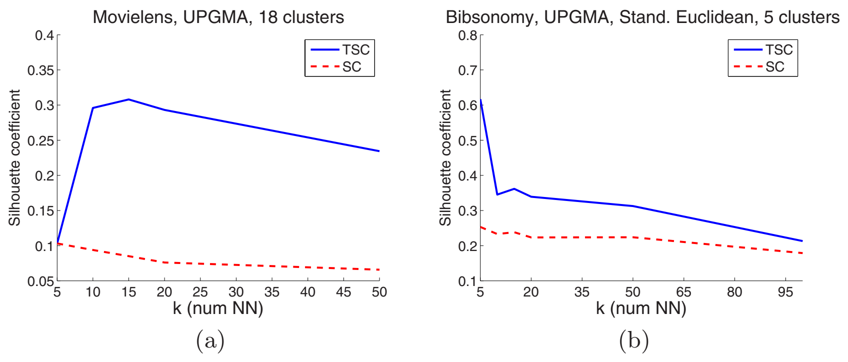


Fig. 7. Experimental results on sensitivity to the number of nearest neighbors, k

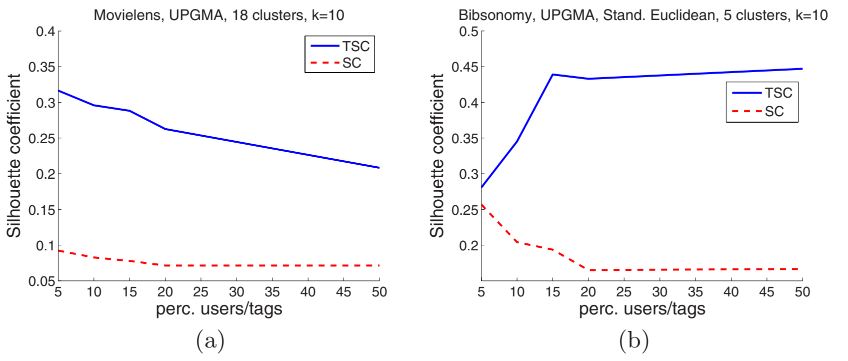


Fig. 8. Experimental results on sensitivity to the percentage of maintained users/tags

7 Conclusions

We have considered the data mining task of clustering items (e.g., Web resources, images, scientific papers, etc.) stored in social-tagging systems. To overcome the problems of existing approaches and avoid breaking the original 3-way relationships that are present in social-tagged data, we proposed the extension of the popular spectral clustering algorithms to directly handle all dimensions without suppressing them with 2-way relationships. The proposed approach consists of the following contributions: (i) We provided the insight that it is necessary to capture and exploit the *multiple* similarity values reflected in the tags assigned to the same item by different users. (ii) To support multiple similarity values, we extended the modeling based on k-NN similarity graphs by using k-NN similarity multigraphs, which allow the existence of multiple edges between two nodes. (iii) We modeled the multigraph structures as *tensors* (i.e., multidimensional arrays) and extended the popular spectral-clustering algorithms by developing a method to construct their corresponding Laplacian tensors. (iv) We described the use of tensor factorization, which extends the Singular Value Decomposition (SVD) to multi-dimensional arrays, to extract spectral features from the Laplacian tensors. (v) Our experimental results with real data indicate the clear advantage, in terms of quality of the final clustering, of the proposed method against conventional spectral clustering that suppresses the original data by considering only 2-way relationships.

As future work, we will extend the proposed clustering method in other kind of data, like documents. Also, we plan to develop model-based CF algorithms that will exploit the proposed clustering method.

References

1. Banerjee, A., Basu, S., Merugu, S.: Multi-way clustering on relation graphs. In: Proceedings of the 7th SIAM International Conference on Data Mining, SDM 2007 (2007)
2. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society for Information Science 41, 391–407 (1990)
3. Giannakidou, E., Koutsonikola, V., Vakali, A., Kompatsiaris, Y.: Co-clustering tags and social data sources. In: Proceedings of the 9th International Conference on Web-Age Information Management (WAIM 2008), pp. 317–324 (2008)
4. Jäschke, R., Marinho, L.B., Hotho, A., Schmidt-Thieme, L., Stumme, G.: Tag recommendations in folksonomies. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 506–514. Springer, Heidelberg (2007)
5. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Review 51(3) (to appear, 2009)
6. de Lathauwer, L., de Moor, B., Vandewalle, J.: A multilinear singular value decomposition. SIAM Journal of Matrix Analysis and Applications 21(4), 1253–1278 (2000)
7. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: Proceedings of the Advances in Neural Information Processing Systems (NIPS 2001), pp. 849–856 (2001)
8. Rendle, S., Marinho, L., Nanopoulos, A., Schmidt-Thieme, L.: Learning optimal ranking with tensor factorization for tag recommendation. In: Proceedings of the ACM Conf. on Knowledge Discovery and Data Mining, KDD 2009 (to appear, 2009)
9. Selee, T.M., Kolda, T.G., Kegelmeyer, W.P., Griffin, J.D.: Extracting clusters from large datasets with multiple similarity measures using IMSCAND. In: Parks, M.L., Collis, S.S. (eds.) CSRI Summer Proceedings 2007, Technical Report SAND2007-7977, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, pp. 87–103 (2007)
10. Shashua, A., Zass, R., Hazan, T.: Multi-way clustering using super-symmetric nonnegative tensor factorization. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3954, pp. 595–608. Springer, Heidelberg (2006)
11. Symeonidis, P., Nanopoulos, A., Manolopoulos, Y.: A unified framework for providing recommendations in social tagging systems based on ternary semantic analysis. IEEE Transactions on Knowledge and Data Engineering (accepted, 2009)
12. Tan, P.-N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Wiley, Chichester (2004)
13. von Luxburg, U.: A tutorial on spectral clustering. Technical report (No. TR-149) Max Planck Institute for Biological Cybernetics (2006)