Enhancing Link-Based Similarity Through the Use of Non-Numerical Labels and Prior Information

Christian Desrosiers
Software Eng. and IT Department
Ecole de Technologie Superieure
1100 Notre-Dame W.
Montreal (QC) H3C1K3, Canada
christian.desrosiers@etsmtl.ca

George Karypis
Computer Science & Eng. Department
University of Minnesota, Twin Cities
200 Union Street SE
Minneapolis (MN) 55455, USA
karypis@cs.umn.edu

ABSTRACT

Several key applications like recommender systems require to compute similarities between the nodes (objects or entities) of a bipartite network. These similarities serve many important purposes, such as finding users sharing common interests or items with similar characteristics, as well as the automated recommendation and categorization of items. While a broad range of methods have been proposed to compute similarities in networks, such methods have two limitations: (1) they require the link values to be in the form of *numerical weights* representing the strength of the corresponding relation, and (2) they do not take into account prior information on the similarities. This paper presents a novel approach, based on the *SimRank* algorithm, to compute similarities between the nodes of a bipartite network. Unlike current methods, this approach allows one to model the agreement between link values using any desired function, and provides a simple way to integrate prior information on the similarity values directly in the computations. To evaluate its usefulness, we test this approach on the problem of predicting the ratings of users for movies and jokes.

Keywords

Networks, link-based similarity, SimRank, item recommendation

1. INTRODUCTION

Computing significant similarities between objects or entities of a network is an important problem that has several key applications such as image processing, classifying web pages, classifying protein interaction and gene expression data, part-of-speech tagging, detecting malicious or fraudulent activities, and recommending new and interesting items to consumers. For instance, in recommender systems, one of the most crucial tasks is to find users that share common interests, or items with similar characteristics. This task has several valuable uses, among which are the recommendation of new items, the discovery of groups of like-minded individuals, and the automated categorization of items.

A popular method to compute the similarity between users or items, found in many recommender systems, is based on the correlation between the ratings made by users on common items. As recognized by several recent works on this topic, such as [7, 12, 28], this method is very sensitive to sparse data. For instance, while two users can be similar if they have rated different items, this method is unable to evaluate their similarity in such cases. Moreover, although efficient approaches based on dimensionality reduction and graph theory have been proposed for this problem, they generally suffer from two limitations:

- They do not allow categorical ratings or other non-numerical rating types;
- They do not provide an easy way to integrate prior information on the similarity values.

To illustrate these limitations, consider the example of Figure 1. This example shows the categorical evaluation (good, bad, ok and N/A) of two unknown criteria, given by three users (John, Mary and Simon) to items i and j. Since the ratings are non-numerical, traditional methods based on correlation, dimensionality reduction or graph theory cannot be used to compute the similarity between these users or items. Furthermore, prior information on the similarity between users or items, obtained from user profiles or item contents, is often available in recommender systems. Although not considered by current methods, this information can be used to enhance the computation of similarities.

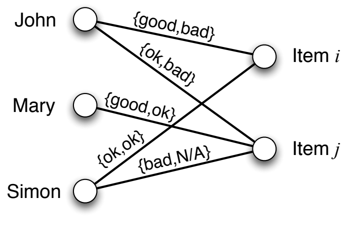

Figure 1: A bipartite graph representing responses (sets of categorical values) given by users to items.

This paper presents a general approach to compute similarities between the nodes of a bipartite network. Based on the well-known *SimRank* algorithm [14], this approach models the relations between the similarities of the two node groups (e.g., users and items) as a system of equations, and computes the similarity values by solving this system. However, unlike *SimRank* and its recent extensions, our approach has the additional advantage of allowing one to evaluate the *agreement* between any type of link values (edge labels), and integrate prior similarity information.

The rest of this paper is organized as follows. In Section 2, we present some of the most relevant work on the topic and describe the advantages of our approach over these works. We then present the details of our approach in Section 3, and illustrate in Section 4 its usefulness on the problem of predicting the ratings of users for movies and jokes. Finally, Section 5 provides a brief summary of our work and contributions, and describes some of its possible extensions.

2. RELATED WORK

2.1 Similarity in networks

A wide range of methods have been proposed to compute similarities between nodes in a network. A popular approach is to measure the similarity between two nodes as a function of their proximity and connectivity in the network. Methods using this approach include those based on geodesic distance [1, 22], diffusion and randomized kernels [15, 26], associative retrieval [12], random forests [13], and random walks [7, 9, 28]. A common problem with these methods is their lack of interpretability. For instance, in the case of item recommendation, it is not clear how the proximity between two users or two items in the network can be translated into a similarity value. Moreover, these methods are limited to numerical link values representing weights. Thus, it would not be possible to use such methods in the item recommendation context where users give non-numerical responses to items, for instance, categorical responses or a set of numerical responses (see the example of Figure 1). Another recently proposed method for this task, which proved useful in the problem of classifying nodes of a network, evaluates the similarity between two nodes in the network by comparing the structural information of their neighborhood [5]. While this approach considers non-numerical link values, it is not so applicable to the context of item recommendation, as users with different rating patterns may still be similar.

2.2 SimRank

A different approach, closely related to the one presented in this paper is the bipartite version of the *SimRank* algorithm, proposed by Jeh and Widom [14]. Let $\mathcal{U}$ and $\mathcal{I}$ be the two sets of nodes of a bipartite graph representing, for instance, the users and items of a recommender system. Moreover, denote by $\mathcal{I}_u \subseteq \mathcal{I}$ be the set of nodes connected to a node $u \in \mathcal{U}$ (e.g., the items purchased by user u), and let $\mathcal{U}_i \subseteq \mathcal{U}$ be the set of nodes connected to a node $i \in \mathcal{I}$ (e.g., the users that have purchased item i). The similarity between two nodes u and v of $\mathcal{U}$, $s(u,v)$, is obtained as the average similarity of their neighbors:

$$s(u,v) \;=\; \frac{C_1}{|\mathcal{I}_u||\mathcal{I}_v|} \sum_{i \in \mathcal{I}_u} \sum_{j \in \mathcal{I}_v} s(i,j), \qquad (1)$$

where $C_1 \in [0,1]$ is a constant controlling the flow of similarity values on the links. Likewise, the similarity between two nodes i and j of $\mathcal{I}$, $s(i,j)$, can be computed as the average similarity of their neighbors in $\mathcal{U}$:

$$s(i,j) \;=\; \frac{C_2}{|\mathcal{U}_i||\mathcal{U}_j|} \sum_{u \in \mathcal{U}_i} \sum_{v \in \mathcal{U}_j} s(u,v), \qquad (2)$$

C_2 having the same role as C_1. *SimRank* computes the similarity values by updating them iteratively using equations (1) and (2), until a fixed-point is reached. Several techniques to accelerate the computation of the similarity values have been proposed, for instance, [19, 21]. Moreover, an extension of *SimRank*, which computes the similarities over a *maximum matching* of neighbor nodes, was recently proposed in [20].

A significant limitation of *SimRank* is that it does not consider the nature nor the strength of the links. For instance, in the context of item recommendation, this method considers the interactions between users and items (e.g., purchases) but not the corresponding ratings. Another method called *SimRank++*, recently proposed in [2], extends *SimRank* by taking into account the link weights as modified transition probabilities. In this method, the similarity between two nodes is computed as a weighted average of the similarities of their adjacent nodes:

$$s(u,v) \;=\; C_1 \sum_{i \in \mathcal{I}_u} \sum_{j \in \mathcal{I}_v} w_{ui}\, w_{vj}\, s(i,j), \qquad (3)$$

where w_{ui} is the normalized weight of the link between u and i. Like *SimRank*, this method also has some limitations. First, non-numerical link values may not be used, since these values are simply simply multiplied in equation (3). Furthermore, this method does not allow one to integrate prior knowledge on the similarity values, for instance, obtained by comparing the content of items in a recommender system.

2.3 Recommendation methods

Since the approach presented in this paper is evaluated in the context of item recommendation, it is necessary to give a brief overview of the most significant works on this topic. Essentially, approaches proposed for the task of recommending items can be divided into two categories: *neighborhood-* and *model-* based methods. Neighborhood-based approaches recommend new items to a user either based on the ratings of alike users, i.e. *user-based* recommendation [16], or the ratings of this user on similar items, i.e. *item-based* recommendation [4]. While neighborhood methods are rather simple and have the advantage of exploiting local information, they tend to suffer from sparse rating data [7, 28].

In contrast, model-based approaches use the rating data to learn a predictive model using latent characteristics of the users and items in the system, like the preference class of users and the category class of items. This model is then trained using the available data, and later used to predict ratings of users for new items. Model-based recommendation methods are numerous and include Latent Semantic Analysis (LSA) [11], Latent Dirichlet Allocation (LDA) [3], Boltzmann machines [23], Support Vector Machines (SVM) [10], and dimensionality reduction [17, 25, 27]. Because ratings are predicted based on higher-level factors, model approaches are usually less sensitive to data sparsity than neighborhood ones [17].

Finally, a few studies, such as [6, 18], have focused on evaluating the usefulness of link-based similarity in the task of predicting the ratings of users for new items. While these studies have found such methods to be somewhat inferior to approaches specifically designed for this task, the experimental results described in this paper suggest that link-based similarity methods could be useful in the context where the rating data is sparse.

2.4 Contributions

This paper makes three contributions:

1. It presents a novel approach to compute similarities, sharing common characteristics with the *SimRank* algorithm. Unlike *SimRank* and its extensions, this approach also:
   - Allows one to use an arbitrary function to evaluate the agreement between links, which makes possible the use of non-numerical values.
   - Provides an elegant way to integrate prior information on the similarity values directly in the computations.
2. In the context of item recommendation, unlike similarity measures based on correlation which only use the ratings on common items, this approach considers all the available ratings. This make possible the computation of similarities between users that have rated different items, thereby reducing the sensitivity to sparse data.
3. Finally, this paper presents a first comprehensive experimental evaluation of a *SimRank*-based method on the problem of predicting new ratings.

3. A NOVEL APPROACH

3.1 Modeling similarities as a linear system

Following the notation introduced in the previous section, let $\mathcal{U}$ and $\mathcal{I}$ represent the two disjoint sets of nodes of a bipartite graph, $\mathcal{I}_u$ the nodes of $\mathcal{I}$ linked to a node $u \in \mathcal{U}$, and $\mathcal{U}_i$ the nodes of $\mathcal{U}$ linked to a node $i \in \mathcal{I}$. Moreover, if nodes u and i are linked, denote as r_{ui} the *numerical* or *categorical* value of this link.

Consider the task of evaluating the similarity $s(u, v)$ between two nodes $u, v \in \mathcal{U}$. A simple approach, used in several item recommendation systems, is to evaluate $s(u, v)$ as the correlation evaluated over the values of links to common nodes, i.e. $\mathcal{I}_u \cap \mathcal{I}_v$. On top of being limited to numerical values, this approach has another significant problem: similarities can only be evaluated for nodes having common neighbors, e.g. users that have evaluated common items, and the correlation values are only significant if there is a sufficient number of common nodes. For these reasons, the correlation approach gives poor results when the bipartite graph is sparse.

As in *SimRank*, our approach overcomes these limitations by using the information of links to all neighbors of u and v, not only their common ones. Thus, we evaluate the similarity between nodes u and v as the average link agreement for all pairs of neighbors, weighted by the similarity of these neighbors:

$$s(u,v) \;=\; \frac{1}{Z_{uv}} \sum_{i \in \mathcal{I}_u} \sum_{j \in \mathcal{I}_v} s(i,j)\; k(r_{ui}, r_{vj}), \qquad (4)$$

where k is a function that evaluates the agreement between two (possibly non-numerical) link values, and Z_{uv} is a normalization constant, for instance, $Z_{uv} = |\mathcal{I}_u||\mathcal{I}_v|$. Examples of agreement function k for *numerical* link values are the *Radial Basis Function* (RBF) Gaussian kernel

$$k_{\text{RBF}}(r_{ui}, r_{vj}) \;=\; \exp\{-(r_{ui} - r_{vj})^2/\gamma^2\}, \qquad (5)$$

where γ controls the width of the kernel, and the *Correlation* kernel

$$k_{\text{Cor}}(r_{ui}, r_{vj}) \;=\; \frac{(r_{ui} - \overline{r}_u)(r_{vj} - \overline{r}_v)}{\sigma_u\, \sigma_v}, \qquad (6)$$

$\overline{r}_u$ and σ_u being the mean and standard deviation of values for links connecting u to its neighbor nodes. Note that k does not need to be semi-definite positive (SDP), and the term *kernel* is used in a more general way to represent a function measuring similarity.

A benefit of this formulation is that the agreement between the link values is abstracted in function k, which can be tailored to model specific characteristics or constraints of the system, as well as to measure the agreement between any value types. Moreover, this formulation can be easily extended to include prior information on the similarity between nodes u and v, obtained, for example, by comparing user profiles or item content. Denote $\hat{s}(u, v)$ the *a priori* similarity capturing this information, (4) can be extended to include $\hat{s}(u, v)$ as

$$s(u,v) \;=\; (1-\alpha)\, \hat{s}(u,v) \;+\; \frac{\alpha}{Z_{uv}} \sum_{i \in \mathcal{I}_u} \sum_{j \in \mathcal{I}_v} s(i,j)\; k(r_{ui}, r_{vj}), \qquad (7)$$

where $\alpha \in [0, 1]$ controls the importance of the *a priori* similarity in the computation. Likewise, the similarity $s(i, j)$ between two nodes $i, j \in \mathcal{I}$ can be modeled as

$$s(i,j) \;=\; (1-\alpha)\, \hat{s}(i,j) \;+\; \frac{\alpha}{Z_{ij}} \sum_{u \in \mathcal{U}_i} \sum_{v \in \mathcal{U}_j} s(u,v)\; k(r_{ui}, r_{vj}), \qquad (8)$$

where $\hat{s}(i, j)$ models prior knowledge on the similarity between i and j, and Z_{ij} has the same role as Z_{uv}.

3.2 Solving the general system

The relations between similarity values, as defined by equations (7) and (8), form a linear system which can be described using a matricial notation. Let $\vec{x} \in \mathbb{R}^{|\mathcal{U}|^2}$ and $\vec{y} \in \mathbb{R}^{|\mathcal{I}|^2}$ be vectors such that, for each pair of nodes $u, v \in \mathcal{U}$, $\vec{x}_{(uv)} = s(u, v)$ and, for each pair of nodes $i, j \in \mathcal{I}$, $\vec{y}_{(ij)} = s(i, j)$. Furthermore, let $\vec{c} \in \mathbb{R}^{|\mathcal{U}|^2}$ and $\vec{d} \in \mathbb{R}^{|\mathcal{I}|^2}$ be vectors such that $\vec{c}_{(uv)} = \hat{s}(u, v)$ and $\vec{d}_{(ij)} = \hat{s}(i, j)$. The linear system can be written in matrix form as

$$\left(\begin{array}{c} \vec{x} \\ \hline \vec{y} \end{array}\right) \;=\; (1-\alpha) \left(\begin{array}{c} \vec{c} \\ \hline \vec{d} \end{array}\right) \;+\; \alpha \left(\begin{array}{c|c} 0 & A \\ \hline B & 0 \end{array}\right) \left(\begin{array}{c} \vec{x} \\ \hline \vec{y} \end{array}\right). \qquad (9)$$

where A is a $(|\mathcal{U}|^2 \times |\mathcal{I}|^2)$ matrix such that

$$A_{(uv)(ij)} = \begin{cases} \frac{1}{Z_{uv}} k(r_{ui}, r_{vj}), & \text{if } i \in \mathcal{I}_u \text{ and } j \in \mathcal{I}_v, \\ 0, & \text{otherwise,} \end{cases}$$

and B is a $(|\mathcal{I}|^2 \times |\mathcal{U}|^2)$ matrix such that

$$B_{(ij)(uv)} = \begin{cases} \frac{1}{Z_{ij}} k(r_{ui}, r_{vj}), & \text{if } u \in \mathcal{U}_i \text{ and } v \in \mathcal{U}_j, \\ 0, & \text{otherwise.} \end{cases}$$

The solution of this system is then given by

$$\left(\begin{array}{c}\vec{x}\\ \hline \vec{y}\end{array}\right) = (1-\alpha)\left(\begin{array}{c|c} I & -\alpha A \\ \hline -\alpha B & I\end{array}\right)^{-1}\left(\begin{array}{c}\vec{c}\\ \hline \vec{d}\end{array}\right)$$
$$= (1-\alpha)\left(\begin{array}{c|c} R^{-1} & \alpha A S^{-1} \\ \hline \alpha B R^{-1} & S^{-1}\end{array}\right)\left(\begin{array}{c}\vec{c}\\ \hline \vec{d}\end{array}\right), \quad (10)$$

where $R = (I - \alpha^2 AB)$ and $S = (I - \alpha^2 BA)$.

Note that R^{-1} and S^{-1} exist and can be computed using a *von Neumann series* expansion [15, 18]:

$$R^{-1} = \sum_{n=0}^{\infty} \alpha^{2n}(AB)^n \quad (11)$$

$$S^{-1} = \sum_{n=0}^{\infty} \alpha^{2n}(BA)^n. \quad (12)$$

The solution for $\vec{x}$ can therefore be expressed as

$$\vec{x} = (1-\alpha)\left(\sum_{n=0}^{\infty}\alpha^{2n}(AB)^n\vec{c} \;+\; \alpha A\sum_{n=0}^{\infty}\alpha^{2n}(BA)^n\vec{d}\right)$$
$$= \left(\sum_{n=0}^{\infty}\alpha^{2n}(AB)^n\right)\vec{p}$$
$$= R^{-1}\vec{p} \quad (13)$$

where $\vec{p} = (1-\alpha)\left(\vec{c} \,+\, \alpha A\vec{d}\right)$. Using the same approach, $\vec{y}$ is obtained as

$$\vec{y} = \left(\sum_{n=0}^{\infty}\alpha^{2n}(BA)^n\right)\vec{q}$$
$$= S^{-1}\vec{q}, \quad (14)$$

where $\vec{q} = (1-\alpha)\left(\alpha B\vec{c} \,+\, \vec{d}\right)$.

3.3 Computing the similarities

Although A and B may be very sparse matrices, their large size can render difficult the direct computation of R^{-1} and S^{-1}. A more efficient approach consists in using an iterative method based on the *von Neumann series* expansion of the matrices. While we only describe how this method can be used to compute $\vec{x}$, the same principles can also be applied to find $\vec{y}$.

This method first initializes $\vec{x}$ to the *null* vector and initializes a temporary vector $\vec{w}$ to $\vec{p}$. The following two steps are then repeated until convergence or a maximum number of iterations is reached:

1. Update the similarities vector:
$$\vec{x} \;\leftarrow\; \vec{x} + \vec{w},$$
2. Update the temporary vector:
$$\vec{w} \;\leftarrow\; \alpha^2 AB\vec{w}.$$

THEOREM 1. *Denote by $\lambda_{\max}$ the largest eigenvalue of matrix AB, also known as its spectral radius. The iterative method presented above converges if $\alpha^2|\lambda_{\max}| < 1$.*

PROOF. Let $X\Lambda X^{-1}$ be the eigen-decomposition of matrix AB. At the n-th iteration, we have

$$||\alpha^{2n}(AB)^n|| = ||X(\alpha^2\Lambda)^n X^{-1}||$$
$$\leq ||X|| \cdot \sqrt{\sum_i(\alpha^2\lambda_i)^{2n}} \cdot ||X^{-1}||.$$

If $\alpha^2|\lambda_{\max}| < 1$ then $||\alpha^{2n}(AB)^n||$ will converge to 0 as n approaches infinity. As a consequence, $\vec{x}$ will converge to a fixed value. □

To analyze the complexity of this approach, we suppose the maximum number of neighbors of a node $u \in \mathcal{U}$ to be bounded by a constant m. This is consistent with many applications like item recommendation systems, where users typically purchase/rate a limited number of items, independent of the total number of available items. Since $A_{(uv)(ij)}$ is non-zero only if $i \in \mathcal{I}_u$ and $j \in \mathcal{I}_v$, assuming an even distribution of links among the nodes, the expected number of non-zero values in A is given by

$$\frac{|\mathcal{U}|^2\,|\mathcal{I}|^2}{2} \times \left(\frac{m}{|\mathcal{I}|}\right)^2 \;=\; \frac{|\mathcal{U}|^2\,m^2}{2} \;\in\; O(|\mathcal{U}|^2).$$

Likewise, we find the expected number of non-zero elements of B to be in $O(|\mathcal{U}|^2)$. Moreover, because the method has to store the non-zero values of A and B, as well as the values of possibly dense vectors $\vec{x}$ and $\vec{p}$, the expected space complexity of the method is $O(|\mathcal{U}|^2)$. For the time complexity, the dominant operations are the two matrix multiplications: $B\vec{w} = \vec{w}'$ and $A\vec{w}'$. Since the complexity of these operations is proportional to the number of non-zero elements in the multiplying matrices, the total expected time complexity of the method is $O(n_{\max}|\mathcal{U}|^2)$, where $n_{\max}$ is the maximum number of iterations made by the method. While $n_{\max}$ largely depends on the normalization constants Z_{uv} and Z_{ij}, as well as on the link agreement function k, in our experiments, the method would normally take 5 to 10 iterations to converge.

3.4 Solving without prior information

Although it is always possible to use default values for $\vec{c}$ and $\vec{d}$, for instance $\vec{c}_{(uv)} = 1$ if $u = v$ and 0 otherwise, the approach proposed in this paper could also be used without such information. The following theorem explains how this can be done.

THEOREM 2. *Let G be a directed weighted bipartite graph constructed such that each pair of users u, v corresponds to a node (uv) from the first set of nodes, each pair of items i, j is a node (ij) from the second set, and whose adjacency matrix is*

$$adj(G) \;=\; \left(\begin{array}{c|c} 0 & A \\ \hline B & 0\end{array}\right).$$

If $\alpha = 1$, A,B are non-negative matrices and G is connected, then vectors $\vec{x}$ and $\vec{y}$ correspond, respectively, to the unique eigenvectors of matrices AB and BA associated with the largest eigenvalue of these matrices. Moreover, these eigenvectors can be computed using a power iteration *method [8].*

PROOF. Suppose we constrain $\vec{x}$ and $\vec{y}$ to a specific length, for instance $||\vec{x}|| = ||\vec{y}|| = 1$, then equations (7) and (8) can be expressed as $\vec{x} = \frac{1}{\omega}A\vec{y}$ and $\vec{y} = \frac{1}{\sigma}B\vec{x}$, where ω and σ are normalization constants. Inserting the second one into the first, we get $(\sigma\omega)\vec{x} = AB\vec{x}$ and, thus, $\vec{x}$ is an eigenvector of AB corresponding to the eigenvalue $\lambda = \sigma\omega$. Likewise, $\vec{y}$ is an eigenvector of BA corresponding to the *same* eigenvalue.

Furthermore, since A and B are non-negative, so are matrices AB and BA. Also, because G is connected, and since $A_{(uv)(ij)} > 0$ if and only if $B_{(ij)(uv)} > 0$, G is also strongly connected. Consequently the graph with node set $\mathcal{U}^2$ and adjacency matrix AB, and the graph with node set $\mathcal{I}^2$ and adjacency matrix BA are also strongly connected. This, in turn, is equivalent to saying that AB and BA are irreducible matrices. Finally, since AB and BA are square, non-negative, irreducible matrices, by the Perron-Frobenius theorem on non-negative matrices, the eigenspace corresponding to the eigenvalue $\lambda_{\max}$ of largest magnitude is of dimension one and contains an eigenvector whose components are all positive. Running two parallel *power iteration* methods on matrices AB and BA will therefore converge to the unique positive eigenvectors of AB and BA, associated to $\lambda_{\max}$ [8]. The convergence of this method is geometric with respect to $\frac{|\lambda'_{\max}|}{|\lambda_{\max}|} < 1$, where $\lambda'_{\max}$ is the eigenvalue of second largest magnitude. $\square$

Following Theorem 2, the similarity values can be computed by repeating the following two steps until convergence:

1. Update the *normalized* user similarities with:
$$\vec{x} \leftarrow \frac{A\vec{y}}{||A\vec{y}||},$$
2. Update the *normalized* item similarities with:
$$\vec{y} \leftarrow \frac{B\vec{x}}{||B\vec{x}||}.$$

Once again, this approach usually converges within a few iterations and the complexity of each iteration is reduced by the fact that matrices A and B are normally quite sparse.

4. EXPERIMENTAL EVALUATION

In this section, we evaluate our approach on the task of predicting the ratings of users for movies and jokes. As it is tailored to compute similarities, and not specifically to predict ratings, it should be recognized that our approach is not directly comparable with state-of-the-art methods for this task. Yet, evaluating our approach on this problem still provides valuable information, as it allows us to measure the quality of its computed similarities. To this end, we compare the similarities obtained by our method with those computed with correlation-based and dimensionality reduction methods, in the nearest-neighbor prediction of ratings. Since all three types of similarities use the same approach to predict ratings, more accurate predictions indicate more relevant similarity values.

4.1 Tested methods

In our experiments we compared three methods to compute similarities. The first one, called ESR (*Enhanced SimRank*), is the approach described in this paper. For these experiments, we used $Z_{uv} = |\mathcal{I}_u||\mathcal{I}_v|$ and $Z_{ij} = |\mathcal{U}_i||\mathcal{U}_j|$ as normalization constants and the Gaussian RBF kernel of (5) with $\gamma = 0.05$ as the rating agreement function. This kernel has the advantage of being non-negative (*see Theorem* 2). However, this kernel was used in a slightly different way for matrices A and B. Thus, for A, the kernel was computed on the *normalized* ratings $(r_{ui} - \overline{r}_u)/(r_{\max} - r_{\min})$, where $\overline{r}_u$ is the average rating given by user u and $r_{\min}$, $r_{\max}$ are the minimum and maximum values of the rating range. For B, however, the kernel was computed on ratings normalized as $(r_{ui} - \overline{r}_i)/(r_{\max} - r_{\min})$, where $\overline{r}_i$ is the average rating given to item i. Finally, we used $\alpha = 0.95$ as the blending factor and defined the *a priori* similarity values as

$$\hat{s}(u,v) \ \ (\text{resp. } \hat{s}(i,j)) = \begin{cases} 1.0, & \text{if } u = v \ (\text{resp. } i = j), \\ 0.1, & \text{otherwise.} \end{cases}$$

These parameter values were selected based on cross-validation.

The second method, denoted by PCC, is the Pearson correlation similarity. Following the literature (e.g., see [24]), we computed user similarities as

$$s(u,v) \ = \ \frac{\sum\limits_{i \in \mathcal{I}_{uv}} (r_{ui} - \overline{r}_u)(r_{vi} - \overline{r}_v)}{\sqrt{\sum\limits_{i \in \mathcal{I}_{uv}} (r_{ui} - \overline{r}_u)^2 \sum\limits_{i \in \mathcal{I}_{uv}} (r_{vi} - \overline{r}_v)^2}}, \qquad (15)$$

where $\mathcal{I}_{uv} = \mathcal{I}_u \cap \mathcal{I}_v$. Likewise, the item similarities were obtained as follows:

$$s(i,j) \ = \ \frac{\sum\limits_{u \in \mathcal{U}_{ij}} (r_{ui} - \overline{r}_i)(r_{uj} - \overline{r}_j)}{\sqrt{\sum\limits_{u \in \mathcal{U}_{ij}} (r_{ui} - \overline{r}_i)^2 \sum\limits_{u \in \mathcal{U}_{ij}} (r_{uj} - \overline{r}_j)^2}}, \qquad (16)$$

where $\mathcal{U}_{ij} = \mathcal{U}_i \cap \mathcal{U}_j$,

Finally, the third method, called SVD, is based on the decomposition of the rating matrix [25]. Similar to the approach described in [27], we represented each user u by a vector $\vec{p}_u \in \mathbb{R}^f$ and each item by a vector $\vec{q}_i \in \mathbb{R}^f$, where f is the dimensionality of the latent space. Vectors $\vec{p}_u$ and $\vec{q}_i$ were then learned from the data by solving the following problem:

$$\begin{aligned} &\min_{\vec{p}_\cdot, \vec{q}_\cdot} \sum_{z_{ui} \in \mathcal{D}} \big(z_{ui} - \vec{p}_u^{\top} \vec{q}_i\big)^2 \\ &\text{s.t. } ||\vec{p}_u|| = ||\vec{q}_i|| = 1, \ \forall u \in \mathcal{U}, \ \forall i \in \mathcal{I}, \end{aligned} \qquad (17)$$

where $z_{ui} = (r_{ui} - \overline{r}_i)/(r_{\max} - r_{\min})$. This problem corresponds to finding, for each user u and item i, coordinates on the surface of the f-dimensional unit sphere such that u will give a high rating to i if their coordinates are near on the surface[1]. If two users u and v are nearby on the surface, then they will give similar ratings to the same items, and, thus, the similarity between these users can be computed as $s(u,v) = \vec{p}_u^{\top} \vec{p}_v$. Likewise, the similarity between two items i and j can be obtained as $s(i,j) = \vec{q}_i^{\top} \vec{q}_j$. Based on cross-validation, we have used $f = 50$ in our experiments.

The similarities obtained with these three methods were used to predict ratings r_{ui} in two different ways. In the first approach, called *user-based* prediction [16], the K nearest-neighbors of u that have rated i, denoted by $\mathcal{N}_i(u)$, are found with the users similarities. The ratings of these users for i are then used to predict r_{ui} as

$$\hat{r}_{ui} \ = \ \overline{r}_u \ + \ \frac{\sum\limits_{v \in \mathcal{N}_i(u)} s(u,v) \cdot (r_{vi} - \overline{r}_v)}{\sum\limits_{v \in \mathcal{N}_i(u)} |s(u,v)|}. \qquad (18)$$

The second approach, known as *item-based* prediction [4], instead uses the item similarities to find the K nearest-neighbors of item i that have been rated by u, denoted by

[1]This differs from the approach described in [27], where the norm of the user and item vectors is free but *penalized* through regularization.

$\mathcal{N}_u(i)$, and predicts ratings as

$$\hat{r}_{ui} \;=\; \overline{r}_i \;+\; \frac{\sum\limits_{j \in \mathcal{N}_u(i)} s(i,j) \cdot (r_{uj} - \overline{r}_j)}{\sum\limits_{j \in \mathcal{N}_u(i)} |s(i,j)|}. \qquad (19)$$

In the experiments presented in this section, we used $K = 50$ as the number of nearest-neighbors considered in the prediction.

4.2 Benchmark datasets

We tested the prediction approaches on three different real-life datasets, *MovieLens*[2], *Netflix*[3] and *Jester*[4], coming from systems recommending movies and jokes. The properties of these datasets are given in Table 1. Compared to the other two, the *Jester* dataset is particularly dense, with 410,000 ratings per joke on average. This dataset also differs from the others by the fact that its rating scale is continuous.

Table 1: Properties of the benchmark datasets.

Property	*MovieLens*	*Netflix*	*Jester*
Type	Movies	Movies	Jokes
Users	6,040	480,189	72,421
Items	3,952	17,770	100
Ratings	1 M	100 M	4.1 M
Range	[1,5]	[1,5]	[-10,10]
Integer	Yes	Yes	No

To generate datasets of various sparsity levels, we randomly selected 5,000 users from the *Netflix* and *Jester* datasets, and discarded the ratings that were not made by these users (the ratings of the *MovieLens* dataset were all kept). Then, for all three datasets, we sub-sampled the ratings of the remaining users by randomly selecting a user $u \in \mathcal{U}$ with a probability proportional to $|\mathcal{I}_u|$ and randomly removed one of its ratings from $\mathcal{I}_u$. We repeated this sub-sampling process until $|\mathcal{U}| \times \rho_u$ ratings were left, where ρ_u is the desired average number of ratings per user. To avoid having users with too few ratings, however, we allowed removing a rating from user u only if $|\mathcal{I}_u| > 0.5 \times \rho_u$. Using an average number of ratings ρ_u of 5, 10, 15 and 20, we obtained with this approach four subsets for each of the *MovieLens*, *Netflix* and *Jester* datasets. Note that, although the *MovieLens* and *Netflix* datasets contain information on the users and movies, as well as timestamps indicating when the ratings were made, we did not take such information into account in these experiments.

To assess the performance of these strategies, we used a 10-fold cross-validation scheme, where the dataset $\mathcal{D}$ was randomly split in 10 equal sized subsets $\mathcal{D}_k$, $k = 1, \ldots, 10$. For each k, we used $\bigcup_{l \neq k} \mathcal{D}_l$ to compute the user and item similarities (training phase) and then evaluated the *Mean Absolute Error* (MAE) and the *Root Mean Squared Error* (RMSE) on subset $\mathcal{D}_k$. The reported error values were taken as the mean errors over all 10 subsets.

4.3 Prediction results

[2]`http://www.grouplens.org/`

[3]http://www.netflixprize.com/

[4]`http://www.ieor.berkeley.edu/~goldberg/jester-data/`

Tables 2, 3, 4 present the results for the six rating prediction methods on the *MovieLens*, *Netflix* and *Jester* data subsets. The lower the MAE and RMSE values, the more accurate are the methods at predicting ratings. Moreover, the #NN values give the average number of neighbors used in the predictions. A low value indicates that a significant portion of the user or item similarities are equal to zero, due to data sparsity.

Table 2: Average MAE and RMSE (and corresponding standard deviation) obtained for the *MovieLens* data subsets with average number of ratings per user $\rho_u \in \{5, 10, 15, 20\}$. #NN gives the average number of neighbors used in the predictions.

ρ_u	Result	USER-BASED PREDICTION		
		PCC	SVD	ESR
5	MAE	0.934 (.011)	0.870 (.014)	**0.854** (.011)
	RMSE	1.236 (.012)	1.156 (.017)	**1.128** (.012)
	#NN	0.7	24.5	24.5
10	MAE	0.897 (.009)	0.798 (.010)	**0.783** (.009)
	RMSE	1.170 (.009)	1.060 (.010)	**1.036** (.011)
	#NN	8.2	34.1	34.1
15	MAE	0.841 (.008)	0.776 (.006)	**0.762** (.010)
	RMSE	1.104 (.010)	1.033 (.009)	**1.006** (.010)
	#NN	19.7	38.7	38.7
20	MAE	0.807 (.007)	0.773 (.005)	**0.753** (.006)
	RMSE	1.063 (.006)	1.027 (.007)	**0.991** (.005)
	#NN	29.3	41.4	41.4

ρ_u	Result	ITEM-BASED PREDICTION		
		PCC	SVD	ESR
5	MAE	0.857 (.018)	0.883 (.018)	**0.811** (.011)
	RMSE	1.134 (.017)	1.162 (.017)	**1.076** (.012)
	#NN	0.6	4.2	4.2
10	MAE	0.860 (.004)	0.832 (.008)	**0.754** (.010)
	RMSE	1.133 (.007)	1.096 (.011)	**1.005** (.012)
	#NN	3.9	10.4	10.4
15	MAE	0.818 (.008)	0.803 (.007)	**0.735** (.008)
	RMSE	1.079 (.010)	1.061 (.007)	**0.979** (.010)
	#NN	8.1	15.6	15.6
20	MAE	0.785 (.007)	0.786 (.010)	**0.723** (.006)
	RMSE	1.039 (.007)	1.038 (.011)	**0.965** (.007)
	#NN	13.3	21.1	21.1

From these results, we can see that the similarity values obtained by our method leads to more accurate predictions than those of the SVD method, even though these predictions were made with the same number of neighbors. Moreover, compared to PCC, our method also leads to better results on the sparser datasets *MovieLens* and *Netflix*. However, in the denser *Jester* dataset, PCC similarities produce more accurate predictions for $\rho_u = 15$ and $\rho_u = 20$. Even though we have used only a sub-sample of the ratings, one should note that the *Jester* data subsets tested in our experiments are still very dense. Thus, for $\rho_u = 15$, users still have rated on average 15% of the jokes. Nevertheless, the results of this experiments seem to indicate that our method provides better similarity values when the data is sparse, but correlation based approaches might be superior when a large number of ratings is available.

5. SUMMARY AND FUTURE WORKS

This paper presented a novel approach to compute similarities, related to the *SimRank* algorithm. Like this algorithm, our approach uses a formulation that associates similarities

Table 3: Average MAE and RMSE (and corresponding standard deviation) for the *Netflix* data subsets with average number of ratings per user $\rho_u \in \{5, 10, 15, 20\}$. #NN gives the average number of neighbors used in the predictions.

ρ_u	Result	USER-BASED PREDICTION		
		PCC	SVD	ESR
5	MAE	0.914 (.016)	0.896 (.020)	**0.877** (.020)
	RMSE	1.216 (.021)	1.190 (.021)	**1.166** (.022)
	#NN	0.5	18.7	18.7
10	MAE	0.890 (.013)	0.845 (.007)	**0.811** (.011)
	RMSE	1.170 (.014)	1.117 (.007)	**1.081** (.012)
	#NN	5.2	26.9	26.9
15	MAE	0.867 (.008)	0.832 (.011)	**0.790** (.011)
	RMSE	1.134 (.011)	1.102 (.011)	**1.055** (.011)
	#NN	12.9	31.0	31.0
20	MAE	0.839 (.007)	0.824 (.007)	**0.776** (.008)
	RMSE	1.103 (.009)	1.090 (.007)	**1.037** (.009)
	#NN	20.5	33.7	33.7

ρ_u	Result	ITEM-BASED PREDICTION		
		PCC	SVD	ESR
5	MAE	0.929 (.012)	0.960 (.019)	**0.881** (.011)
	RMSE	1.220 (.015)	1.247 (.021)	**1.164** (.016)
	#NN	0.4	4.3	4.3
10	MAE	0.920 (.010)	0.894 (.013)	**0.819** (.007)
	RMSE	1.213 (.011)	1.171 (.012)	**1.086** (.010)
	#NN	2.5	10.6	10.6
15	MAE	0.893 (.010)	0.867 (.008)	**0.792** (.011)
	RMSE	1.175 (.011)	1.137 (.010)	**1.058** (.013)
	#NN	5.5	15.9	15.9
20	MAE	0.860 (.006)	0.848 (.005)	**0.776** (.005)
	RMSE	1.138 (.005)	1.114 (.008)	**1.039** (.006)
	#NN	9.2	21.4	21.4

Table 4: Average MAE and RMSE (and corresponding standard deviation) for the *Jester* data subsets with average number of ratings per user $\rho_u \in \{5, 10, 15, 20\}$. #NN gives the average number of neighbors used in the predictions.

ρ_u	Result	USER-BASED PREDICTION		
		PCC	SVD	ESR
5	MAE	4.076 (.072)	3.940 (.050)	**3.896** (.064)
	RMSE	5.194 (.081)	5.063 (.079)	**5.017** (.073)
	#NN	39.5	50.0	50.0
10	MAE	3.710 (.059)	3.675 (.053)	**3.655** (.062)
	RMSE	4.695 (.067)	4.702 (.054)	**4.651** (.074)
	#NN	50.0	50.0	50.0
15	MAE	3.665 (.035)	**3.571** (.029)	3.581 (.038)
	RMSE	4.617 (.049)	4.567 (.032)	**4.538** (.049)
	#NN	50.0	50.0	50.0
20	MAE	3.634 (.018)	**3.505** (.019)	3.541 (.015)
	RMSE	4.568 (.027)	4.490 (.024)	**4.480** (.025)
	#NN	50.0	50.0	50.0

ρ_u	Result	ITEM-BASED PREDICTION		
		PCC	SVD	ESR
5	MAE	4.060 (.047)	4.714 (.083)	**3.809** (.058)
	RMSE	5.212 (.065)	5.953 (.109)	**4.891** (.069)
	#NN	4.1	4.1	4.1
10	MAE	**3.588** (.052)	4.345 (.042)	3.603 (.055)
	RMSE	**4.587** (.069)	5.410 (.041)	4.592 (.067)
	#NN	9.1	9.1	9.1
15	MAE	**3.476** (.039)	4.193 (.038)	3.539 (.039)
	RMSE	**4.434** (.050)	5.184 (.038)	4.493 (.051)
	#NN	13.9	13.9	13.9
20	MAE	**3.431** (.020)	4.143 (.031)	3.511 (.018)
	RMSE	**4.365** (.028)	5.105 (.032)	4.444 (.027)
	#NN	18.9	18.9	18.9

between linked objects of two different sets. However, unlike *SimRank* and its extensions, our approach allows one to model the agreement between link values using any desired function. Moreover, it also provides an elegant way to integrate prior information on the similarity values directly in the computations.

To illustrate its usefulness, we have described how this approach can be used to evaluate the similarities between the users or the items of a recommender system, based on the ratings of users on items. In contrast to the traditional methods using rating correlation, our approach has the benefit of considering all the available ratings made by two users, making possible the computation of similarities between users that have rated different items. Also, as opposed to more recent recommendation methods, this approach is not limited to numerical ratings and provides a simple way to integrate information on item content or user profile similarity. Finally, experiments conducted on the problem of predicting new ratings on three different real-life datasets have shown the similarities obtained with our approach to lead to more accurate predictions than those obtained by two other methods based on Pearson correlation and on SVD, when the data is sparse.

In future works, we would like to deeper investigate the impact of using prior knowledge on the similarities, for instance, obtained from user profiles and item content. Moreover, we also consider defining and evaluating other types of rating agreement functions, in particular, in the setting where ratings are non-numerical.

6. REFERENCES

[1] C. C. Aggarwal, J. L. Wolf, K.-L. Wu, and P. S. Yu. Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In *KDD '99: Proc. of the 5th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 201–212, New York, NY, USA, 1999. ACM.

[2] I. Antonellis, H. G. Molina, and C. C. Chang. Simrank++: query rewriting through link analysis of the click graph. *Proc. of the VLDB Endowment*, 1(1):408–421, 2008.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[4] M. Deshpande and G. Karypis. Item-based top-N recommendation algorithms. *ACM Transaction on Information Systems*, 22(1):143–177, 2004.

[5] C. Desrosiers and G. Karypis. Within-network classification using local structure similarity. In *ECML PKDD '09: Proc. of the European Conf. on Machine Learning and Knowledge Discovery in Databases*, pages 260–275, Berlin, Heidelberg, 2009. Springer-Verlag.

[6] F. Fouss, A. Pirotte, and M. Saerens. A novel way of computing similarities between nodes of a graph, with application to collaborative recommendation. In *WI '05: Proc. of the 2005 IEEE/WIC/ACM Int. Conf. on Web Intelligence*, pages 550–556, Washington, DC, USA, 2005. IEEE Computer Society.

[7] F. Fouss, J.-M. Renders, A. Pirotte, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):355–369, 2007.

[8] G. H. Golub and C. F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, 1996.

[9] M. Gori and A. Pucci. Itemrank: a random-walk based scoring algorithm for recommender engines. In *Proc. of the 2007 IJCAI Conf.*, pages 2766–2771, 2007.

[10] M. Grcar, B. Fortuna, D. Mladenic, and M. Grobelnik. knn versus svm in the collaborative filtering framework. *Data Science and Classification*, pages 251–260, 2006.

[11] T. Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *SIGIR '03: Proc. of the 26th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 259–266, New York, NY, USA, 2003. ACM.

[12] Z. Huang, H. Chen, and D. Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems*, 22(1):116–142, 2004.

[13] T. Ito, M. Shimbo, T. Kudo, and Y. Matsumoto. Application of kernels to link analysis. In *KDD '05: Proc. of the eleventh ACM SIGKDD Int. Conf. on Knowledge Discovery in Data Mining*, pages 586–592, New York, NY, USA, 2005. ACM.

[14] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *KDD '02: Proc. of the eighth ACM SIGKDD Int. Conf. on Knowledge discovery and data mining*, pages 538–543, New York, NY, USA, 2002. ACM.

[15] R. I. Kondor and J. D. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *ICML '02: Proc. of the Nineteenth Int. Conf. on Machine Learning*, pages 315–322, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[16] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. GroupLens: applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

[17] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD'08: Proc. of the 14th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 426–434, New York, NY, USA, 2008. ACM.

[18] J. Kunegis, A. Lommatzsch, and C. Bauckhage. Alternative similarity functions for graph kernels. In *Proc. of the Int. Conf. on Pattern Recognition*, 2008.

[19] C. Li, J. Han, G. He, X. Jin, Y. Sun, Y. Yu, and T. Wu. Fast computation of simrank for static and dynamic information networks. In *EDBT '10: Proc. of the 13th Int. Conf. on Extending Database Technology*, pages 465–476, New York, NY, USA, 2010. ACM.

[20] Z. Lin, M. R. Lyu, and I. King. Matchsim: a novel neighbor-based similarity measure with maximum neighborhood matching. In *CIKM '09: Proc. of the 18th ACM Conf. on Information and Knowledge Management*, pages 1613–1616, New York, NY, USA, 2009. ACM.

[21] D. Lizorkin, P. Velikhov, M. Grinev, and D. Turdakov. Accuracy estimate and optimization techniques for simrank computation. volume 1, pages 422–433. VLDB Endowment, 2008.

[22] H. Luo, C. Niu, R. Shen, and C. Ullrich. A collaborative filtering framework based on both local user similarity and global user similarity. *Machine Learning*, 72(3):231–245, 2008.

[23] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *ICML '07: Proc. of the 24th Int. Conf. on Machine learning*, pages 791–798, New York, NY, USA, 2007. ACM.

[24] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proc. of the 10th Int. Conf. on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM.

[25] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Application of dimensionality reduction in recommender systems - A case study. In *ACM WebKDD Workshop*, 2000.

[26] A. Smola and R. Kondor. Kernels and regularization on graphs. In *Proc. of the 2003 Conf. on Computational Learning Theory (COLT) and Kernels Workshop*, pages 144–158. M.Warmuth and B. Schölkopf, 2003.

[27] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Major components of the gravity recommendation system. *SIGKDD Exploration Newsletter*, 9(2):80–83, 2007.

[28] H. Yildirim and M. S. Krishnamoorthy. A random walk method for alleviating the sparsity problem in collaborative filtering. In *RecSys '08: Proc. of the 2008 ACM Conf. on Recommender systems*, pages 131–138, New York, NY, USA, 2008. ACM.