A Novel Approach to Compute Similarities and Its Application to Item Recommendation

Christian Desrosiers[1] and George Karypis[2]

[1] Computer Engineering & IT dept., Ecole de Technologie Superieure, Montreal, Canada
christian.desrosiers@etsmtl.ca

[2] Computer Science & Engineering dept., University of Minnesota, Minneapolis, USA
karypis@cs.umn.edu

Abstract. Several key applications like recommender systems deal with data in the form of ratings made by users on items. In such applications, one of the most crucial tasks is to find users that share common interests, or items with similar characteristics. Assessing the similarity between users or items 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. It has been recognized that popular methods to compute similarities, based on correlation, are not suitable for this task when the rating data is sparse. This paper presents a novel approach, based on the *SimRank* algorithm, to compute similarity values when ratings are limited. Unlike correlation-based methods, which only consider user ratings for common items, this approach uses all the available ratings, allowing it to compute meaningful similarities. To evaluate the usefulness of this approach, we test it on the problem of predicting the ratings of users for movies and jokes.

1 Introduction

Many important applications like recommender systems deal with data in the form of ratings made by users on items. In such applications, one of the most crucial tasks is to find users that share common interests, or items with similar characteristics. Assessing the similarity between users or items 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 two users, found in many collaborative filtering recommender systems, is based on the correlation between the ratings made by these users on common items. As recognized by several recent works on this topic, such as [5,18], 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 recent approaches based on dimensionality reduction and graph theory have been proposed for this problem, they also have their limitations. For example, they cannot be used in situations where there are categorical

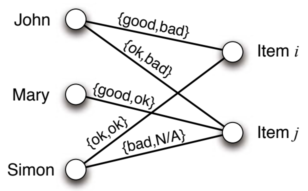

Fig. 1. A bipartite graph representing responses (sets of categorical values) given by users to items

ratings or other non-numerical rating types, such as the one shown in Figure 1, and do not provide an easy way to integrate prior information on the similarities.

This paper presents a novel approach to compute similarities between users or items when only a limited number of ratings are available. Based on the well-known algorithm *SimRank* [9], this approach models the relations between user similarities and item similarities using a system of linear 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 ratings, 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 Item Recommendation and Sparsity

Sparsity is a problem occurring frequently in recommender systems when many users have provided ratings to a limited number of items, or many items have received only a few ratings. A solution proposed for this problem consists in using item content information to enhance the computation of similarities [10,14]. However, reliable content information may not be available, for example, if obtaining this information requires expensive resources (e.g., hand made annotations) or is simply too difficult (e.g., audio or video data).

Dimensionality reduction methods have also been developed to alleviate the problem of sparsity. These methods work by decomposing the user-item rating matrix [2,17] or a sparse similarity matrix [5,6] into a limited number of latent factors. These factors, which represent high-level characteristics of users and items, are then used to predict new ratings. While decomposition approaches are among the most accurate rating prediction methods, they generally lack the ability to discover local relations in the data. Moreover, this class of techniques can only be used with numerical ratings, not categorical ones.

Another category of methods proposed for recommending items in sparse data uses graph theory to model the interactions between users and items and measure the strength of these relations. Such methods include approaches based on geodesic distance [15], diffusion kernels [11], and random walks [5,8,18]. A common problem with these methods is their lack of interpretability and the difficulty of translating ratings into link weights, for instance, if the ratings are negative or non-numerical.

Finally, a different approach, proposed in [3], computes item similarities by solving a global regression problem which finds the similarity values that best predict known ratings using an item-based nearest-neighbor formulation. This approach has three main limitations. First, it relies on a correlation-based method to compute the nearest neighbors, which may be sensitive to sparsity. Also, the item-based formulation used in this approach only considers the ratings made by common users, which also creates problems when the rating data is sparse. Finally, the item similarities computed by this method depend on the rating that is predicted, which is not suitable to the task of finding general similarities between all items.

2.2 SimRank

The method introduced in this paper is closely related to the bipartite version of the *SimRank* algorithm proposed by Jeh and Widom [9]. 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 items purchased by a given user $u \in \mathcal{U}$, and let $\mathcal{U}_i \subseteq \mathcal{U}$ be the set of users that have purchased an item $i \in \mathcal{I}$. The similarity between two users u and v, $s(u, v)$, is obtained as the average similarity of the items purchased by these users:

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

where $C_1 \in [0, 1]$ is a constant controlling the flow of similarity values on the graph links. Likewise, the similarity between two items i and j, $s(i, j)$, can be computed as the average similarity of users that have purchased these items:

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

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.

A significant limitation of this approach, in the context of item recommendation, is that it only considers the interactions between users and items (e.g., purchases) but not the ratings. Another method called *SimRank*++, recently proposed in [1], 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:

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

where w_{ui} is the normalized weight of the link between u and i. Like *SimRank*, this method also has some limitations. First, since link weights are simply multiplied it may not be possible to compare the agreement between the ratings made by two users on similar items, especially if these ratings are non-numerical. Also, this method does not allow one to integrate prior knowledge on the similarity values, for instance, obtained by comparing the content of items.

2.3 Contributions

This paper makes the following contributions:

1. It describes a novel approach to compute similarities that extends the *SimRank* algorithm and its extensions in two important ways:
   (a) It uses an arbitrary function to compare the agreement between link weights, which allows the use of non-numerical ratings.
   (b) It provides an elegant way to integrate prior information on the similarity values directly in the computations.
2. Unlike similarity measures based on correlation which only use the ratings on common items, this approach considers all the available ratings, allowing it to compute similarities between users that have rated different items, thereby reducing the sensitivity to sparse data.
3. It presents a first comprehensive experimental evaluation of a *SimRank*-based method on the problem of predicting new ratings.

3 A Novel Approach

3.1 The General Formulation

Consider the task of evaluating the similarity $s(u,v)$ between two users u and v. A simple approach, used in several item recommendation systems, is to compute $s(u,v)$ as the correlation between the ratings given by u and v on common items. Besides being limited to numerical ratings, this approach has another significant problem: similarities can only be evaluated for users that have rated common items, and the correlation values are only significant if there is a sufficient number of common items. For these reasons, the correlation approach gives poor results when the rating data is sparse.

As in *SimRank*, our approach overcomes these limitations by using all the ratings given by u and v, not only those given to common items. Thus, we evaluate the similarity between users u and v as the average rating agreement for all pairs of rated items, weighted by the similarity of these items:

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

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

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

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

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

$\overline{r}_u$ and σ_u being the mean and standard deviation of the ratings given by u. 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 two ratings 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 rating types. Moreover, this formulation can be easily extended to include prior information on the similarity between users u and v, obtained, for example, by comparing their profiles (*gender, age, etc.*). Denote $\hat{s}(u,v)$ the *a priori* similarity capturing this information, (4) can be extended to include $\hat{s}(u,v)$ as

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

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

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

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

3.2 Modeling Similarities as a Linear 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. Denote the user and item similarities as vectors $\boldsymbol{x} \in \mathbb{R}^{|\mathcal{U}|^2}$ and $\boldsymbol{y} \in \mathbb{R}^{|\mathcal{I}|^2}$ such that each pair of users u,v is mapped to a unique element $\boldsymbol{x}_{(uv)} = s(u,v)$, and each pair of items i,j maps to a unique element $\boldsymbol{y}_{(ij)} = s(i,j)$. Also, let $\boldsymbol{c} \in \mathbb{R}^{|\mathcal{U}|^2}$ and $\boldsymbol{d} \in \mathbb{R}^{|\mathcal{I}|^2}$ be vectors such that $\boldsymbol{c}_{(uv)} = \hat{s}(u,v)$ and $\boldsymbol{d}_{(ij)} = \hat{s}(i,j)$. Moreover, define A as the $(|\mathcal{U}|^2 \times |\mathcal{I}|^2)$ matrix such that $A_{(uv)(ij)} = \frac{1}{Z_{uv}} k(r_{ui}, r_{vj})$, if $i \in \mathcal{I}_u$ and $j \in \mathcal{I}_v$, and $A_{(uv)(ij)} = 0$ otherwise. Likewise, let B is a $(|\mathcal{I}|^2 \times |\mathcal{U}|^2)$ matrix such that $B_{(ij)(uv)} = \frac{1}{Z_{ij}} k(r_{ui}, r_{vj})$ if $u \in \mathcal{U}_i$ and $v \in \mathcal{U}_j$, and $B_{(ij)(uv)} = 0$ otherwise.

The linear system formed of equations (7) and (8) can thus be written in matrix form as

$$\begin{pmatrix} \boldsymbol{x} \\ \hline \boldsymbol{y} \end{pmatrix} = (1-\alpha) \begin{pmatrix} \boldsymbol{c} \\ \hline \boldsymbol{d} \end{pmatrix} + \alpha \left(\begin{array}{c|c} 0 & A \\ \hline B & 0 \end{array}\right) \begin{pmatrix} \boldsymbol{x} \\ \hline \boldsymbol{y} \end{pmatrix}, \tag{9}$$

and has the following solution:

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

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

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 these matrices [11,13]:

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

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

$$\boldsymbol{x} = (1-\alpha) \left(\sum_{n=0}^{\infty} (\alpha^2 AB)^n \boldsymbol{c} + \alpha A \sum_{n=0}^{\infty} (\alpha^2 BA)^n \boldsymbol{d} \right) = \left(\sum_{n=0}^{\infty} (\alpha^2 AB)^n \right) \boldsymbol{p}. \tag{11}$$

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

$$\boldsymbol{y} = \left(\sum_{n=0}^{\infty} (\alpha^2 BA)^n \right) \boldsymbol{q}, \tag{12}$$

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

This new formulation leads to a simple method to compute $\boldsymbol{x}$ and $\boldsymbol{y}$. Since a similar approach can be used for $\boldsymbol{y}$, we limit our presentation to the computation of $\boldsymbol{x}$. First, the method initializes $\boldsymbol{x}$ to the *null* vector and initializes a temporary vector $\boldsymbol{w}$ to $\boldsymbol{p}$. Then, the following two steps are repeated until convergence or a maximum number of iterations is reached:

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

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

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

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

To analyze the complexity of this approach, as observed in most recommender systems, we suppose the number of ratings given by any user to be bounded by a constant m independent of the number of 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 ratings among the users and items, 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 $\boldsymbol{x}$ and $\boldsymbol{p}$, the expected space complexity of the method is $O(|\mathcal{U}|^2)$. For the time complexity, the dominant operations are the two matrix multiplications: $B\boldsymbol{w} = \boldsymbol{w}'$ and $A\boldsymbol{w}'$. 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 $\boldsymbol{c}$ and $\boldsymbol{d}$, for instance $\boldsymbol{c}_{(uv)} = 1$ 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 $\boldsymbol{x}$ and $\boldsymbol{y}$ 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 [7].*

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

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

1. Update the *normalized* user similarities: $\boldsymbol{x} \leftarrow A\boldsymbol{y} \;/\; ||A\boldsymbol{y}||$,
2. Update the *normalized* item similarities: $\boldsymbol{y} \leftarrow B\boldsymbol{x} \;/\; ||B\boldsymbol{x}||$.

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 in sparse data, 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 SVD 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. 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 [16]), we computed user similarities as

$$s(u,v) \;=\; \frac{\sum\limits_{i \in \mathcal{I}_{uv}} (r_{ui} - \overline{r}_u)(r_{vi} - \overline{r}_v)}{\sqrt{\sum\limits_{i \in \mathcal{I}_{uv}} (r_{ui} - \overline{r}_u)^2 \sum\limits_{i \in \mathcal{I}_{uv}} (r_{vi} - \overline{r}_v)^2}}. \tag{13}$$

and the item similarities as

$$s(i,j) \;=\; \frac{\sum\limits_{u \in \mathcal{U}_{ij}} (r_{ui} - \overline{r}_i)(r_{uj} - \overline{r}_j)}{\sqrt{\sum\limits_{u \in \mathcal{U}_{ij}} (r_{ui} - \overline{r}_i)^2 \sum\limits_{u \in \mathcal{U}_{ij}} (r_{uj} - \overline{r}_j)^2}}. \tag{14}$$

Finally, the third method, called SVD, is based on the decomposition of the rating matrix. Like the approach described in [17], we represented each user u by a vector $\boldsymbol{p}_u \in \mathbb{R}^f$ and each item by a vector $\boldsymbol{q}_i \in \mathbb{R}^f$, where f is the dimensionality of the latent space. Vectors $\boldsymbol{p}_u$ and $\boldsymbol{q}_i$ were then learned from the data by solving the following problem:

$$\min_{\boldsymbol{p}_\cdot, \boldsymbol{q}_\cdot} \sum_{z_{ui} \in \mathcal{D}} \left(z_{ui} - \boldsymbol{p}_u^\top \boldsymbol{q}_i \right)^2 \quad \text{s.t. } ||\boldsymbol{p}_u|| = ||\boldsymbol{q}_i|| = 1, \; \forall u \in \mathcal{U}, \; \forall i \in \mathcal{I}, \tag{15}$$

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 close together on the surface. 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 $s(u,v) = \boldsymbol{p}_u^\top \boldsymbol{p}_v$. Likewise, the similarity between two items i and j can be obtained as $s(i,j) = \boldsymbol{q}_i^\top \boldsymbol{q}_j$. 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 [12], 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

$$\hat{r}_{ui} \;=\; \overline{r}_u \;+\; \sum_{v\in\mathcal{N}_i(u)} s(u,v)\cdot(r_{vi} - \overline{r}_v) \;\;/ \sum_{v\in\mathcal{N}_i(u)} |s(u,v)|. \qquad (16)$$

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

$$\hat{r}_{ui} \;=\; \overline{r}_i \;+\; \sum_{j\in\mathcal{N}_u(i)} s(i,j)\cdot(r_{uj} - \overline{r}_j) \;\;/ \sum_{j\in\mathcal{N}_u(i)} |s(i,j)|. \qquad (17)$$

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*[1], *Netflix*[2] and *Jester*[3], 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

Dataset	Type	Nb. users	Nb. items	Nb. ratings	Rating range
MovieLens	Movies	6,040	3,952	1 M	$\{1,2,3,4,5\}$
Netflix	Movies	480,189	17,770	100 M	$\{1,2,3,4,5\}$
Jester	Jokes	72,421	100	4.1 M	$[-10,10]$

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,

[1] `http://www.grouplens.org/`
[2] `http://www.netflixprize.com/`
[3] `http://www.ieor.berkeley.edu/~goldberg/jester-data/`

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.

MovieLens Data Subsets

ρ_u	Result	USER-BASED PREDICTION			ITEM-BASED PREDICTION		
		PCC	SVD	ESR	PCC	SVD	ESR
5	MAE	0.934 (.011)	0.870 (.014)	**0.854** (.011)	0.857 (.018)	0.883 (.018)	**0.811** (.011)
	RMSE	1.236 (.012)	1.156 (.017)	**1.128** (.012)	1.134 (.017)	1.162 (.017)	**1.076** (.012)
	#NN	0.7	24.5	24.5	0.6	4.2	4.2
10	MAE	0.897 (.009)	0.798 (.010)	**0.783** (.009)	0.860 (.004)	0.832 (.008)	**0.754** (.010)
	RMSE	1.170 (.009)	1.060 (.010)	**1.036** (.011)	1.133 (.007)	1.096 (.011)	**1.005** (.012)
	#NN	8.2	34.1	34.1	3.9	10.4	10.4
15	MAE	0.841 (.008)	0.776 (.006)	**0.762** (.010)	0.818 (.008)	0.803 (.007)	**0.735** (.008)
	RMSE	1.104 (.010)	1.033 (.009)	**1.006** (.010)	1.079 (.010)	1.061 (.007)	**0.979** (.010)
	#NN	19.7	38.7	38.7	8.1	15.6	15.6
20	MAE	0.807 (.007)	0.773 (.005)	**0.753** (.006)	0.785 (.007)	0.786 (.010)	**0.723** (.006)
	RMSE	1.063 (.006)	1.027 (.007)	**0.991** (.005)	1.039 (.007)	1.038 (.011)	**0.965** (.007)
	#NN	29.3	41.4	41.4	13.3	21.1	21.1

Netflix Data Subsets

ρ_u	Result	USER-BASED PREDICTION			ITEM-BASED PREDICTION		
		PCC	SVD	ESR	PCC	SVD	ESR
5	MAE	0.914 (.016)	0.896 (.020)	**0.877** (.020)	0.929 (.012)	0.960 (.019)	**0.881** (.011)
	RMSE	1.216 (.021)	1.190 (.021)	**1.166** (.022)	1.220 (.015)	1.247 (.021)	**1.164** (.016)
	#NN	0.5	18.7	18.7	0.4	4.3	4.3
10	MAE	0.890 (.013)	0.845 (.007)	**0.811** (.011)	0.920 (.010)	0.894 (.013)	**0.819** (.007)
	RMSE	1.170 (.014)	1.117 (.007)	**1.081** (.012)	1.213 (.011)	1.171 (.012)	**1.086** (.010)
	#NN	5.2	26.9	26.9	2.5	10.6	10.6
15	MAE	0.867 (.008)	0.832 (.011)	**0.790** (.011)	0.893 (.010)	0.867 (.008)	**0.792** (.011)
	RMSE	1.134 (.011)	1.102 (.011)	**1.055** (.011)	1.175 (.011)	1.137 (.010)	**1.058** (.013)
	#NN	12.9	31.0	31.0	5.5	15.9	15.9
20	MAE	0.839 (.007)	0.824 (.007)	**0.776** (.008)	0.860 (.006)	0.848 (.005)	**0.776** (.005)
	RMSE	1.103 (.009)	1.090 (.007)	**1.037** (.009)	1.138 (.005)	1.114 (.008)	**1.039** (.006)
	#NN	20.5	33.7	33.7	9.2	21.4	21.4

Jester Data Subsets

ρ_u	Result	USER-BASED PREDICTION			ITEM-BASED PREDICTION		
		PCC	SVD	ESR	PCC	SVD	ESR
5	MAE	4.076 (.072)	3.940 (.050)	**3.896** (.064)	4.060 (.047)	4.714 (.083)	**3.809** (.058)
	RMSE	5.194 (.081)	5.063 (.079)	**5.017** (.073)	5.212 (.065)	5.953 (.109)	**4.891** (.069)
	#NN	39.5	50.0	50.0	4.1	4.1	4.1
10	MAE	3.710 (.059)	3.675 (.053)	**3.655** (.062)	**3.588** (.052)	4.345 (.042)	3.603 (.055)
	RMSE	4.695 (.067)	4.702 (.054)	**4.651** (.074)	**4.587** (.069)	5.410 (.041)	4.592 (.067)
	#NN	50.0	50.0	50.0	9.1	9.1	9.1
15	MAE	3.665 (.035)	**3.571** (.029)	3.581 (.038)	**3.476** (.039)	4.193 (.038)	3.539 (.039)
	RMSE	4.617 (.049)	4.567 (.032)	**4.538** (.049)	**4.434** (.050)	5.184 (.038)	4.493 (.051)
	#NN	50.0	50.0	50.0	13.9	13.9	13.9
20	MAE	3.634 (.018)	**3.505** (.019)	3.541 (.015)	**3.431** (.020)	4.143 (.031)	3.511 (.018)
	RMSE	4.568 (.027)	4.490 (.024)	**4.480** (.025)	**4.365** (.028)	5.105 (.032)	4.444 (.027)
	#NN	50.0	50.0	50.0	18.9	18.9	18.9

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

4.3 Prediction Results

Figure 2 presents 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.

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 result 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. Like *SimRank*, our approach uses a formulation that associates similarities between linked objects of two different sets. However, our approach also allows one to model the agreement between link weights using any desired function and 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.

References

1. Antonellis, I., Molina, H.G., Chang, C.C.: Simrank++: query rewriting through link analysis of the click graph. Proceedings of the VLDB Endowment 1(1), 408–421 (2008)
2. Bell, R.M., Koren, Y., Volinsky, C.: Modeling relationships at multiple scales to improve accuracy of large recommender systems. In: KDD 2007: Proc. of the 13th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 95–104. ACM, New York (2007)
3. Bell, R.M., Koren, Y.: Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In: ICDM 2007: Proc. of the 2007 Seventh IEEE Int. Conf. on Data Mining, pp. 43–52. IEEE Computer Society, Washington (2007)
4. Deshpande, M., Karypis, G.: Item-based top-N recommendation algorithms. ACM Transaction on Information Systems 22(1), 143–177 (2004)
5. Fouss, F., Renders, J.-M., Pirotte, A., Saerens, M.: 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)
6. Goldberg, K., Roeder, T., Gupta, D., Perkins, C.: Eigentaste: A constant time collaborative filtering algorithm. Information Retrieval 4(2), 133–151 (2001)
7. Golub, G.H., Van Loan, C.F.: Matrix computations, 3rd edn. Johns Hopkins University Press, Baltimore (1996)
8. Gori, M., Pucci, A.: Itemrank: a random-walk based scoring algorithm for recommender engines. In: Proc. of the 2007 IJCAI Conf., pp. 2766–2771 (2007)
9. Jeh, G., Widom, J.: Simrank: a measure of structural-context similarity. In: KDD 2002: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 538–543. ACM, New York (2002)
10. Kim, B.M., Li, Q., Park, C.S., Kim, S.G., Kim, J.Y.: A new approach for combining content-based and collaborative filters. Journal of Intelligent Information Systems 27(1), 79–91 (2006)
11. Kondor, R.I., Lafferty, J.D.: Diffusion kernels on graphs and other discrete input spaces. In: ICML 2002: Proc. of the Nineteenth Int. Conf. on Machine Learning, pp. 315–322. Morgan Kaufmann Publishers Inc., San Francisco (2002)
12. Konstan, J.A., Miller, B.N., Maltz, D., Herlocker, J.L., Gordon, L.R., Riedl, J.: GroupLens: applying collaborative filtering to usenet news. Communications of the ACM 40(3), 77–87 (1997)
13. Kunegis, J., Lommatzsch, A., Bauckhage, C.: Alternative similarity functions for graph kernels. In: Proc. of the Int. Conf. on Pattern Recognition (2008)
14. Li, J., Zaiane, O.R.: Combining usage, content, and structure data to improve Web site recommendation. In: Bauknecht, K., Bichler, M., Pröll, B. (eds.) EC-Web 2004. LNCS, vol. 3182, pp. 305–315. Springer, Heidelberg (2004)
15. Luo, H., Niu, C., Shen, R., Ullrich, C.: A collaborative filtering framework based on both local user similarity and global user similarity. Machine Learning 72(3), 231–245 (2008)
16. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms. In: WWW 2001: Proc. of the 10th Int. Conf. on World Wide Web, pp. 285–295. ACM, New York (2001)
17. Takács, G., Pilászy, I., Németh, B., Tikk, D.: Major components of the gravity recommendation system. SIGKDD Exploration Newsletter 9(2), 80–83 (2007)
18. Yildirim, H., Krishnamoorthy, M.S.: A random walk method for alleviating the sparsity problem in collaborative filtering. In: RecSys 2008: Proc. of the 2008 ACM Conf. on Recommender systems, pp. 131–138. ACM, New York (2008)