Augmenting Collaborative Recommender by Fusing Explicit Social Relationships

Quan Yuan, Shiwan Zhao
IBM China Research Laboratory
Beijing, 100193, China
{quanyuan, zhaosw}@cn.ibm.com

Li Chen
Department of Computer Science, Hong Kong Baptist University
Hong Kong
lichen@comp.hkbu.edu.hk

Yan Liu
IBM T.J. Watson Research Center
Yorktown, NY 10598
liuya@us.ibm.com

Shengchao Ding
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, 100190, China
dingshengchao@ict.ac.cn

Xiatian Zhang
IBM China Research Laboratory
Beijing, 100193, China
xiatianz@cn.ibm.com

Wentao Zheng
IBM China Research Laboratory
Beijing, 100193, China
zhengwt@cn.ibm.com

ABSTRACT

Nowadays social websites have become a major trend in the Web 2.0 environment, enabling abundant social data available. In this paper, we explore the role of two types of social relationships: membership and friendship, while being fused with traditional CF (Collaborative Filtering) recommender methods in order to more accurately predict users' interests and produce recommendations to them. Through an exploratory evaluation with real-life dataset from Last.fm, we have revealed respective effects of the two explicit relationships and furthermore their combinative impacts. In addition, the fusion is conducted via random walk graph model in comparison with via weighted neighborhood similarity matrix, so as to identify the best performance platform. In-depth analysis on the experimental data particularly shows the significant improvement by up to 8% on recommendation accuracy, by embedding social relationships in CF via graph model.

Categories and Subject Descriptors

H.5.3 [**Group and Organization Interfaces**]: [Collaborative Filtering, Computer-supported cooperative work, Evaluation/methodology]; H.3.3 [**Information Storage and Retrieval**]: [Information Filtering]

General Terms

Algorithms, Experimentation, Human Factors

Keywords

Recommender Systems, Collaborative Filtering, Social Relationship, Random Walk

1. INTRODUCTION

In recent years, collaborative-filtering (CF) based recommender systems have been widely developed in order to effectively support users' decision-making process especially when they are confronted with overwhelming information (e.g. a large amount of product options that popularly appear in the current Web environment). There are two basic entities considered by the recommender: the user and the item. The user provides his rates on items (e.g. movies, music, books, etc.) that he has experienced, based on which the system can connect him with persons who have similar interests and then recommend to him items that are preferred by these like-minded neighbors. In some cases which don't have rating values available, the user's interaction with items can also be considered. That is, if we can only get the information that a user watched a movie, the "rating" is 1; otherwise it is 0. The recommendation method based on this binary rating matrix is also named as log-based CF [24]. The traditional approaches can be hence regarded as implicit ways to infer the social relationship between users. However, it inevitably brings the limitation when few users rated or viewed few items (i.e. the sparsity problem), to make it hard to infer such preference relationship. With the increasing emergence of social network services, many websites support online user communities, such as *Youtube*, *Last.fm*, *del.icio.us*, and e-commerce sites including *Amazon.com* and *eBay.com*. Community facilities are provided so that users can create and access to their community information and communicate with friends or members. For example, on Last.fm (a popular music recommender website), the user can establish friendship with others by "finding people" and/or join a group of users having similar music tastes (e.g. through "finding groups").

We term this kind of community relationship as explicit social relationship, since it is directly defined by users, rather than inferred by the system. As a matter of fact, two types

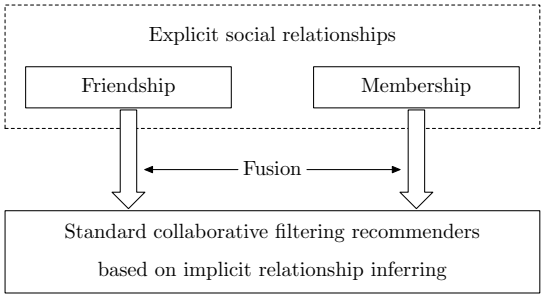

Figure 1: Fusion of Friendship and Membership into Standard CF

of explicit relationships are commonly available on the current websites (as in the example of Last.fm): *friendship* and *membership*. For instance, users A and B are friends given that A adds B in his friend list (or conversely), but it does not refer that they must have similar music tastes. On the other hand, if A and B join in the same group, it indicates that they have the membership relation and are likely with the similar interest (e.g. on "Beatles" provided it is the group's title).

It is believed that explicit social relationship can be likely applied to compensate the limitation of implicit relationship inference approach and improve the accuracy of recommender systems [1]. Some researchers have recently been engaged in fusing such kind of information into traditional CF methods [1, 15, 19]. However, to our knowledge, most works purely concentrate on friendship data [4, 16]. Their tentative studies unfortunately show that the fusion sometimes does not work well due to the friendship's inherent ambiguity as a relational descriptor [5]. Moreover, existing fusion approaches have been mainly based on the rating-matrix. It is in essence lack of exploration of other possible ways that may be potentially more effectively able to fuse the explicit social relationship with CF recommenders. As the social data is inherently in a graph structure, fusion via graph may be a good approach.

Given that the membership contains more information implying the user's preferences, such as his interest on the music genre or singer as indicated by the group's property and his like-minded people involved in the same group, we are interested in understanding whether this additional information could produce any practical benefits. Thus, in this paper, our objective is to study whether and how to best fuse both of membership and friendship, by means of comparing their respective effects and potential combinative impacts via different fusion platforms. We believe that our study will shed light on the role and applicability of social information in boosting collaborative intelligence of current recommender systems. More specifically, our contributions can be summarized as follows:

- We demonstrate the exact value of fusing explicit social relationships into recommender systems. Particularly, in some cases, the neighborhood of users learnt from membership has been found more accurate than from purely embracing friendship with traditional CF.
- We develop a framework to fuse and evaluate multiple types of social relationships in a systematical approach through weighted-similarity calculation. Moreover, we propose a novel graph model to fuse the social relationship with rating matrix, and adopt random walk algorithm to produce neighborhood similarities for recommendations. With a real-life dataset, we compared it with the weighted-similarity approach and identify the superior performance of graph fusion in improving recommendation accuracy.

In the next section, we first provide a brief review of related work, and then propose a systematical approach to study the use of explicit social relations and incorporate them in collaborative recommenders via graph model in addition to via weighted-similarity. Next, we show the experiment design and results analysis, followed by the final section of conclusion and future work.

2. RELATED WORK

We review the related literature of fusing heterogeneous data sources with rating matrix to improve standard CF from two perspectives: *Fusion of Social Relationship*, and *Graph-based Recommender Algorithm*.

2.1 Fusion of Social Relationship

Since popular user-based and item-based CF algorithms that only rely on user-item rating matrix always suffer from sparse and imbalance of rating data, researchers have started to incorporate other data sources to improve standard CF. Balabanovic et al. [6] were among those who first investigate *content-based* systems that make use of the descriptive data about an item. Melville et al. [7] enhanced CF by using content of a movie, e.g., movie genre. Pazzani [8] investigated hybrid methods using both of user data (demographic information) and item data (content) for improving recommendation accuracy.

Recently, with the increasing development of social websites and appearance of social data, researchers have begun to pay attention to the social data and explored its usage in recommender systems. Konstas [21] adopted Random Walk with Restart to model the friendship and social annotation (tagging) in a music track recommendation system. Golbeck [11] used trust relationship in social network to improve movie recommendations. [15] used social network data for neighborhood generation. In a Munich-based German community, friends are compared to neighbors of collaborative filtering for rating prediction. Their results showed that the social friendship can benefit the traditional recommender system. [19] proposed an online social recommender system attempting to use more social information for recommendation generation. The social data they introduced are the friendship of users (from GeekBuddy), which was used to refine the description of each user. [10] proposed a factor analysis approach based on probabilistic matrix factorization to solve the data sparsity and poor prediction accuracy problems, by employing both of users' social network information and rating records. This work also concentrated on using friendship to improve recommendations. However, it has been shown that online friendship sometimes does not work well due to its inherent ambiguity as a relational descriptor [4, 16]. Compared to online friendship, online community membership contains more information about users' preferences. [2] used membership for recommending online communities to members of the Orkut social network. However, their recommendations were on a per-community,

rather than on a per-user basis. I. Guy et al. [22] [23] built a people recommendation system named SONAR by leveraging data from multiple channels including membership in project wiki.

2.2 Graph-based Recommender Algorithm

The computation of user/item similarity plays a key role in user/item-based collaborative recommenders. Popular measurements of user similarity are Cosine similarity and Pearson's correlation coefficient (see [9] for examples) based on the user-item rating matrix. The limitation is that they only use the local pairwise user information for neighborhood searching.

Recent years, graph-based methods have been introduced to model relations between users and items from a global perspective, and been used to seamlessly incorporate heterogeneous data sources into the traditional user-item rating matrix. Huang proposed a two-level graph model for products [18], in which the two layers of nodes represent products and customers respectively, and three types of links between nodes are: the product-product, the user-user, and the user-product link. The recommendation is generated based on the association strengths between a customer and products.

Random walks on graph have been extensively discussed [12] and shown a rather good performance in the recommendation area. M. Gori and A. Pucci proposed a random-walk based scoring algorithm, ItemRank [14], which can be used to rank products according to expected user preferences, so as to recommend top ranked items to potentially interested users. Similarly, Baluja et al. [3] made video recommendations for *YouTube* through random walk on the view graph, which is a bipartite graph containing users and videos where links are visiting logs of users on videos. F. Fouss et al. [13] presented a new perspective on characterizing the similarity between elements of a database or, more generally, nodes of a weighted and undirected graph. This similarity called L^+, the pseudoinverse of the Laplacian matrix of the graph. Their experimental results on the MovieLens database showed that the Laplacian-based similarity computation performed well in comparison with other methods.

However, the limitation of related work in the "fusion of social relationship" (the first subsection) is that few have considered the potential positive role of membership. Moreover, in the related work on "graph-based recommender algorithm", no work on its performance as a fusion platform has been conducted. In this paper, we therefore aim at investigating how to effectively fuse both of friendship and membership into CF algorithm via random walk approach in order to improve the performance and solve the sparsity problem. To the best of our knowledge, our work is one of the first attempts to use both friendship and membership to enhance recommender systems.

3. FUSING SOCIAL RELATIONSHIPS INTO RECOMMENDERS

In this section, we mainly take into account of two types of explicit social relationships: friendship and membership, and propose a generic framework to fuse them with the user-item rating matrix. Given that user-user similarity computation is crucial to collaborative recommenders, a more accurate user-user similarity always leads to better recommendation results. Our fusion framework aims at leveraging the two social relationships to strengthen the user similarity calculation process by two means: one is combining the user-similarity from friendship and/or membership with similarity from rating matrix in a weighted approach; the other is modeling the social relations and rating matrix all in a graph, and then applying random walk on this graph to compute the user similarity.

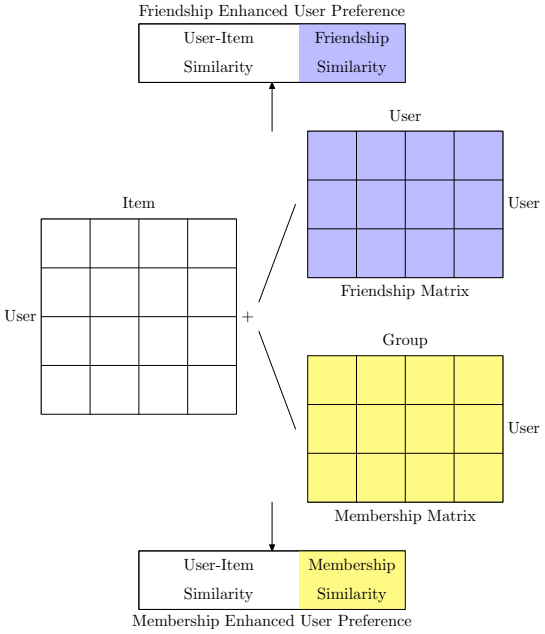

Figure 2: Fuse Friendship and Membership into Collaborative Recommender via Weighted-Similarity

3.1 Fusing via weighted-similarity

In the rating matrix, we can view the preferences of users as feature vectors. Every user vector consists of n feature slots, one for each available item. The values used to fill those slots can be either the rating r_{ak} that a user u_a provides to the corresponding item i_k, or 0 if no such rating exists. Now, we can compute the proximity between two users u_a and u_b, by calculating the similarity between their vectors. For example, we can use the Cosine similarity for this calculation as follows:

$$Similarity(u_a, u_b) = \frac{R_{u_a} \cdot R_{u_b}}{||R_{u_a}||||R_{u_b}||} \quad (1)$$

where R_{u_a} and R_{u_b} are two vectors of ratings from users u_a and u_b respectively.

According to the user similarity calculated from the rating matrix, we give a detailed description of fusing social relationships via weighted user similarity based on Fig 2.

When fusing friendship with user similarity from rating matrix, we need to get a user similarity based on the friendship firstly. We represent the friendship in the form of a user-user matrix. If two users u_i and u_j are friends, then the value of cell u_{ij} is set to 1, otherwise 0. Based on this user-user matrix, we calculate a friendship similarity by adopting Cosine Correlation, named Sim_{fri}. Next, when calculating the final user similarity between u_a and u_b, we combine the Sim_{fri} with Sim_{ui}(user similarity calculated from user-item

matrix) in a weighted approach as follows:

$$Sim_{ui+fri}(u_a, u_b) = \lambda Sim_{ui}(u_a, u_b) + (1-\lambda) Sim_{fri}(u_a, u_b) \quad (2)$$

The parameter λ is used to adjust the weight of Sim_{fri} and Sim_{ui}, the bigger the λ is, the rating matrix plays a more important role in the combined similarity. Finally, we use this combined similarity Sim_{ui+fri} in finding neighbors for each user.

When fusing the membership, firstly we also need to get a user-user similarity based on the membership data. Since membership is the relationship between the user and the communities/groups he/she joined, a new type of entity was introduced besides the two entities (user and item). We concretely represent the membership in the form of a user-group matrix, where the rows indicate users and the columns indicate the groups joined by users. If a user u_i joins group g_j, the value of cell u_{ij} is set to 1, otherwise 0. Based on this user-group matrix, we can get a membership similarity by using Cosine Correlation too, named Sim_{mem}. As for the generation of final user-user similarity, a weighted formula is applied where λ plays the same role as before.

$$Sim_{ui+mem}(u_a, u_b) = \lambda Sim_{ui}(u_a, u_b) + (1-\lambda) Sim_{mem}(u_a, u_b) \quad (3)$$

Furthermore, we are interested in seeing what will happen if two types of social relations are fused together with the rating matrix. In this condition, we first calculate the user-user similarity Sim_{fri} from friendship and Sim_{mem} from membership independently, and then introduce two parameters: λ and β to adjust the weights of three data sources as shown in the equation 4.

$$Sim_{ui+fri+mem}(u_a, u_b) = \lambda Sim_{ui}(u_a, u_b) + (1-\lambda)(\beta Sim_{mem}(u_a, u_b) + (1-\beta) Sim_{fri}(u_a, u_b)) \quad (4)$$

At the first level, λ is used to adjust the weight between rating matrix and the other two social relationships; and then β is used to adjust the remaining weight between friendship and membership. The bigger the λ is, the rating matrix plays a more important role; the bigger the β is, the membership plays a more dominant role in the combined user-user similarity.

After the computation of user-user similarity for finding neighbors, the next step is to recommend items to users by predicting each item's ratings. The predicted rating $r_{i,m}$ of a test item m for the user i is hence computed as:

$$r_{i,m} = \frac{\sum_{j=1}^{N} sim(u_i, u_j) * r_{j,m}}{\sum_{j=1}^{N} sim(u_i, u_j)} \quad (5)$$

where $r_{j,m}$ is the rating of user u_j on the item i_m, and $sim(u_i, u_j)$ is the similarity between the current user u_i and the neighbor u_j. In fusing via weighted-similarity approach, $sim(u_i, u_j)$ can be similarity measures in equation 2, 3, and 4, depends on the fusing strategy and data used each time.

3.2 Fusing via Graph

In the "fusing via weighted-similarity" method, we calculated the user-user similarity based on the static "local" pairwise user information. As we know, social network is inherently in a graph structure with the transitivity characteristics as a key feature of social relations, therefore we have been motivated to further use graph-based random walk algorithm for modeling these social data (i.e. friendship and membership). We think that the transitivity of the graph will improve the computation of the similarity between two users.

3.2.1 Graph Construction for Social Community

The first key issue is to construct a meaningful graph so that the resulting similarity can truly reflects the preference similarity between users. Considering a graph $G = (V, E, W)$, where V is the set of nodes (users, items or communities, etc.), E is the set of edges which represent relationships between all types of nodes, and W is the set of weights for all edges.

The relationship data in a social community can be interactive relationship, such that user watched a movie or listened to a song; or be social relationship like membership or friendship. Let us use Last.fm which contains all types of these data for example. It can be modeled by a graph G in the following way: there are three types of nodes in Last.fm, the user, artist (item) and group, and each element of the user, artist and group corresponds to a node of the graph; and the interactive relationship like a user *listens_to* an artist, social relationship like user is a *member_of* a group, and user is a *friend_of* another user is expressed as an edge. When a random walker walks on the graph, the difference of node types were ignored, and we only care about how many short paths existed between two nodes which have directly impact on their similarity computation, so special treatment is not needed for any type of nodes in our graph.

The weight w_{ij} of the edge connecting node i and node j should have a meaningful value. Traditionally, we usually deal with interactive data in the following convention: the more frequent the interaction between node i and node j, the larger the value of w_{ij}, and consequently, the easier the communication through the edge. However, in the case of social community, besides interactive data we also have social relationship, and they usually are not associated with a frequent number, e.g. most users join a group only once, and so is the same for adding friends. For the consistency of the two types of edges and simplicity, we set the weight of *member_of* edge and *friend_of* edge to be 1, and treat *listens_to* edge as follows: if a user listened to the songs of an artist more than 3 times, then the edge connecting the user and the artist was weighted 1. We require the weights to be both positive $w_{ij} > 0$ and symmetric $w_{ij} = w_{ji}$, so the graph we built is an undirected graph.

When handling friendship between users, there are several approaches to transforming the pairwise similarity between nodes of the same type (e.g. users) into a graph [17], such as the ϵ-neighborhood graph, k-nearest neighbor graph, and the fully connected graph. We choose the ϵ-neighborhood graph to fuse the friendship on the graph, not only because it has computational advantage by using a sparse representation of the data, but also because it can filter out the noisy data so that we can create a concrete friend set for each user. When constructing the ϵ-neighborhood graph, we connect user-node pairs whose friendship similarities are greater than ϵ. Given that the range of friendship similarity is [0,1], the range of ϵ is also [0,1]. We enumerate ϵ in this range with step 0.01, and finally we get the optimal ϵ.

When the graph was built, for the corresponding symmetric adjacency matrix A of graph G, the element a_{ij} was

defined as: $a_{ij} = w_{ij}$ if node i is connected to node j and a_{ij} = 0 otherwise. Thus, people who listen to the same artists and join same groups, will be connected by a comparatively larger number of short paths.

3.2.2 Random Walk and Similarity Measures

Random walk is a mathematical formalization of a trajectory that consists of taking successive random steps. At each step, the next node in the walk is selected randomly from the neighbors of the last node in the walk. The sequence of visited nodes is a Markov Chain [20], with the transition probability:

$$p_{ij} = \begin{cases} \frac{1}{d(i)}, \text{ if } (i,j) \in E \\ 0 \quad , \text{otherwise} \end{cases} \tag{6}$$

where $d(i)$ is the degree of node i.

From the viewpoint of collaborative recommender systems, finding accurate neighbor set for each user is the cornerstone, so a good similarity measure on the graph is a crucial step.

Fouss et al. [13] has discussed several approaches to computing similarities between nodes of graph and their application to collaborative recommendations, that mainly included distance-based measure like ECTD, and inner-product based measure like L^+.

ECTD is the abbreviation for Euclidean Commute-Time Distance. Average commute-time is the average number of steps that a random walker, starting in node i ($i \neq j$), will take to enter node j for the first time and go back to i, represented as $n(i,j)$. Average commute-time is symmetric by definition, and it is a distance measure. $n(i,j)^{1/2}$ is also a distance in Euclidean space, and it is named as Euclidean Commute-Time Distance.

L^+ is the *Moore-Penrose pseudoinverse* of the Laplacian matrix L. The Laplacian matrix $\mathbf{L}$ of the graph is defined as, $\mathbf{L} = \mathbf{D}$ - $\mathbf{A}$, where $D = Diag(a_{i.})$, with $d_{ii} = [D]_{ii} = a_{i.} = \sum_{j=1}^{n} a_{ij}$, and A is the adjacent matrix of graph G. Let e be a column vector with 1(i.e., $e = [1, 1, \ldots, 1]^T$, where T denotes the matrix transpose), then L^+ can be computed with the formula:

$$L^+ = (L - ee^T/n)^{-1} + ee^T/n, \tag{7}$$

where n is the number of nodes. If we define e_i as the ith column of $\mathbf{I}$, $e_i = [0, \ldots, 0, 1, 0, \ldots, 0]^T$(1 is in the ith column), then we can explain the relation between average commute-time and L^+ in the form

$$n(i,j) = V_G(e_i - e_j)^T L^+ (e_i - e_j), \tag{8}$$

where each node i is represented by a unit vector e_i in the node space spanned by $\{e_i\}$. It can be proved that the node vector e_i can be transformed into a new Euclidean space, and the elements of $L^+(l_{ij})$ are the inner products between these transformed node vectors. Therefore L^+ is a kernel matrix(a Gram matrix) and can be used as a similarity matrix for the nodes.

According to Fouss' study, the inner-product based similarity measure L^+ provides better and more stable results for collaborative recommenders, so we adopt L^+ metric to measure the similarity between users, which means in equation 5, $sim(u_i, u_j)$ equals to l_{ij} in this case.

4. EXPERIMENTS

4.1 Data Sets

Traditional data sets used in the evaluation of collaborative filtering systems, such as MovieLens, do not include explicit social relationship, while in *Last.fm*, a popular social music site, community information is available so that an entity-relation model can be generated which includes the relationship between users and items.

For our purpose, we extracted two typical social relationships: the friendship between users and the membership which describes the user's participation in groups, by accessing its Web Service APIs [1]. Besides, we think it is more meaningful to recommend artists instead of individual music since music is variable while preference on artists is more constant, so we use artist as the "item" in our recommendations. A user and an item is linked if the user listened to song(s) of the artist, and a user and a community is linked if the user joined the group. The relationship between an artist and a community is formed if the artist's songs were frequently listened by users in this group. There are also links among users describing their friendship.

We concretely established an active data set consisting of 943 users, 1,001 artists and 676 groups. There are 36,424 records in the user-artist matrix which sparsity degree (percentage of zero values in the matrix) is 96.14%, and 7,038 records in the user-group matrix which sparsity is 98.89%. The total number of friendship of 943 selected users is 33776, which means each user on average has 35.8 friends. Please note that the rating matrix here is the user-artist matrix, and if a user listened to song(s) of an artist, there is "1" in the corresponding cell.

By means of 5 fold cross-validation [2], each row (represent a user) of the user-artist matrix is randomly split into five different sets. For each time of experiment, four-fifths of of the data is included in the training set and the other is used as the testing data.

4.2 Evaluation Metrics

We adopted standard metrics in the area of information retrieval to evaluate our recommenders. During each round of cross-validation, we recommend and rank a set of potential artists for each user. We then compare the predicted recommendation list with true preferences on artists in the test set, and compute precision, recall, and F-measure scores.

1. **Recall.** The score measures the average (on all users) of the proportion (in percentages) of artists from the testing sets that appear among the top n ranked list from the training sets, for some given n. It should be as high as possible for good performance. We computed the recall from Top-1 to Top-20 artists(for a total of 1001 artists).

2. **Precision.** This metric measures the proportion of recommended items that are ground truth items. Note that the items in the profiles of the testing data represent only a fraction of the items that the user truly accessed.

[1]The web page is `http://www.last.fm/api`

[2]The number of fold is the number of tested sets.

3. **F-measure.** F-measure is the weighted harmonic mean of precision and recall. The equation is as follows:

$$F = \frac{2 * precision * recall}{(precision + recall)} \quad (9)$$

In the following, we report the results of recommending the top 1, 2, 5, 10 and 20 artists. At each pass, 50 users are taken as neighbors based on different similarity measures for recommendation.

4.3 Experiment Design and Results

At first, we evaluated the fusion approach via weighted-similarity. A user-based collaborative filtering recommender was first run on the user-artist rating matrix, resulting in the baseline represented by CF_{UI} (see Tables 1 to 3 respectively showing precision, recall and F-measure scores).

Table 1: Precision of Fusion via Weighted-Similarity Approaches

Precision	Top 1	Top 2	Top 5	Top 10	Top 20
CF_{UI}	**29.93**	25.12	19.09	15.23	11.33
WS_{fri+UI}	29.67	25.33	**19.64**	**15.51**	**11.48**
WS_{mem+UI}	29.44	**25.49**	19.58	15.33	11.40
$WS_{fri+mem+UI}$	29.37	25.45	19.58	15.34	11.42

Table 2: Recall of Fusion via Weighted-Similarity Approaches

Recall	Top 1	Top 2	Top 5	Top 10	Top 20
CF_{UI}	**3.87**	6.50	12.36	19.72	29.34
WS_{fri+UI}	3.84	6.56	**12.71**	**20.07**	**29.73**
WS_{mem+UI}	3.81	**6.60**	12.67	19.84	29.52
$WS_{fri+mem+UI}$	3.80	6.59	12.68	19.86	29.58

Table 3: F-measure of Fusion via Weighted-Similarity Approaches

F-measure	Top 1	Top 2	Top 5	Top 10	Top 20
CF_{UI}	**6.85**	10.33	15.01	17.19	16.35
WS_{fri+UI}	6.80	10.42	**15.43**	**17.50**	**16.56**
WS_{mem+UI}	6.75	**10.49**	15.38	17.29	16.45
$WS_{fri+mem+UI}$	6.73	10.47	15.39	17.31	16.48

Then, we tried to fuse friendship with the rating matrix and used λ to adjust the weight of rating matrix and friendship while computing user similarity. Figure 3 shows how F-measure changes with the changing of λ. It achieves the peak when $\lambda = 0.7$, which means that the rating matrix contributes to 70% percent of the weight while friendship contributes to 30% in calculating the user-user similarity. Specifically, results in the second rows of Tables 1 to 3, represented by WS_{fri+UI}, give the precision, recall and F-measure scores when $\lambda = 0.7$. It can be seen that it is better than the baseline, especially when returning top 5 recommendations (the improvement of F-measure achieved up to 2.79%). In the process of tuning λ and β in the following, we considered the average score of Top-1 to Top-20 recommendations.

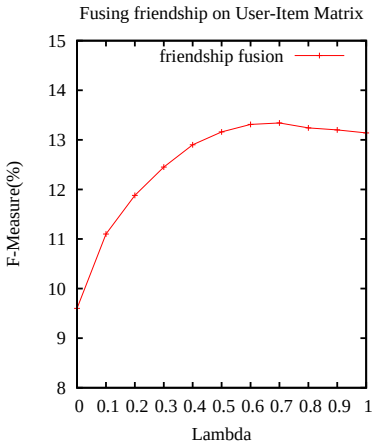

Figure 3: F-measure changes as lambda change in friendship fusion

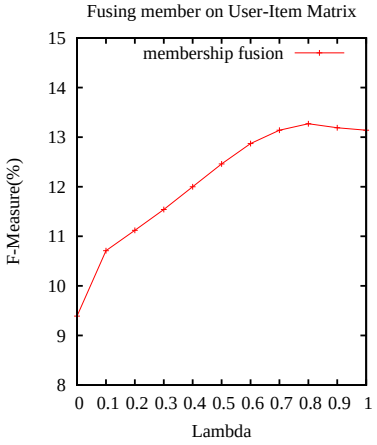

Figure 4: F-measure changes as lambda change in membership fusion

Next, we fused membership with rating matrix and also used the parameter λ to adjust the weight of rating matrix and membership in the similarity calculation.

From the figure 4, it can be seen that when $\lambda = 0.8$, we can get best results overall. The third rows of Tables 1 to 3, represented by WS_{mem+UI}, respectively show precision, recall and F-measure values when $\lambda = 0.8$. It shows that the results slightly improve on the baseline, but are a little weaker than the fusion of friendship on the rating matrix.

We further fused two relationships together on the rating matrix and adopted two parameters: λ and β to adjust the weights for the three data sources as shown in Formula 4. Experimental results indicate that the hybrid fusion performs best when lambda = 0.8 and beta = 0.5 (which means rating matrix contributes 80%, friendship and membership respectively contributes 10% and 20% in the user-user similarity calculation). These results are illustrated in the last rows of Tables 1 to 3. To our surprise, compared to fusion of friendship and membership separately, it did not have dis-

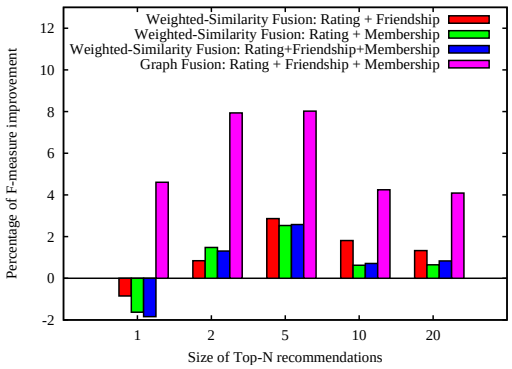

Figure 5: Improvements on F-Measure of All the Fusion Approaches

tinct difference and was actually even a little weaker than the pure fusing of friendship.

In order to identify whether the fusion via graph model would perform better than via weighted similarity approach given that social data are inherently in the graph structure, we finally did the experiment of fusing the two social relationships on the graph and applied random walk algorithm to calculate neighborhood similarity.

Based on the graph construction method described before, we firstly run a series of simulations to learn the optimal ϵ. After running 100 times, the optimal ϵ that we got is 0.05. The comparative results under threshold 0.05 are then computed and listed in Table 5. It shows that precision, recall and F-measure scores are all highly improved compared to the fusion via weighted-similarity approach. In particular, when returning top 2 and top 5 recommendations, the improvements significantly reached 7.94% and 7.99% respectively.

Table 4: Fusing friendship and membership via Graph

$G_{fri+mem+UI}$	Top 1	Top 2	Top 5	Top 10	Top 20
Precision	31.30	27.12	20.62	15.88	11.79
Recall	4.05	7.02	13.35	20.55	30.54
F-Measure	7.18	11.15	16.21	17.92	17.01

Figure 5 further shows that while returning Top-1 recommendation, the fusion via graph can achieve an improvement at 4.82%, and others are however all slightly weaker than the baseline. When returning top N recommendations from 2 to 10, all of the fusion approaches enhance the baseline CF method, which positively proves the usefulness of social relationship data especially when multiple recommendations are computed. It also indicates that the fusion via graph can boost the baseline significantly by up to 8%, demonstrating that the graph model is a more proper way to fuse explicit social relationships.

5. CONCLUSION AND FUTURE WORK

In this paper, we presented two principal methods to integrate explicit social relationships into traditional CF methods: the weighted-similarity fusion and the graph fusion. We demonstrated the effectiveness of social relationships in augmenting recommendations, and particularly that the graph-based fusion is more effective in bringing into play of the power of social data. To the best of our knowledge, the work is one of the first attempts to explore the effect of membership in addition to friendship, and to fuse both of them based on random walk graph model with collaborative filtering (CF) systems.

Table 5: Table of improvements on F-Measure by Comparing with the Baseline

Improvements	Top 1	Top 2	Top 5	Top 10	Top 20
WS_{fri+UI}	-0.73	0.87	2.80	1.80	1.28
WS_{mem+UI}	-1.46	1.55	2.47	0.58	0.61
$WS_{fri+mem+UI}$	-1.75	1.36	2.53	0.70	0.80
$G_{fri+mem+UI}$	4.82	7.94	7.99	4.25	4.04

For the next step, we are interested in further exploring the impact of social relationships on recommender systems from three aspects: one is to explore other potential relationships, such as the relation between items and associated groups, other social relationships besides friendship and membership, such as the reporting chain in a company, to see how to model and utilize these data in order to make better recommendations; another direction is to explore how to enhance Random Walk model so as to handle heterogeneous data in a more fine-grained way, based on the method proposed in [25]; finally, as the explosion of the size of social websites, we need to pay more attention to the algorithm's scalability and efficiency, when the social graph grows with millions of nodes.

6. ACKNOWLEDGMENTS

We thank Lawrence Bergman of IBM T.J. Watson Research Center for his generous help and valuable comments on this work.

7. REFERENCES

[1] C. Lam. Snack: incorporating social network information in automated collaborative filtering. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 254–255, New York, NY, USA, 2004. ACM.

[2] E. Spertus, M. Sahami, and O. Buyukkokten. Evaluating similarity measures: a large-scale study in the orkut social network. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 678–684, New York, NY, USA, 2005. ACM.

[3] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly. Video suggestion and discovery for youtube: taking random walks through the view graph. In J. Huai, R. Chen, H.-W. Hon, Y. Liu, W.-Y. Ma, A. Tomkins, and X. Zhang, editors, *WWW*, pages 895–904. ACM, 2008.

[4] D. Boyd. Friends, friendsters, and myspace top 8: Writing community into being on social network sites. http://www.firstmonday.org/issues/issue11_12/boyd/index.html, December 2006.

[5] N. Baym. How Good A Friend Is A Last.fm Friend? http://www.last.fm/user/popgurl/journal/2008/04/28/3d9l_how_good_a_friend_is_a_last.fm_friend.

[6] M. Balabanović and Y. Shoham. Fab: content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, 1997.

[7] P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Eighteenth national conference on Artificial intelligence*, pages 187–192, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[8] M. J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artif. Intell. Rev.*, 13(5-6):393–408, 1999.

[9] J. S. Breese, D. Heckerman, and C. M. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In G. F. Cooper and S. Moral, editors, *Proceedings of the 14^{th} Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[10] H. Ma, H. Yang, M. R. Lyu, and I. King. Sorec: social recommendation using probabilistic matrix factorization. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 931–940, New York, NY, USA, 2008. ACM.

[11] J. Golbeck. Generating predictive movie recommendations from trust in social networks. pages 93–104. 2006.

[12] P. G. Doyle and J. L. Snell. Random walks and electric networks, 2000.

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

[14] M. Gori and A. Pucci. Itemrank: A random-walk based scoring algorithm for recommender engines. In M. M. Veloso, editor, *IJCAI*, pages 2766–2771, 2007.

[15] G. Groh and C. Ehmig. Recommendations in taste related domains: collaborative filtering vs. social filtering. In *GROUP '07: Proceedings of the 2007 international ACM conference on Supporting group work*, pages 127–136, New York, NY, USA, 2007. ACM.

[16] R. Gross, A. Acquisti, and J. H. Heinz. Information revelation and privacy in online social networks. In *WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 71–80, New York, NY, USA, 2005. ACM Press.

[17] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, December 2007.

[18] Z. Huang. *Graph-based analysis for e-commerce recommendation*. PhD thesis, Tucson, AZ, USA, 2005. Adviser-Hsinchun Chen and Adviser-Daniel D. Zeng.

[19] H. G. Hummel, B. van den Berg, A. J. Berlanga, H. Drachsler, J. Janssen, R. Nadolski, and R. Koper. Combining social-based and information-based approaches for personalised recommendation on sequencing learning activities. *International Journal of Learning Technology*, 3:152–168(17), 12 August 2007.

[20] J. G. Kemeny and J. L. Snell. *Finite Markov Chains*. Springer-Verlag, 1976.

[21] I. Konstas, V. Stathopoulos, and J. M. Jose. On social networks and collaborative recommendation. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 195–202, New York, NY, USA, 2009. ACM.

[22] I. Guy, I. Ronen, and E. Wilcox. Do you know?: recommending people to invite into your social network. In *IUI '09: Proceedings of the 13th international conference on Intelligent user interfaces*, pages 77–86, New York, NY, USA, 2009. ACM.

[23] J. Chen, W. Geyer, C. Dugan, M. Muller, and I. Guy. Make new friends, but keep the old: recommending people on social networking sites. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 201–210, New York, NY, USA, 2009. ACM.

[24] J. Wang, A. P. de Vries, and M. J. T. Reinders. A user-item relevance model for log-based collaborative filtering. In M. Lalmas, A. MacFarlane, S. M. Rüger, A. Tombros, T. Tsikrika, and A. Yavlinsky, editors, *ECIR*, volume 3936 of *Lecture Notes in Computer Science*, pages 37–48. Springer, 2006.

[25] J. Zhang, J. Tang, B. Liang, Z. Yang, S. Wang, J. Zuo, and J. Li. Recommendation over a heterogeneous social network. *Web-Age Information Management, International Conference on*, 0:309–316, 2008.