TrustWalker: A Random Walk Model for Combining Trust-based and Item-based Recommendation

Mohsen Jamali
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada
mohsen_jamali@cs.sfu.ca

Martin Ester
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada
ester@cs.sfu.ca

ABSTRACT

Collaborative filtering is the most popular approach to build recommender systems and has been successfully employed in many applications. However, it cannot make recommendations for so-called cold start users that have rated only a very small number of items. In addition, these methods do not know how confident they are in their recommendations. Trust-based recommendation methods assume the additional knowledge of a trust network among users and can better deal with cold start users, since users only need to be simply connected to the trust network. On the other hand, the sparsity of the user item ratings forces the trust-based approach to consider ratings of indirect neighbors that are only weakly trusted, which may decrease its precision. In order to find a good trade-off, we propose a random walk model combining the trust-based and the collaborative filtering approach for recommendation. The random walk model allows us to define and to measure the confidence of a recommendation. We performed an evaluation on the Epinions dataset and compared our model with existing trust-based and collaborative filtering methods.

Categories and Subject Descriptors

H.2.8.d [**Information Technology and Systems**]: Database Applications - Data Mining

General Terms

Algorithms, Design, Experimentation, Measurement

Keywords

Trust, Recommendation, Random Walk

1. INTRODUCTION

With the rapidly growing amount of information available on the WWW, it becomes necessary to have tools to help users to select the relevant part of online information. To satisfy this need, recommender systems have emerged, e.g. there are popular recommenders for movies[1], books[2], music[3], etc.

Typically in a recommender system, we have a set of *users* and a set of *items*. Each user u rates a set of items by some values. The recommender has the task to predict the rating for user u on a non-rated item i or to generally recommend some items for the given user u based on the ratings that already exist. Collaborative Filtering [6] methods make recommendations based on the ratings of item i by a set of users whose rating profiles are most similar to that of user u. With the advent of online social networks, the trust-based approach to recommendation has emerged. This approach assumes a trust network among users and makes recommendations based on the ratings of the users that are directly or indirectly trusted by u.

Collaborative Filtering is most effective when users have expressed enough ratings to have common ratings with other users, but it performs poorly for so-called cold start user. Cold start users are new users who have expressed only a few ratings. Using similarity based approaches, it is unlikely to find similar users since the cold start users only have a few ratings. Trust-based recommenders, however, can make recommendations as long as a new user is connected to a large enough component of the trust network.

Using a trust network therefore improves the coverage of recommendations. However, when we go far from the source user u in the trust network, the trust between these users and the source user will become fairly weak and their ratings will be noisy and unreliable. Therefore, we have to use the ratings expressed by users in the neighborhood close to the user u. But, in this case the probability of finding a rating expressed on the item will be very low and we will not be able to compute a prediction.

In order to consider enough ratings without suffering from noisy data, we propose a random walk method (TrustWalker) which combines trust-based and item-based recommendation. TrustWalker considers not only ratings of the target item, but also those of similar items. The probability of using the rating of a similar item instead of a rating for the target item increases with increasing length of the walk. Our framework contains both trust-based and item-based collaborative filtering recommendations as special cases. Most traditional recommender systems do not provide confidence in their predictions. The random walk model allows us to com-

[1]http://www.netflix.com
[2]http://www.amazon.com
[3]http://www.last.fm

KDD'09, June 28–July 1, 2009, Paris, France.

pute the confidence in our predictions. Also our system is able to explain and justify its results.

The rest of the paper is organized as follows: Section 2 describes the problem definition. Related works are discussed in section 3. We discuss the details of our proposed model in section 4. In section 5 we introduce some desirable properties of our model. The experimental results and comparison with existing methods are discussed in section 6. Finally we conclude the paper in section 7 and introduce some directions for future research.

2. PROBLEM DEFINITION

In recommender systems we have a set of users $U = \{u_1, ... u_N\}$ and a set of items $I = \{i_1, ... i_M\}$. Each user u rates a set of items $RI_u = \{i_{u_1}, ... i_{u_k}\}$. The rating of user u on item i is denoted by $r_{u,i}$. $r_{u,i}$ can be any real number, but often ratings are integers in the range $[1, 5]$. In a trust-based system, we also have a trust network among users. If u trusts v, then $t_{u,v}$ denotes the value of this trust as a real number in $[0, 1]$. Zero means no trust and one means full trust. Binary trust networks are the most common trust networks (Amazon[4], eBay[5], ...). We define $TU_u = \{v \in U |\ t_{u,v} = 1\}$ where TU_u denotes the set of users directly trusted by u.

The trust network can now be defined as a graph $G =< U, TU >$ where $TU = \{(u, v)|\ u \in U,\ v \in TU_u\}$. There is a node corresponding to each user, and an edge corresponding to each trust statement.

The task of a recommender is as follows: Given a user $u \in U$ and an item $i \in I$ for which $r_{u,i}$ is unknown, predict the rating for u on item i. We call u the source user and i the target item. The predicted rating is denoted by $\widehat{r}_{u,i}$. Normally, users rate only a very small percentage of the items, and $r_{u,i}$ is unknown for most pairs of (u,i).

Traditional recommenders[6] estimate $\widehat{r}_{u,i}$ based on ratings expressed by similar users. Basically, they find a neighborhood of *raters* (users who have rated target item i) who have a rating profile similar to the source user, and aggregate their ratings. In trust-based recommenders, the trust network will be used instead of rating similarities to define the neighborhood. To predict a rating we ask our directly trusted neighbors whether they know the rating for the item. If so, they return it, otherwise they recursively ask their direct neighbors. The neighborhood in a trust based recommendation is defined as the set of raters trusted (directly or indirectly) by the source user. The ratings from these rates are aggregated to produce a recommendation.

Trust-based recommendation works because of the effects of selection and social influence that have been postulated by sociologists for a long time. Selection means that people tend to relate to people with similar attributes, and due to social influence related people in a social network influence each other to become more similar[16]. The increasing availability of online social network data has finally allowed a verification of these sociological models. [4] experimentally verified that people are similar to their neighbors in a social network for these reasons. They had a network of people having social interactions and a similarity network in which users are connected to their most similar users. It was shown that the social interaction and similarity graphs have little overlap, sharing fewer than 15% of their edges. The results of [4] and of similar work confirm that a social network provides an independent source of information which can be exploited to improve the quality of recommendations.

Exploiting the trust network in recommenders does not necessarily enhance the precision of system, but it allows to compute the recommendation for more pairs of (u, i) which leads to better coverage. Coverage is the percentage of $< user, item >$ pairs in the test set for which we can compute a recommendation. This information can in particular help to produce recommendations for cold start users. Moreover, using a trust network will protect the recommender system against attacks like fake profiles. As faked profiles are not being trusted, they can not affect the recommender.

Figure 1 illustrates the trust-based recommendation problem. As shown in this figure, we have a social (trust) network of users. Each user has rated some items. The source user u wants a prediction on the target item i (the green diamond). Normally just a few users have rated the target item. Other users have ratings on other items which may include items already rated by u or items similar to the target item i. The goal of the recommender is to predict unknown ratings based on the ratings expressed by trusted friends.

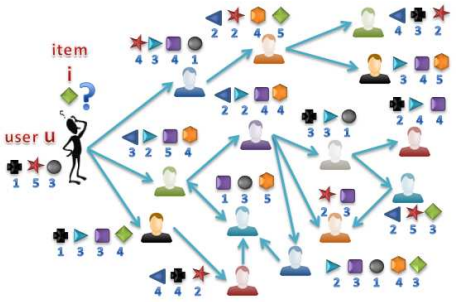

Figure 1: Illustration of a social network including ratings expressed by users. The ratings are shown below the item icon beside the user.

3. RELATED WORKS

In this section, we review related work on trust-based recommendations. We can distinguish two types of trust: Explicit trust and Implicit trust. Explicit trust denotes the trust values explicitly indicated by users, while implicit trust is the trust value inferred from some evidence such as feature similarity of users or email exchange among two persons. In this paper we deal with the explicit trust indicated by users. In the case of explicit trust, we have direct trust and indirect trust. Direct trust is the trust value explicitly indicated by users, but indirect trust is the trust inferred from direct trust using transitivity of trust. how to compute indirect trust is one of the main issues of trust models. Two different approaches can be distinguished for trust computation: Model-based[13][9] and Memory-based[10][5][18][8]. In Model-based approaches, a model with its parameters will be learnt to compute indirect trust. But, in Memory-based approaches, no model is being learnt and normally exploration and heuristics are being used.

TidalTrust[5] performs a modified breadth first search in the trust network to compute a prediction. Basically, it finds all raters with the shortest path distance from the source user and aggregates their ratings weighted by the trust be-

[4]www.amazon.com
[5]www.ebay.com

tween the source user and these raters. To compute the trust value between user u and v who are not directly connected, TidalTrust aggregates the trust value between u's direct neighbors and v weighted by the direct trust values of u and its direct neighbors. Since TidalTrust only uses information from raters at the nearest distance, it may loose a lot of valuable ratings from users a little further apart in the network.

[10] introduces MoleTrust. The ideas used in MoleTrust and TidalTrust are similar. But MoleTrust considers all raters up to a *maximum-depth* given as an input. *maximum-depth* is independent of any specific user and item. Also, to compute the trust value between u and v in MoleTrust, we perform a backward exploration. It means that the trust value from u to v is the aggregation of trust values between u and users directly trusting v weighted by the direct trust values.

The Advogato[8] maximum flow trust metric has been proposed in order to discover which users are trusted by members of an online community. The input for Advogato is given by an integer number n, the number of members to trust. To assign capacities to the edges of the network, they need to transform the network, so it needs to know the whole structure of the network. Moreover, it only computes the nodes to trust and does not compute different degrees of trust. Since the number of users to trust is independent of users and items and there is no distinction between the trusted users, this approach is not appropriate for trust-based recommendation.

AppleSeed has been proposed in [18]. The basic intuition of AppleSeed is motivated by spreading activation models. Source node u is activated through an injection of energy e, which is then fully propagated to other nodes along edges proportional to the weight of the edge. AppleSeed considers the trust to be additive. If there are many weakly trusted paths between two users, this pair of users will obtain a high trust value, which is not intuitive.

[1] presents a set of axioms for trust-based recommender systems and analyzes which combination of these axioms can be satisfied simultaneously. In the context of this discussion, a simple random walk method for binary (+1,-1) ratings is presented, which can be formulated as a special case of our model. [1] does not perform any experimental evaluation or comparison to other methods.

Trust has been defined and used in a different way by [12]. They extract the social network from the similarity of users' profiles, which is not providing additional information as is provided by trust network. They have two definitions for trust: profile level trust, which is a global reputation metric; and item level trust, which measures the trustworthiness of a user according to his recommendations for an item rather than a user-user local trust metric. They use the trust values to filter raters, and they aggregate the ratings weighted by a combination of trust and similarity values. Moreover, they do not use the transitivity of trust, but only directly trusted users. This method is not a trust-based recommendation method in the sense in which we use this term in this paper.

4. TRUSTWALKER MODEL

The main challenge in trust-based recommendation is to decide how far to go in exploring the network. There is a trade-off between precision and coverage: the further you go, the more likely to find *raters*, but the less trust-worthy their ratings become. Our approach to find a good trade-off is based on the following observation. Ratings expressed by strongly trusted friends on similar items are more reliable than ratings expressed by weakly trusted far neighbors on the exact target item. This motivates us to combine the trust-based and item-based approach.

We propose a random walk model, called TrustWalker, which considers not only ratings of the target item, but also those of similar items. The probability of using the rating of a similar item instead of a rating for the target item increases with increasing length of the walk. Basically, our model consists of two major components: the random walk on the trust network and the probabilistic item selection. The random walk performs the search in the trust network, and the item selection part considers ratings on similar items to avoid going too deep in the network. So our model improves the precision by preferring raters at a nearer distance and improves the coverage by considering similar items as well as the exact target item.

To recommend a rating for a source user u_0 on target item i, we perform random walks on the trust network, each starting at u_0 to find a user having expressed rating for i or items similar to i. The details of the random walks will be discussed later in this section. Each random walk returns a rating. We perform several random walks, and the aggregation of all ratings returned by different random walks are considered as the predicted rating $\widehat{r}_{u_0,i}$.

In the following subsections, we will discuss the details of our random walk model. In our notations, we use symbols $u, v, w, ...$ for users, $i, j, ...$ for items, and k for the step of a random walk. Table 1 lists all notations used in our model.

Notation	Description
$\phi_{u,i,k}$	probability of stopping the random walk at node u in step k.
$X_{u,i,k}$	RV for being at node v in k steps starting from u
$X_{u,i}$	RV for being at node v at some steps starting from node u
S_u	RV for selecting a user v out of members of set TU_u.
$Y_{u,i}$	RV for selecting item j amongst items rated by u
$XY_{u,i}$	RV for stopping at node v and selecting item j rated by v, while starting from u.
$r_{u,i}$	The rating expressed by u on item i
$\hat{r}_{u,i}$	The predicted rating of user u on i
$t_{u,v}$	The trust value among users u and v

Table 1: Notations used in TrustWalker. RV stands for Random Variable. All the notations have index i denoting target item i.

4.1 A Single Random Walk

Starting from source user u_0, we perform our random walk. At each step k of a random walk, we are at a certain node u. If u already has the rating on target item i, then we stop our random walk and return $r_{u,i}$ as the result of random walk. If u does not have a rating on i, then we have two options:

- With probability $\phi_{u,i,k}$, we don't continue the random walk. We stay at node u and randomly select one of the items (j) similar to i rated by u and return $r_{u,j}$.
- With probability $1 - \phi_{u,i,k}$, we continue our random

walk and walk to another user v who is one of u's direct trusted neighbors ($v \in TU_u$).

If we decide to continue the random walk at node u, we have to select one of directly trusted neighbors of u to continue the random walk to that user. We define S_u as the random variable for selecting a user v from TU_u:

$$P(S_u = v) = \frac{t_{u,v}}{\sum_{w \in TU_u} t_{u,w}} = \frac{1}{|TU_u|} \quad (1)$$

Now, we have:

$$P(X_{u_0,i,k+1} = v | X_{u_0,i,k} = u, \widetilde{R_{u,i}}) = (1 - \phi_{u,i,k}) \times P(S_u = v)) = (1 - \phi_{u,i}) \times \frac{1}{|TU_u|} \quad (2)$$

Here, $X_{u_0,i,k}$ denotes the random variable for being at node v in step k while looking for a prediction on target item i for source user u_0. Details of computing $P(X_{u,i,k} = v)$ are discussed later. Also we have a condition that the user u in step k-1 does not have the rating for item i (denoted by $\widetilde{R_{u,i}}$). The probability of walking from user u to v is independent of previous steps. But, since $\phi_{u,i,k}$ depends on the step k, it is not independent of the step of the random walk.

With probability $\phi_{u,i,k}$, if we decide to stay at a user u, we select one of the items rated by u which is similar to the target item i. The idea is that we define a similarity measure between items, and for each item $j \in RI_u$, we assign a probability of selecting proportional to the similarity of i and$j$. We'll discuss the details of the similarity metric later.

$$P(Y_{u,i} = j) = \frac{sim(i,j)}{\sum_{l \in RI_u} sim(i,l)} \quad (3)$$

In this equation $Y_{u,i}$ denotes the random variable for selecting item j amongst items rated by u while looking for an item similar to target item i. We return $r_{u,j}$ as the result of this random walk.

To define the whole probability distribution, we define the probabilities for conditions where $R_{u,i}$ is true, or for items not rated by u as follows:

$$\forall v \neq u \quad P(X_{u_0,i,k+1} = v | X_{u_0,i,k} = u, R_{u,i}) = 0 \quad (4)$$

$$\forall j.j \notin RI_u \quad P(Y_{u,i} = j) = 0 \quad (5)$$

4.1.1 Item Similarities

In content-based recommendation, the similarity of items can be computed using their features. However in collaborative filtering, the only information available about items is their ratings. Hence, to compute the similarity of two items, we use the *Pearson Correlation* of ratings expressed for both items, as used in [15]. Values of the Pearson correlation are in the range [-1,1]. Negative correlations mean that the ratings expressed for two items are in opposite directions, so these items are not useful for our purpose. Therefore, we only consider items with positive correlation.

$$corr(i,j) = \frac{\sum_{u \in UC_{i,j}} (r_{u,i} - \overline{r}_u)(r_{u,j} - \overline{r}_u)}{\sqrt{\sum_{u \in UC_{i,j}} (r_{u,i} - \overline{r}_u)^2} \sqrt{\sum_{u \in U} (r_{u,i} - \overline{r}_u)^2}} \quad (6)$$

$UC_{i,j}$ is the set of common users who have rated both items i and j, and $\overline{r}_u$ denotes the average of ratings expressed by u. $corr(i,j)$ denotes the correlation of items i,j.

The size of the set of common users is also important. For example, if $corr(i,j) = corr(i,l)$, but $|UC_{i,j}| > |UC_{i,l}|$, then, since i and j have been rated by more common users, so the correlation between them is stronger and $sim(i,j)$ should be greater than $sim(i,l)$. We consider $|UC_{i,j}|$ in the similarity measure as follows:

$$sim(i,j) = \frac{1}{1 + e^{-\frac{|UC_{i,j}|}{2}}} \times corr(i,j) \quad (7)$$

We used the sigmoid function to avoid favoring the size of $UC_{i,j}$ too much and to keep the similarity value in the range [0,1]. If the size of the set of common users is big enough, then the first part of equation 7 would converge to 1, but for small sets of common users, the factor would be 0.6. The number 2 in the denominator of the exponent is because we wanted to have a factor of greater than .9 if the size is greater than 5.

4.1.2 Termination of a Single Random Walk

At each user u, we have a probability $\phi_{u,i,k}$ of staying at u to select one of his items at step k of the random walk, while we are looking for a prediction on target item i. This probability should be related to the similarities of items rated by u and the target item i. similarity values are real numbers in $[0,1]$, so they can also be considered as probabilities. We consider the maximum similarity of items rated by u with target item i as the probability of staying at node u.

Furthermore, ratings on target item i from users far away from source user u_0 are noisy, but ratings expressed by trusted users nearby in the network are more reliable. So, the deeper we go into the network, the probability of continuing our random walk should decrease and so $\phi_{u,i,k}$ should increase.

To inject the factor k in $\phi_{u,i,k}$, we should use a function $f(k)$ which gives value 1 for big values of k, and a small value for small values of k. Since the sigmoid function satisfies our constraints for $f(k)$, we consider a sigmoid function of the k as another factor affecting $\phi_{u,i,k}$.

$$\phi_{u,i,k} = \max_{j \in RI_u} sim(i,j) \times \frac{1}{1 + e^{-\frac{k}{2}}} \quad (8)$$

Each random walk has three alternatives to stop:

1. Reaching a node which has expressed a rating on the target item i.
2. At some user node u, we decide to stay at the node and select one of the items rated by u and return the rating for that item as the result of random walk.
3. There is chance for a single random walk to continue for ever. To avoid such a case in our implementation of random walk, we terminate the random walk when we go very far from the source user ($k > max\text{-}depth$). Based on the idea of "six-degrees of separation"[11], we set $max\text{-}depth = 6$.

4.2 Recommendation

In TrustWalker, we have the probability of selecting items rated by different users and returning that rating as the result of a random walk. These items could be either the

exact target item i, or another item. The estimated rating for source user u on target item i would be the expected value of ratings returned by different random walks.

$$\widehat{r}_{u,i} = \sum_{\{(v,j)|R_{v,j}\}} P(XY_{u,i} = (v,j))\ r_{v,j} \tag{9}$$

In the above equation, $XY_{u,i}$ is the random variable for stopping the random walk at node v and selecting item j rated by v, while we start the random walk from source user u looking for target item i. Notice that the value for XY are ordered pairs. As used before, $R_{v,j}$ is a boolean variable denoting whether v has a rating on item j. Now we have:

$$P(XY_{u,i} = (v,j)) = \begin{cases} P(X_{u,i} = v)\phi_{u,i}P(Y_{v,i} = j) & v \neq u; i \neq j \\ P(X_{u,i} = v) & v \neq u; i = j \\ \phi_{v,i,1}P(Y_{v,i} = j), & v = u; i \neq j \end{cases} \tag{10}$$

In this equation, $X_{u,i}$ is the random variable for being at node v at some step in a random walk starting from source user u looking for item i. Notice that in above formula, we used $\phi_{u,i}$ instead of $\phi_{u,i,k}$ for the first case. Since we don't know the number of steps needed to reach v, we don't consider the factor k (Actually $\phi_{u,i} = \phi_{u,i,\infty}$). It should be noted that if we actually perform random walks, we can consider the step k in the first case. But to have a closed form formula, we ignore the factor k at the last user v which gives us a pretty good approximation of the probability. Also, we should note that the case $v = u$ and $i = j$ is trivial since the user himself has the rating on the target item.

A random walk starting from u can reach v using different number of steps. As mentioned before, we use random variable $X_{u,i,k}$ for being at node v in k steps

$$P(X_{u,i,k} = v) = \sum_{w \in U} P(X_{u,i,k-1} = w)(1 - \phi_{w,i,k})P(S_w = v) \tag{11}$$

Also we have $P(X_{u,i,0} = u) = 1$ as the base for the above equation. Since the random walks have a probability of stopping at each step, $\sum_{v \in U} P(X_{u,i,k} = v) \neq 1$. To make $P(X_{u,i,k} = v)$ a probability distribution, we define a *dead state* $\perp$ to which all users go after deciding to terminate that random walk. So, we have

$$P(X_{u,i,k} = \perp) = 1 - \sum_{v \in U} P(X_{u,i,k} = v)$$

This state $\perp$ will be added to U for convenience in formalization of our method, but we don't consider this state in any actual random walk. Now, we can compute $P(X_{u,i} = v)$ as follows:

$$P(X_{u,i} = v) = \frac{\sum_{k=1}^{\infty} P(X_{u,i,k} = v)}{\sum_{w \in U} \sum_{k=1}^{\infty} P(X_{u,i,k} = w)} \tag{12}$$

4.3 Matrix Notation of TrustWalker

Similar to any random walk model, we can represent TrustWalker using matrix notations. We consider a probability matrix $\mathbf{P}$ in which $\mathbf{P}_{u,v} = P(S_u = v) = \frac{1}{|N_u|}$ for all users u and v for which $t_{u,v} = 1$. To formulate the values of $\phi_{u,i,k}$ in matrix notation, we define a diagonal matrix $\mathbf{\Phi}_{k,i}$ for each item i and step k. $\mathbf{\Phi}_{k,i}$ is a $|U| \times |U|$ matrix containing $1 - \phi_{u,i,k}$ in its diagonal elements. In other words, $\mathbf{\Phi_{k,i}}_{u,u} = 1 - \phi_{u,i,k}$.

It is easy to check that the elements of $\mathbf{\Phi_{1,i}P}$ are the probabilities of reaching from u to v in step 1. We can also define a closure on the $\mathbf{\Phi_{k,i}P}$ as follows:

$$\begin{aligned} \mathbf{P_i^*} &= \sum_{K=1}^{\infty} \prod_{k=1}^{K} \mathbf{\Phi_{k,i}P} \\ &= \mathbf{\Phi_{1,i}P} + \mathbf{\Phi_{1,i}P\Phi_{2,i}P} + \mathbf{\Phi_{1,i}P\Phi_{2,i}P\Phi_{3,i}P} + ... \\ &= \mathbf{\Phi_{1,i}P}(\mathbf{I} + \mathbf{\Phi_{2,i}P}(\mathbf{I} + \mathbf{\Phi_{3,i}P}(...))) \end{aligned} \tag{13}$$

$\prod_{k=1}^{K} \mathbf{\Phi_{d,i}P}$ is the matrix containing $P(X_{u,i,K}) = v$ in its cells. In the above equation, k denotes the current step of each single random walk and K denotes then number of steps each random walk has. Now, we can compute $\widehat{\mathbf{P}}_\mathbf{i}$ (corresponding to equation 12) which is the probability matrix containing $P(X_{u,i} = v)$ in its elements as follows:

$$\widehat{\mathbf{P}}_\mathbf{i} = \mathbf{C_i P_i^*} \tag{14}$$

Here, $\mathbf{C_i}$ is an $N \times N$ diagonal matrix used for normalization. The values of the diagonal are $\mathbf{C_i}_{u,u} = \frac{1}{\sum_{w \in U} \mathbf{P_i^*}_{u,w}}$. Now we can use $\widehat{\mathbf{P}}_\mathbf{i}$ to compute $P(X_{u,i} = v)$ in equation 12.

If, we do not consider step k as a factor in $\phi_{u,i}$, it can be proved that $\mathbf{P_i^*}$ has a closed formula solution, and this closed formula is $(\mathbf{I} - \mathbf{\Phi_i P})^{-1}\mathbf{\Phi_i P}$. In this formula, $\mathbf{\Phi_i}$ is the same as $\mathbf{\Phi_{k,i}}$ ignoring the factor k. The proof of convergence of equation 13 is a straightforward application of the Perron-Frobenius theorem[6], and the lemma that all eigenvalues of $\mathbf{\Phi_i P}$ are strictly less than one.

Since matrices $\mathbf{\Phi_{k,i}}$ are different for different k values, we can not find a closed formula for $\mathbf{P_i^*}$. We propose two alternatives to compute equation 13.

- Performing the random walks. This way, we can see the results in action and the estimated value would be the aggregation of results of different random walks.
- Based on the idea of "six degrees of separation" [11], most nodes would be reachable with a walk of length at most 6. So, we can have a pretty good approximation of $\mathbf{P_i^*}$ by

$$\mathbf{P_i^*} = \sum_{K=1}^{6} \prod_{k=1}^{K} \mathbf{\Phi_{k,i}P} \tag{15}$$

This formula can be easily computed. But, there is a problem with the second approach. We have to store $\mathbf{P_i^*}$ associated with each item, which is expensive. For example if we have 10K items and 40K users, and each cell of the matrix occupies just one byte, then each matrix $\mathbf{P_i^*}$ would occupy almost $1.6GB$ memory. To store all matrices we would need $16TB$ of memory, which is not feasible. This issue motivates performing actual random walk and aggregate the results of different random walks. Moreover, computing this matrix needs a global information on the whole network, but TrustWalker can be computed in in a local manner.

Notice that there are major differences between our random walk model and existing random walk approaches such as [3] and [17]. In PageRank [3], there is a random walk on the links among WebPages, which correspond to users. But there is no item in PageRank and walks do not depend

[6]http://en.wikipedia.org/wiki/Perron-Frobenius_theorem

on the step of the random walk. Moreover, PageRank is a global reputation metric. These differences make PageRank simpler than TrustWalker to find the closed form solution. Nevertheless, they do not compute the closed formula because of the scale of network. In [17], they have an item graph on which they perform the random walk. This graph is independent of users, and hence they do not perform random walks on users. Also, unlike TrustWalker, their walks are independent of step k of the random walk. Having a simpler model, they are able to compute the closed formula for walking on the item graph.

4.4 Termination of the Overall Method

The results of performing actual random walks approximate the results given by equation 13. We perform several random walks to be able to get a more reliable prediction. We need to be able to decide when we have done enough random walks to have a precise estimate of $\widehat{r}_{u,i}$.

We compute the variance in the results of all the walks as follows:

$$\sigma^2 = \frac{\sum_{i=1}^{T} (r_i - \overline{r})^2}{T}$$

Here, r_i is the result of i^{th} random walk, and $\overline{r}$ denotes the average of the ratings returned by random walks. T is the number of random walks we perform to compute the prediction. We also define σ_i^2 as the variance in the results of the first i random walks. Since the values of ratings are in finite range of [1,5], it can be proved[7] that σ^2 converges to a constant value. So we can terminate TrustWalker if $|\sigma_{i+1}^2 - \sigma_i^2| \leq \epsilon$.

It should be noted that we have a constant threshold of 10000 for the maximum number of unsuccessful random walks, and after that we consider the pair <user,item> as non-covered.

5. PROPERTIES OF TRUSTWALKER

Our random walk model has some desirable properties, which include generality of the model, confidence in the result, and explainability of predictions.

5.1 Special Cases of TrustWalker

Our model includes Item-based Collaborative Filtering and pure Trust-based Recommendation as its extreme special cases. If $\phi_{u,i} = 1$ for all $u \in U$, then our random walk will never start, and it will return the rating expressed by the source user u_0 on one of its rated items. Since the probability of selecting an item is proportional to its similarity to the target item i, the expected value of the recommended rating would be the weighted average of the ratings on items in RI_{u_0} with weights proportional to the similarities of these items to the target item i. This is the same as the result of Item-based collaborative filtering proposed in [15].

On the other hand, if we set $\phi_{u,i} = 0$ for all $u \in U$, then all random walks will continue until they have found a rating for the exact target item i. The recommended rating would be the aggregation of ratings expressed by users having the rating on i weighted by the probability of reaching these users from u_0. Existing methods [5][10] try to approximate these probabilities by simplifying the problem. So our TrustWalker, in one of its extreme cases, can be considered as an ideal trust-based recommender.

[7]Proof of this claim and convergence of equation 13 are available in www.sfu.ca/šja25/trustwalker-proofs.pdf

5.2 Confidence in Recommendation

As discussed in the introduction, most existing recommenders do not give users the confidence in their predictions. However, users of a recommender like to know how confident the predicted ratings are. [2] defines a confidence score using the error function for its prediction. Basically, they consider the squared error as the confidence. Their experiments show that predictions with better confidence have higher quality. Our random walk model also has the advantage that it can compute the confidence.

We can use the variance σ^2 to compute our confidence in the predicted rating. The lower the variance σ^2, the more confident we are in our results. To convert this σ^2 into a value representing the degree of confidence, we employ the following formula:

$$confidence = 1 - \frac{\sigma^2}{max\sigma^2} \tag{16}$$

where $max\sigma^2$ is the maximum possible variance for the results and is used to normalize the variance values. If the rating values are in a finite range and the size of this range is *Range*, then it is easy to prove that $max\sigma^2 = \frac{Range^2}{4}$.

Two other factors may also potentially affect the confidence: the average number of steps in each random walk $\overline{k}$, and the number of random walks. Including $\overline{k}$ seems to make sense, but our experiments showed that it doesn't make any difference in the precision of results. In other words, the error for predictions with lower $\overline{k}$ is not much better than the overall error. Regarding the number of random walks, more random walks can lead to more information and more confidence, but can also indicate more noise since it took longer for the results to converge.

Using equation 16, if the results of the random walks are the same, then the variance would be zero, and the confidence would be 100%. On the other side, if the variance is very high, then the confidence will be close to 0%.

5.3 Explainability of Recommendations

Explainability means that the recommender system is able to explain how it predicted the rating. There is now a growing recognition that recommender systems must be able to explain and justify the recommendations in order to help users to understand why particular items have been suggested.

In our model, to predict $\widehat{r}_{u,i}$, we compute $P(X_{u,i} = v)$ for all users v. The results of different random walks are from different user. The most frequent users are user with high probability of $P(X_{u,i} = v)$. We can output these users as users whose ratings are most influential on the prediction.

Also considering ratings on some items are more frequently used in the results of different random walks. These items are items with high values of $P(XY_{u,i} = (v, j))$.

Now we can use these most frequent users and items to explain why we predicted the rating with $\widehat{r}_{u,i}$. We can explain to users that this prediction is based on ratings from these trusted users and these similar items.

6. EXPERIMENTS

This section reports our experimental results on a real life data set comparing various versions of TrustWalker against

state of the art methods for trust based and item based recommendation. We implemented TrustWalker as well as the MoleTrust based recommendation proposed by Massa[10] and TidalTrust proposed by Golbeck[5]. We also implemented standard User based Collaborative Filtering [6] and Item based Collaborative Filtering [15] as two fundamental similarity based recommendation methods.

6.1 Experimental Design

Most data sets for recommendation have no social network among users. To best of our knowledge, Epinions'[8] data set is the only data set publicly available which has both trust network and ratings expressed by users.

We used the version of the Epinions' data set[9] published by the authors of [14]. This data set is very sparse. The data set contains 49k users with at least one rating, out of which 24k users are cold start users. We consider users with less than 5 ratings as cold start users (similar to [10]). We also have 104k items with 575k ratings expressed for them. Finally, we have 508k trust statements among pairs of users. As mentioned above, 49% of users are cold start users which is a huge portion of users. So, considering the performance of the recommendation for cold start users is very important. The distribution of the number of ratings per users follows a power law. Notice that there exists another version of the Epinions data set[10] prepared and published by Paolo Massa[10]. We also ran experiments on this data set with very similar results. The data set we used in our experiments has additional features which may be helpful for future works.

We implemented TrustWalker and the other methods in Java. We used an Intel Core2 Duo 2.2 GHz CPU with 2GB RAM to run our experiments on an XP system.

In our experiments, we compare the results for different methods. Following is the description of labels we use to denote each of these algorithms:

- *TidalTrust.* This is the trust-based approach of [5].
- *MoleTrust.* This is the approach used in [10]. It should be noted that we use $max - depth = 6$ for *ModelTrust* as well.
- *CF Pearson.* We implemented the user based collaborative filtering [6], with the Pearson Correlation as similarity measure.
- *Item based.* We also implemented the item based collaborative filtering [15] using Pearson Correlation as the item similarity metric.
- *Random Walk* This is one of the special cases of TrustWalker with $\phi_{u,i,k} = 0$ for all (u, i, k). Also we set different thresholds on the number of steps a random walk. *Random Walk 1* represents the case in which we just walk for one step, and in *Random Walk 6* each random walk could continue until 6 steps.
- *TrustWalker0.* This method is the version of our TrustWalker in which $\phi_{u,i}$ is independent of k. Our item similarity metric considers the size of the set of common users in computing the similarities. We also performed experiments in which we only considered Pearson correlation as similarity metric. The results of this version are denoted by *TrustWalker0-pure.*
- *TrustWalker.* This is our full TrustWalker method. We also have *TrustWalker-pure.*

We set $\epsilon = 0.0001$ for our termination condition.

[8]www.epinions.com

[9]http://alchemy.cs.washington.edu/data/epinions/

[10]http://www.trustlet.org/wiki/Epinions_dataset

6.2 Evaluation Metrics

Typically, the leave-one-out method is used to evaluate recommendation systems [5][10][15]. In the leave-one-out method, we withhold a rating and try to predict it using the trust network and the remaining ratings.

As used in the most recent research papers [17] [7], we use the Root Mean Squared Error (RMSE) to measure the error in recommendation:

$$RMSE = \sqrt{\frac{\sum_{(u,i)|R_{u,i}} (r_{u,i} - \widehat{r}_{u,i})^2}{|\{(u,i)|R_{u,i}\}|}} \tag{17}$$

In the above equation, $R_{u,i}$ is a boolean showing whether u has a rating on i in our data set, and $r_{u,i}$ and $\widehat{r}_{u,i}$ denote the actual and recommended rating respectively. The smaller the value of *RMSE*, the more precise a recommendation. But, as we discussed before, the purpose of using trust is mostly enhancing the coverage without sacrificing the precision. So we define a *coverage* metric to measure the percentage of pairs of <user,item> for which we can predict a rating.

To combine *RMSE* and coverage into a single evaluation metric, we compute the *FMeasure* (Coverage has been already defined in section 2). For this purpose, we have to convert *RMSE* into a precision metric in the range [0,1]. So we define *precision* as follows:

$$Precision = 1 - \frac{RMSE}{4} \tag{18}$$

In this equation, 4 is the maximum possible error since the values of ratings are in the range $[1, 5]$.

$$FMeasure = \frac{2 \times Precision \times Coverage}{Precision + Coverage} \tag{19}$$

If none of the random walks can find a prediction on the rating, then we say that the recommender can not cover this pair of <user,item>.

6.3 Experimental Results

In this subsection, we present the results of our experiments, first for cold-start users and then for all users.

Table 2 shows the RMSE, Coverage, and F-Measure for all comparison partners for cold start users. Figures 2 and 3 show the charts comparing the results of different methods according to each of three evaluation measures separately.

As shown in figure 2, *TrustWalker* has lower error than all the other methods except for *RandomWalk1*. But according to table 2, *RandomWalk1* has extremely low coverage (12%). The comparison between *TrustWalker* and *TrustWalker0* shows that considering the step k in $\phi_{u,i}$ reduces the error. Also comparing the RMSE of *TrustWalker* and *TrustWalker-pure* shows that considering the size of the set of common users in the item similarity metric reduces the error.

Figure 3 shows the F-Measure together with precision and coverage for all methods. It shows that all four versions of

Method	RMSE	Coverage(%)	F-Measure
Tidal Trust	1.216	60.92	0.650
Mole Trust	1.430	57.75	0.608
CF Pearson	1.495	17.94	0.279
RandomWalk 6	1.206	61.74	0.655
RandomWalk 1	1.084	12.77	0.217
Item based	1.520	23.19	0.338
TrustWalker0-pure	1.293	72.08	0.698
TrustWalker0	1.263	74.22	0.712
TrustWalker-pure	1.259	72.08	0.703
TrustWalker	1.192	74.22	0.722

Table 2: Results for *COLD START users*.

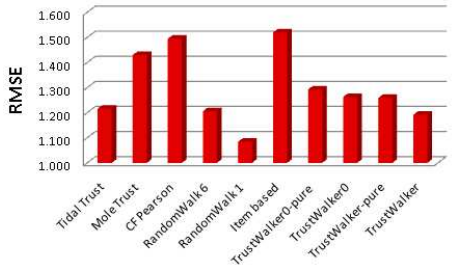

Figure 2: Comparison of RMSE for different methods on COLD START users.

TrustWalker outperform all other methods according to the combination of precision and coverage. Notice that TidalTrust[5] outperforms all versions of our model in terms of precision except for the full *TrustWalker* which has a slightly lower error. However, TrustWalker's coverage is 13% more than that of TidalTrust, which makes TrustWalker better in terms of F-Measure.

The results for all users are shown in table 3. Figure 4 shows the results of different methods according to each of the three evaluation measures on all users. We observe similar relative performance of all methods as for cold start users. It should be noted that all methods perform significantly better over all users since there is less information available for cold start users. TrustWalker outperforms all other methods in terms of F-Measure, although the gain for all users is less than the gain for cold start users. Notice that the errors of both TidalTrust and MoleTrust for All users are very close to TrustWalker, but TrustWalker clearly has a better coverage.

Method	RMSE	Coverage(%)	F-Measure
TidalTrust	1.096	84.87	0.783
MoleTrust	1.093	82.13	0.771
CF Pearson	1.327	71.63	0.691
RandomWalk 6	1.089	87.19	0.793
RandomWalk 1	1.154	30.62	0.428
Item based	1.244	70.69	0.698
TrustWalker0-pure	1.158	95.29	0.814
TrustWalker0	1.112	95.36	0.822
TrustWalker-pure	1.109	95.29	0.822
TrustWalker	1.077	95.36	0.827

Table 3: Experimental results for *ALL users*.

In summary, TrustWalker substantially improves the coverage of existing trust-based approaches while maintaining the same or even slightly better precision. This improvement is achieved by considering ratings for similar items as well as

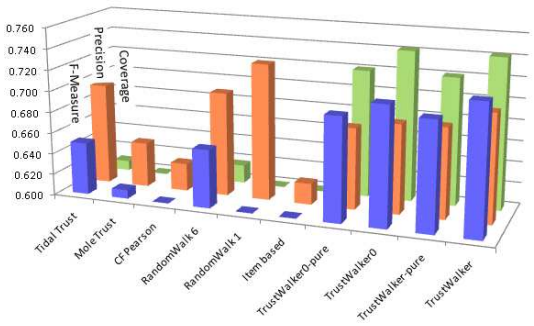

Figure 3: Comparison of Coverage and Precision together with F-Measure for different methods on COLD START users.

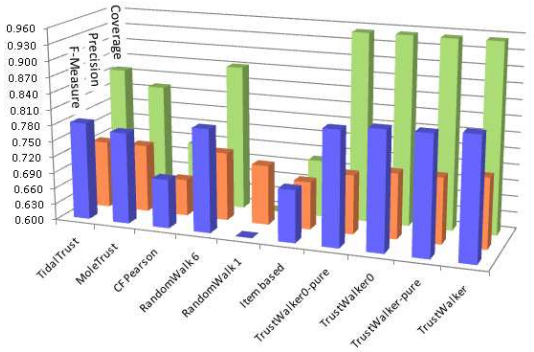

Figure 4: Comparison of evaluation measures for different methods on ALL users.

the exact target item. TrustWalker also clearly outperforms Collaborative Filtering (both user-based and item-based) in terms of coverage because of exploiting the trust-network in its random walks. Moreover, TrustWalker clearly outperforms Collaborative Filtering methods in terms of precision due to the restriction to ratings from highly trusted users. Since cold start users have only a few ratings, the improvement of coverage using trust-based approaches (specially TrustWalker) compared to Collaborative Filtering is much more for cold start users than for all users.

For example, TrustWalker's coverage is 74% for cold start users while Collaborative Filtering approaches have coverage of 18% and 23%. But in case of all users, TrustWalker's coverage is 95% while Collaborative Filtering approaches have coverage of 71%. TidalTrust and MoleTrust have coverage of 84% and 82% respectively for cold start users while their coverage is 61% and 57% in case of all users.

To evaluate the meaningfulness of our definition of confidence, we compare the error of TrustWalker for different percentiles of most confident predictions (for both *Cold Start* and *All* users) in figure 5. As shown in this figure, the more confident the predictions are the more precise the results will be and hence the lower the RMSE. For example, the difference between the RMSE of the top 50% confident predictions and all predictions is 0.18 for cold start users. As an evidence for the significance of these RMSE reductions, note that in the Netflix prize competition[11], there is a $1 Million reward for a reduction of the RMSE by 10% (around 0.1).

7. CONCLUSION

Recommender systems are emerging as tools of choice to

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

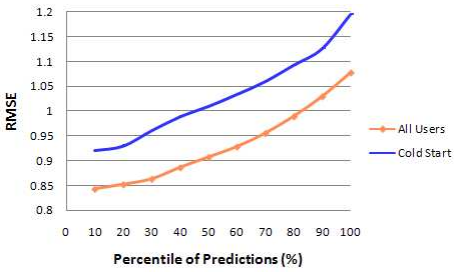

Figure 5: The effect of confidence on the precision of recommendations. The horizontal axis shows percentile of most confident predictions from dataset.

select the online information relevant to a given user. Collaborative filtering is the most popular approach to building recommender systems and has been successfully employed in many applications. However, it cannot make recommendations for so-called cold start users that have rated only a very small number of items. Collaborative filtering methods are rather vulnerable to attacks that insert fraudulent user profiles into the database. Finally, these methods do not know how confident they are in their recommendations. Trust-based recommender systems can better deal with cold start users, since users only need to be simply connected to the trust network. The trust-based approach is also much more robust to fraudulent attacks.

On the other hand, the sparsity of the user item ratings forces the trust-based approach to consider ratings of indirect neighbors that are only weakly trusted, which may decrease its precision. In order to address this problem, we proposed the random walk model TrustWalker to combine the trust-based and the item-based collaborative filtering approach to recommendation. TrustWalker considers not only ratings of the target item, but also those of similar items, with probability increasing with increasing length of the walk. As another contribution, TrustWalker allows us to define and to measure the confidence of a recommendation. We performed an evaluation on the Epinions dataset, demonstrating that TrustWalker outperforms both collaborative filtering methods and purely trust-based methods specially in terms of coverage. Also TrustWalker has a slightly better precision compared to other Trust-based approaches while clearly outperforming Collaborative Filtering approaches in terms od precision. We also showed that highly confident recommendations are of even greater quality.

This work suggests several interesting directions for future work. We want to evaluate TrustWalker on other data sets as soon as they become available. Besides the problem of predicting a rating for a given item, recommending a list of items to a user is also a natural task for recommenders. We plan to investigate the extension of the TrustWalker model for this task. In this paper and in the related work, the ratings are assumed to be stored in a centralized repository. However, applications such as mobile phone-based social networks require a distributed recommender, and a random walk model is a promising approach for such scenarios. The trust concept we considered in this paper is context independent. However, people may trust other people in some context while they do not trust those people in other contexts. Investigating context-based trust models for recommendations is also an interesting direction for future work.

8. ACKNOWLEDGEMENT

We are very grateful to Samaneh Moghaddam for her useful comments and for proof reading this paper.

9. REFERENCES

[1] R. Andersen, C. Borgs, J. Chayes, U. Feige, A. Flaxman, A. Kalai, V. Mirrokni, and M. Tennenholtz. Trust-based recommendation systems: an axiomatic approach. In *WWW 2008*.

[2] R. M. Bell, Y. Koren, and C. Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *KDD 2007*.

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1), 1998.

[4] D. Crandall, D. Cosley, D. Huttenlocher, J. Kleinberg, and S. Suri. Feedback effects between similarity and social influence in online communities. In *KDD 2008*.

[5] J. Golbeck. *Computing and Applying Trust in Web-based Social Networks*. PhD thesis, University of Maryland College Park, 2005.

[6] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12), 1992.

[7] Y. Koren. Factorization meets the neighborhood a multifaceted collaborative filtering model. In *KDD 2008*.

[8] Levien and Aiken. Advogato's trust metric. online at http://advogato.org/trust-metric.html, 2002.

[9] H. Ma, H. Yang, M. R. Lyu, and I. King. Sorec: social recommendation using probabilistic matrix factorization. In *CIKM '08*, 2008.

[10] P. Massa and P. Avesani. Trust-aware recommender systems. In *ACM Recommender Systems Conference (RecSys)*, USA, 2007.

[11] S. Milgram. The small world problem. *Psychology Today*, 2, 1967.

[12] J. O'Donovan and B. Smyth. Trust in recommender systems. In *10th international conference on Intelligent user interfaces*, USA, 2005.

[13] A. Rettinger, M. Nickles, and V. Tresp. A statistical relational model for trust learning. In *AAMAS '08: 7th international joint conference on Autonomous agents and multiagent systems*, 2008.

[14] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD 2002*.

[15] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW 2001*.

[16] S. Wasserman and K. Faust. *Social Network Analysis*. Cambridge Univ. Press, 1994.

[17] H. Yildirim and M. S. Krishnamoorthy. A random walk method for alleviating the sparsity problem in collaborative filtering. In *ACM Conference on Recommender Systems (RecSys)*, Switzerland, 2008.

[18] C. N. Ziegler. *Towards Decentralized Recommender Systems*. PhD thesis, University of Freiburg, 2005.