Trust-based Agent Community for Collaborative Recommendation

Jianshu Weng
School of Computer Engineering,
Nanyang Technological University,
Singapore
WENG0004@ntu.edu.sg

Abstract

There exists a number of similarity-based recommendation communities, within which similar agents' ratings on items are collected by agents to make predictions on new items' ratings. Despite its success, similarity-based recommendation communities suffer from some significant limitations, such as scalability and susceptibility to the noise. In this paper, we propose a trust-based recommendation community to overcome these limitations. In the trust-based recommendation community incorporates, trustworthy users' ratings are taken into account when making predictions on new items' ratings. A trust metric is designed to quantify the degrees of trust an agent should place on others. Experimental results based on a real dataset show that trust does improve the recommendation community's performance in terms of prediction accuracy, coverage, and robustness in the presence of noisy ratings.

1 Introduction

1.1 Motivations

With the development of Web technology and Agent technology, people start to shift many activities from the real world into the cyber world. A typical example is the emergence of online communities in recent years, e.g. communities for trading goods online like eBay.com, communities for products or items recommendation like Movielens[1].

The focus of this paper is the recommendation communities. Users have their agents join the community to collect recommendations on products or items from others[2]. The collected recommendations are then used by their agents to predict the degrees of satisfaction on new products, based on which users make the decisions whether to buy or browse the new products.

Traditionally, similarity-based approach is applied to make predictions: the similarities between pairs of agents are derived according to the "agreements" between individual items over all the items co-rated by agents, and the predictions are made as weighted average of the collected recommendations with the similarities between agents as weights [18].

There exist a number of similarity-based communities, e.g. Movielens and Epinions.com. Despite the success, the similarity-based communities have major limitations, for example:

(1) Agents usually rate few items [5]. It is quite often the case that agents do not co-rate the minimum number of items in common required to compute the similarity. For example, it is required that agents must co-rate at least two items in common to calculate the Pearson correlation coefficient, which is a widely used similarity metrics. Hence, the number of similar agents is usually small. As a consequence, a large number of agents are not able to make predictions, since no similar agents can be located.

(2) By intentionally rating only a small number of items, malicious agents are easier to achieve high similarity since the number of possible common items is small. For example, Pearson correlation gives a similarity of either 1, -1 or 0 when the number of common items is 2. Hence, malicious agents' noisy ratings[3] can easily outweigh others' ratings, leading to the biased predictions on given items.

[1]Movielens: http://movielens.umn.edu/

[2]In the rest of this paper, we shall use the *agent* to denote the user who is the owner of the agent.

[3]Presence of noisy ratings means that some malicious agents might give unfairly positive or negative recommendations for the purpose of biasing other agents' predictions for or against certain items.

(3) Most of the communities are centralized. It is generally required that a centralized server goes through every member to calculate its similarity between the active agent (i.e. the one for whom the community is making prediction). The computation burden of the centralized server increases quickly with the increase of the number of agents, leading to poor scalability.

To overcome the above-mentioned limitations, we incorporate *trust* into the domain of item recommendation.

1.2 Defining Trust

After Marsh's seminal work [11] to formalize trust in a computational way, there have been emerging research efforts to formalize computational trust and to apply trust in different domains of computer science, e.g. multi-agent system [17, 20], Semantic Web [19], and so on. Trust provides a mechanism to allow agents to reason about others' reliability and competence within the multi-agent system. In such systems, agents interact with others whose reliability and competence are usually not known.

Trust is a context-specific concept, and hence it is not surprising to see different interpretation of trust [9]. For the purpose of this work, we adapt Barber's definition[2] which defines *trust* to be *the expectation of technically competent role performance*. More specifically, we interpret *trust* as an agent's expectation of another agent's competence in providing recommendations to reduce its uncertainty in predicting new items' ratings.

Suppose agent Ag_t usually gives high ratings on items which agent Ag_s gives low ratings on, and vice verse. The similarity between Ag_t and Ag_s is very low in that they have low "agreement" on each individual item. However, Ag_s can still place high degree of trust on Ag_s since Ag_s can predict the new items' ratings to be low (high) if Ag_t gives a high (low) rating for the new items. In other words, Ag_t is trustworthy to Ag_s since Ag_t has met Ag_s's expectation in reducing the uncertainty when predicting new items' ratings. Our interpretation of trust also resembles the practical definition of trust adopted by Epinion.com: Ag_s can trust Ag_t if Ag_t's ratings "have been consistently found to be valuable" [1].

1.3 Overview of the Trust-based Recommendation Community

Agents connect with other trustworthy agents within the community, which is basically a P2P overlay network. P2P-based architecture exempts the need for a centralized server, which makes the trust-based community more scalable than the traditional similarity-based community. Each agent acts as a peer servent within the community. It contributes to the community by sharing recommendations on items to others. At the same time, it asks others for their recommendations and makes predictions based on local computation.

The three main components of the trust-based recommendation community are:

(1) A trust metric is designed to quantify the degrees of trust agents should place on the other agents. The trust metric is computable on most agents. It is even computable on pairs of agents who only co-rate one item in common.

(2) The predictions are made as weighted averages of the collected recommendations with agents' trustworthiness as weights.

(3) The P2P-based architecture is exploited to facilitate agents' recommendation discovery and collection, which resembles the content searching in P2P content sharing network.

It should be noted that the work presented in this paper is not only applicable in the domain of recommendation, although the current work is conducted in this specific domain. In fact, our work is generally applicable in domains in which agents exchange information with each other and agents' expectance and experience can be captured in the form of ratings [13].

The rest of this paper is organized as follows. The trust metric is explained in Section 2. How to make predictions is discussed in Section 3. Experimental results are presented and analyzed in Section 4. A brief review of related work is given in Section 5. Finally, Section 6 concludes the paper.

2 The Trust Metric

A trust metric is designed in this section, which quantifies the degree of trust an agent should place on another agent. As discussed earlier, each agent is treated equally as a peer servent. Without loss of generality, the rest of this paper is hence presented from the perspective of a particular agent Ag_s.

For the ease of the presentation, we first present some notations which will be used throughout this paper.

There are two categories of entities within the community.

(1) a set of uniquely identifiable *items* $O = \{O_1, O_2, \ldots, O_j, \ldots, O_m\}$,

(2) a set of uniquely identifiable *agents* $Ag = \{Ag_1, Ag_2, \ldots, Ag_i, \ldots, Ag_n\}$, who give discrete ratings $r_{i,j}$ on the *items*. $r_{i,j}$ denotes Ag_i's *rating* on O_j after browsing O_j.

Ag_s**'s ratings**	Ag_t**'s ratings**					Total
	1	...	j	...	Z	
1	n_{11}	...	n_{1j}	...	n_{1Z}	R_1
$\vdots$			$\vdots$			$\vdots$
i	n_{i1}	...	n_{ij}	...	n_{iZ}	R_i
$\vdots$			$\vdots$			$\vdots$
Z	n_{Z1}	...	n_{Zj}	...	n_{ZZ}	R_Z
Total	C_1	...	C_j	...	C_Z	N

Table 1: Ag_s's experience with Ag_t

Agents' predictions are also given in forms of ratings, with $Pr_{i,j}$ denoted Ag_i's prediction on O_j. Agents' predictions are derived based on trustworthy agents' recommendations. The degree of trust that an agent Ag_s places on another agent Ag_t is termed as Ag_s's evaluation of Ag_t's "*trustworthiness*".

Note that agents' *prediction* for an item is different semantically from its *rating* on an item. The prediction is its pre-experience with the item, while the rating is its post-experience with the item. Usually there is a threshold, and agent Ag_s opts to browse the item only if its prediction on the item is higher than the threshold. More formally, Ag_s can make the decision by taking into account the utility and risk of the browsing [3]. Only after it browses the item will it have a rating on the item. When asked by the other agents for recommendations, an agent shares its rating (i.e. post-experience with items) but not the predictions.

2.1 Derivation of the Trust Metric

The trust metric is experience-based. That is to say, Ag_s's evaluation of agent Ag_t's trustworthiness is calculated based on its previous experience with Ag_t[4]. Ag_s compiles its experience with Ag_t in a table as shown in Table 1. In Table 1, Z denotes the number of possible ratings, and N is the total number of common items co-rated by Ag_s and Ag_t before. Each cell n_{ij} in Table 1 records the number of co-rated items on which Ag_t's recommendations are j while Ag_s's ratings are i. $n_{ij} = 0$ if there is no such common item. R_i is the sum of all the cells in the row i, i.e. $R_i = \sum_{j=1}^{Z} n_{ij}$. And C_j is the sum of all the cells in the column j, i.e. $C_j = \sum_{i=1}^{Z} n_{ij}$. $\sum_i R_i = \sum_j C_j = N$. After compiling the experience with Ag_t, agent Ag_s can evaluate Ag_t's *trustworthiness*:

Definition 1 *Ag_s evaluates Ag_t's* ***trustworthiness*** *$Tr_{s,t}$ based on previous experience with Ag_t as:*

[4]Ag_s has previous experience with Ag_t means Ag_t's recommendations have ever been used by Ag_s to make predictions.

$$Tr_{s,t} = \frac{N \sum_i \sum_j \frac{n_{ij}^2}{C_j} - \sum_i R_i^2}{N^2 - \sum_i R_i^2} \tag{1}$$

∎

Some explanations would be helpful in understanding the derivation of $Tr_{s,t}$. Consider two cases:

Case (1) : Agent Ag_s predicts a new item's rating without Agent Ag_t's recommendation;

Case (2) : Agent Ag_s predicts a new item's rating given Agent Ag_t's recommendation, say j.

In **Case (1)**, Ag_s predicts the new item's rating solely based on its own rating history, i.e. it predicts the new item's rating as a particular rating i with a probability R_i/N. In the long run, the proportion of the correct predictions (i.e. prediction=rating) in this case is $\sum_i (R_i/N)^2$ [6]. In **Case (2)**, the probability that Ag_s predicts the new item's rating as i given Ag_t's recommendation j is essentially the conditional probability of i's occurrence given j's presence in previous experience. That is, Ag_s predicts the item's rating as i with a probability $(\frac{n_{ij}}{N})/(\frac{C_j}{N})$. In the long run, the proportion of the correct predictions in this case is $\sum_i \sum_j (\frac{n_{ij}}{N})^2/(\frac{C_j}{N}) = \frac{1}{N}\sum_i \sum_j (n_{ij}^2/C_j)$ [6].

As discussed before, *trust* is interpreted as an agent's expectation of another agent's competence in providing recommendations to reduce its uncertainty in predicting new items' ratings. Hence, trustworthiness can be quantified as an agent's degree of reduction in the uncertainty of predicting new items' ratings given others' recommendations, which is basically the reduction in the proportion of the *incorrect* predictions as Ag_s goes from **Case (1)** to **Case (2)**. Hence, we have:

$$\begin{aligned} & \frac{(1-\sum_i (R_i/N)^2)-(1-\frac{1}{N}\sum_i \sum_j (n_{ij}^2/C_j))}{1-\sum_i (R_i/N)^2} \\ = & \frac{N \sum_i \sum_j \frac{n_{ij}^2}{C_j} - \sum_i R_i^2}{N^2 - \sum_i R_i^2} \end{aligned}$$

Thus, the definition of $Tr_{s,t}$ in Eq (1) is reached.

$Tr_{s,t}$ takes values in $[0,1]$. $Tr_{s,t} = 0$ implies that Ag_t does not reduce Ag_s's uncertainty at all. $Tr_{s,t} = 1$ implies Ag_s's perfect predictability of the new items' ratings given Ag_t's recommendation.

Note that different agents may have different evaluation of Ag_t's trustworthiness. This is because the trust metric is experience-based and different agent may have different experience with Ag_t.

2.2 Dummy Experience

As discussed before, $n_{ij} = 0$ in Table 1 means there are no co-rated items on which Ag_t's recommendations are j while Ag_s's ratings are i. Intuitively, each cell n_{ij} in Table 1 should be initiated as 0. However, this intuitive treatment will lead to the undefined 0/0 in the value of $Tr_{s,t}$ when there is only one co-rated item between Ag_s and Ag_t.

In order to avoid the undefined 0/0, the *dummy experience* is introduced. That is, each cell in Table 1 is initiated with dummy experience as $1/Z$ instead of 0. After initialization, $Tr_{s,t} = 0$. This is consistent with the common sense that Ag_t does not reduce Ag_s's uncertainty before their first experience. After Ag_t shares recommendation on item as j for the first time, the dummy experience in the column j of Table 1 is cleared first, and then corresponding cell in column j is updated with real co-rated item counts.

There is another case leading to the undefined 0/0: each column of Table 1 has only one non-zero cell, and all the non-zero cells are in the same row. In this case, there is only one non-zero cell in each column, which means Ag_s always has only one possible prediction given Ag_t's recommendations. This implies Ag_s's perfect predictability of new items' ratings give Ag_t's recommendations. Consequently, $Tr_{s,t} = 1$ in this case.

2.3 Update of the Trustworthiness

When Ag_s joins the community, it randomly selects a number of existing member agents, with whom it exchanges ratings on some items. Ratings exchanged at this stage are also taken as Ag_s's experience with the other agents. Ag_s evaluates the trustworthiness of each of those randomly-selected agents using Eq.(1), and connects with at most D agents with highest trustworthiness score as its *direct neighbors*. Then with the help of trust propagation (trust propagation will be discussed in the next section), Ag_s will discover a larger number of trustworthy agents. At the same time, with more items encountered, Ag_s's experience with the other agents evolves. Ag_s update its evaluations of those agents' trustworthiness accordingly since the trust metric is experience-based.

Update of an agent's trustworthiness score is triggered upon Ag_s obtaining its rating on a new item, i.e. its post-experience with the item after browsing the item. Suppose Ag_s just browsed the item O_n, and its rating on O_n is i. When making the prediction, Ag_t contributes its recommendation on O_n as j. If Ag_t is an agent that Ag_s has previous experience, Ag_s updates its experience with Ag_t as $n_{ij} = n_{ij} + 1$ in Table 1[5]. Otherwise, Ag_s creates a new table to

[5]The dummy experience in column j needs to be cleared first if it is the first time that Ag_t shares a recommendation as j.

	Ag_t's ratings					
Ag_s's ratings	1	2	3	4	5	Total
1	1/5	1/5	1/5	1/5	0	4/5
2	1/5	1/5	1/5	1/5	0	4/5
3	1/5	1/5	1/5	1/5	1	9/5
4	1/5	1/5	1/5	1/5	0	4/5
5	1/5	1/5	1/5	1/5	0	4/5
Total	1	1	1	1	1	5

Table 2: Ag_s's experience with Ag_t

record the experience with Ag_t. Dummy experience in column j is cleared first, then n_{ij} is set to 1. $Tr_{s,t}$ is then updated using Eq. (1) correspondingly.

An example is used to illustrate the evaluation of the trustworthiness and its update. Suppose a 5-point discrete rating schema is applied. Agent Ag_t's recommendation on an item is 5, and Ag_s's own rating on the item is 3. Ag_s has no previous experience with Ag_t, so the experience with Ag_t is updated by creating a new table, clearing the dummy experience in the 5th column first, and then filling the cell $n_{35} = 1$. After that, Ag_s's experience with Ag_t is shown in Table 2.

Now $Tr_{s,t}$ can be evaluated as:

$$Tr_{s,t} = \frac{5*(4*5*(1/5)^2+1)-(4*(\frac{4}{5})^2+(\frac{9}{5})^2)}{5^2-(4*(\frac{4}{5})^2+(\frac{9}{5})^2)} = \frac{225-145}{625-145} = 0.1667$$

As this example shows, the trustworthiness score is computable on pairs of agents with only one common item. Moreover, an agent is not able to achieve a high trustworthiness by intentionally sharing recommendations only on a small number of items.

3 Recommendation Collection and Rating Prediction

We have presented the evaluation and update of trustworthiness in Section 2. In this section, we shall discuss how to discover and collect trustworthy agents' recommendations, and how to make predictions based on trustworthy agents' recommendations.

3.1 Recommendation Discovery and Trust Propagation

Ag_s goes to its direct neighbors first when discovering recommendations. Each direct neighbor gives its recommendation if it does have rating on the target item. Moreover, each direct neighbor is also required to returns all its direct neighbors as well as its evaluation of their trustworthiness as referrals to Ag_s. Instead of going to all the referrals, Ag_s applies a probabilistic strategy in determining whether to visit a particular referral at the next step. Suppose Ag_d is the currently-visited agent, and it returns a number of referrals. Ag_s opts to visit a particular referral Ag_m at next step with a probability $Tr_{d,m}$, and opts not to visit Ag_m with a probability $1-Tr_{d,m}$. Here, $Tr_{d,m}$ is Ag_d's evaluation of Ag_m's trustworthiness. Then each visited agent carries out the similar actions as Ag_s's direct neighbors do. The only difference is that it returns referrals only when it has no recommendation to give.

The referral-based recommendation discovery resembles the content searching in the unstructured P2P content sharing network like Gnutella[6]. The main difference is that: Ag_s applies a probabilistic strategy in determining whether to visit a particular referral during the course of recommendation discovery; while in P2P content searching, usually all referrals are visited. The probabilistic strategy has the following advantages:

- Agents with higher trustworthiness have a higher probability to be visited. This helps Ag_s to discover more trustworthy recommendations, which makes the prediction more accurate.
- Nevertheless, agents with lower trustworthiness may also be visited by Ag_s though with lower probability. This gives those less trustworthy agents opportunities to contribute recommendations, which helps to promote Ag_s's evaluation of their trustworthiness.

During the recommendation discovery, Ag_s will visit an agent only once in order to avoid duplicated recommendations from the same agent. For example, in Figure 1, Ag_3 and Ag_6 are two referrals returned by Ag_5. However, Ag_1 does not visit them (shown by the dashed line in the figure) since they are visited and their recommendations has been collected already. The path starting from the initiating agent Ag_s connecting its direct neighbor and the series of visit agents during the recommendation discovery is called the *referral chain*. And agents who can give recommendations are always at the other end of the referral chain. An upper limit (U) is applied on the referral chain length. Ag_s stops discovering recommendations if the current referral chain length reaches U no matter whether recommendation has been collected or not by following this chain.

Recommendation discovery usually discovers more than one agent that gives recommendation. For example, in Figure 1 both Ag_4 and Ag_7 give recommendation on item O_1. Trust metric can help Ag_s evaluate those agents' trustworthiness, if it has previous experience with them. However, it is not possible that

[6]Gnutella RFC: http://rfc-gnutella.sourceforge.net/.

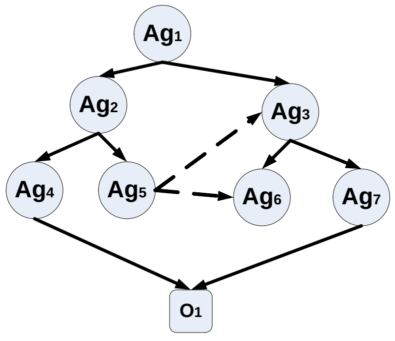


Figure 1: Agent Ag_1 collects recommendations for Item O_1

Ag_s has experience with all of the agents that give recommendations. Hence, trust propagation is necessary to help Ag_s evaluate the trustworthiness of agents who give recommendations but it has no previous experience with.

A number of trust propagation mechanisms have been proposed, e.g. [8, 13]. A detailed discussion and comparison of trust propagation is beyond the scope of this paper. In this paper, we simply employ the trust propagation mechanism proposed in [13]. Suppose in a given referral chain starting from agent Ag_s, Ag_m is one of Ag_d's referrals to Ag_s. Ag_s evaluates Ag_m's trustworthiness via *trust propagation* as:

$$Tr_{s,m} = Tr_{s,d} * Tr_{d,m} + (1 - Tr_{s,d}) * (1 - Tr_{d,m}) \tag{2}$$

where $Tr_{s,d}$ is Ag_s's evaluation of Ag_d's trustworthiness, and $Tr_{d,m}$ is Ag_d's evaluation of Ag_m's trustworthiness.

Then Eq. (2) is computed recursively by Ag_s over all the agents along the referral chain to derive the trustworthiness of the agent at the other end of the referral chain. For example, in Figure 1, there are two referral chains: $Ag_1 \rightarrow Ag_2 \rightarrow Ag_4$ and $Ag_1 \rightarrow Ag_3 \rightarrow Ag_7$. Along each chain, $Tr_{1,2} = 0.4$, $Tr_{2,4} = 0.4$, $Tr_{1,3} = 0.6$, $Tr_{3,7} = 0.1$, so $Tr_{1,4} = 0.4 * 0.4 + 0.6 * 0.6 = 0.52$, and $Tr_{1,7} = 0.6 * 0.1 + 0.4 * 0.9 = 0.42$. Suppose Ag_4 has no recommendation on O_1, and returns Ag_8 as a referral. $Tr_{4,8} = 0.3$, then $Tr_{1,8} = Tr_{1,4} * Tr_{4,8} + (1 - Tr_{1,4}) * (1 - Tr_{4,8}) = 0.52 * 0.3 + 0.48 * 0.7 = 0.492$.

3.2 Rating Prediction

The collected recommendations are adjusted before being used to make predictions if Ag_s has previous experience with the agents who give the recommendations. Suppose agent Ag_x gives recommendation on the item O_n as $r_{x,n} = j$. As discussed in the **Case (2)** in Section 2.1, given Ag_x's recommendation j, Ag_s predicts the new item's rating as i with a probability $\frac{n_{ij}}{N} / \frac{C_j}{N} = n_{ij}/C_v$. Or, to put it in another way, Ag_x's recommendation $r_{x,n} = j$ is adjusted to $Pr_{s,x,n} = i$ with a probability n_{ij}/C_v. If Ag_s has no previous experience with Ag_x, Ag_x's recommendation $r_{x,n} = j$ is used directly without adjustment.

It is possible that recommendation discovery does not collect any recommendation, e.g. no agent in the community gives recommendation, or the upper limit U is reached before any recommendation has been discovered. In this case, Ag_s would resort to its direct neighbors to make the predictions. It selects a *pseudo-recommendation* as $Pr_{s,x,n} = i$ with a probability R_i/N for each direct neighbor Ag_x. This is in fact the **Case (1)** in Section 2.1. With the *pseudo-recommendation*, agent Ag_s is always able to make predictions even on items that are newly-introduced and no agents have rated before.

Ag_s then predicts the new item O_n's rating $Pr_{s,n}$ as:

$$Pr_{s,n} = \frac{\sum_{Ag_x \in \mathcal{A}_{s,n}} Tr_{s,x} * Pr_{s,x,n}}{\sum_{Ag_x \in \mathcal{A}_{s,n}} Tr_{s,x}} \tag{3}$$

Here $\mathcal{A}_{s,n}$ is the set of agents that give recommendations (or pseudo-recommendations when no recommendation is available) on item O_n for agent Ag_s. $Pr_{s,x,n}$ is the adjusted recommendation or pseudo-recommendation given by agent Ag_x on item O_n for agent Ag_s. Each agent in $\mathcal{A}_{s,n}$ is assigned a weight which equals to its trustworthiness $Tr_{s,x}$.

4 Experimental Study

4.1 Experiment Setup

Experiments have been conducted to study the proposed trust-based community's performance against the traditional similarity-based community. A movie recommendations community is setup, which is simulated based on the MovieLens dataset[7]. The dataset contains 100,000 ratings of 5-point scale of 1682 items by 943 agents. Each rating entry has 4 main fields: the ID of the agent who gives the rating, the ID of the item, the rating, and the timestamp when the rating was given. The whole dataset is split into two subsets with 80,000 and 20,000 ratings respectively. The latter subset is used to simulate the running of community. Ratings in the other subset are taken as agents' experience with the items before joining the community. Those ratings are used in the ratings exchange when agents join the community, which facilitates the direct neighbors selection.

For the purpose of comparison with traditional similarity-based community,

[7]MovieLens dataset is publicly available at http://www.cs.umn.edu/research/GroupLens/data/.

a similarity-based community with same dataset is also setup, which employs Pearson correlation efficient as the similarity metric. To make the comparison fair, a similar mechanism of recommendation collection with similarity propagation is also implemented. As reported in [10], agents with negative similarity are ignored when making prediction in the similarity-based community.

The simulation is conducted in a leave-one-out manner. In each single cycle of the simulation, one rating is taken out by the ascending order of the timestamp. In each cycle, the real rating is masked intentionally. Then the active agent (as indicated by the agent ID field in the entry) predicts the rating of the item. The agent will only browse the item when the prediction is higher than a pre-defined threshold. In the simulated community, it is assumed that the threshold is always 0. Thus the agents opt to browse all items encountered. The predictions are then compared against the masked real ratings in order to measure the performance of the recommendation community.

The first group of the experiments is to study the prediction accuracy and coverage without the presence of noisy ratings. The experiments have two parameters to tune: the number of direct neighbors that each agent maintains (D), and the upper limit of referral chain length (U). We tune the value of these two parameters to study accuracy and coverage in different settings, the values of D and U are tuned to be: $D \in \{2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 30, 40, 50, 60, 70, 80\}$, and $U \in [3, 8]$.

The second group of experiments is to study robustness in the presence of noisy ratings. In this paper, only one type of noisy rating is studied, i.e. a number of malicious agents intentionally give noisy ratings in order to manipulate other agents' prediction to certain target ratings. There are two forms of noisy ratings, with different target ratings: *nuke* whose target rating is the minimum of all the possible ratings, and *push* whose target rating is the maximum of all the possible ratings. In the simulated community, the target rating of *nuke* noisy rating is 1, and the target rating of *push* noisy rating is 5. The community's robustness in presence of different ratios of agents giving noisy ratings is studied. The ratios of agents giving noisy ratings are tuned to be 25%, 50%, 75%, and 100%.

Experiments with the same settings have been run 3 times, and the average of the results derived in all the 3 experiments is taken as the final result.

4.2 Measurement of the Results

Mean Absolute Error (MAE) is usually applied to measure the prediction accuracy, which is basically the average absolute difference between the predictions and the (masked) real ratings over all items and all agents. However, since each agent makes predictions individually with only local computation in the trust-based community, it is more reasonable to measure each agent's prediction accuracy separately. For this reason, we employ a different interpretation of the MAE metric, which first calculated the mean prediction error for each agent and then averaged all agents' mean errors. A smaller value of MAE implies a higher accuracy.

Coverage is measured as the percentage of the items on which that the agents are able to predict ratings.

The robustness of the community in the presence of noisy ratings is measured by the *power of attack* (POA) [14]. Let O_t be the targeted item of the malicious agents, and Ω be the set of agents who predicted item O_t's rating. The POA for a particular type of noisy rating ATK is given by:

$$POA(ATK) = \frac{\sum\limits_{Ag_i \in Ag} \frac{\sum\limits_{O_j \in \Omega_i} (|r_{target} - r_{i,j}| - |r_{target} - r'_{i,j}|)}{\|\Omega_i\|}}{\|Ag\|}.$$

Ag is the set of all agents, Ag_i is one agent in set Ag. Ω_i denotes the set of items rated by Ag_i, and O_j is one of the item in set Ω_i. $r'_{i,j}$ and $r_{i,j}$ are predictions on item O_j derived by Ag_i with and without the presence of noisy ratings respectively. r_{target} is the target rating of the malicious agents's noisy ratings, e.g. $r_{target} = 1$ for nuke and $r_{target} = 5$ for push in the simulated community. POA measures the extent that agents' predictions are manipulated towards the target rating by the malicious agents. A positive value of POA means that the predictions have been manipulated towards the target rating. Otherwise, POA gives a negative value, and POA is 0 if there is no difference in predictions derived with and without attack.

4.3 Results: Accuracy and Coverage

First of all, we study the coverage. Since the trust metric is computable on most peers, in most cases agents are able to find trustworthy agents who can give recommendations. Moreover, *pseudo-recommendation* enables agents to make predictions for items that no other agents have rated before. Consequently, the coverage achieved with the trust-based community is always $> 99\%$. There is only a tiny portion of agents that are not able to make predictions. This is because they have not co-rated any items with any other agent, which makes them to be excluded from community. In contrast, the highest coverage achievable with traditional similarity-based community is $< 80\%$. The coverage achieved with the similarity-based CF is shown in Figure 2. It is thus shown that the trust-based community does improve the coverage of the recommendation.

Then the prediction accuracy is studied. Prediction accuracy (i.e. MAE)

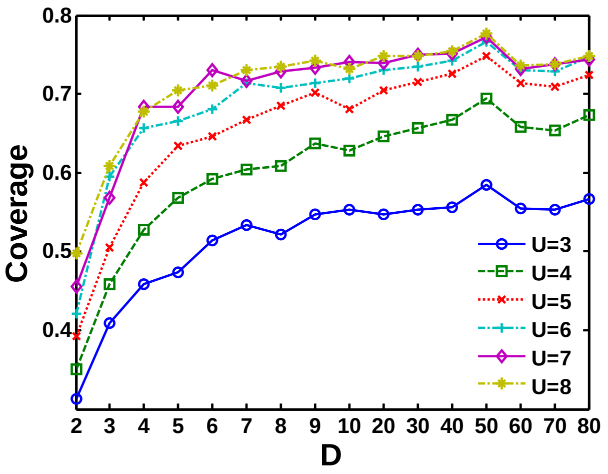


Figure 2: Coverage of similarity-based community

achieved with trust-based community in different setting of D and U are plotted in Figure 3. The comparison of MAE achieved by trust-based and similarity-based community in different settings of D and U is also shown in Figure 4(a)-4(f). Note that similarity-based CF's accuracy is only calculated over the items that agents are able to give predictions.

It is observed from Figure 4(a)-4(f) that the trust-based community does enhance the prediction accuracy, though the improvement over the traditional similarity-based community is not significant (usually $\Delta MAE < 0.05$). It is also observed that generally the improvement is more significant when the values of D and U are larger, e.g. $D > 10$ and $U > 5$. One of the contributing reasons is that an agent is able to discover more agents that give recommendations with the increase of the value of D and U. The average numbers of agents giving recommendations for a single round of prediction in different settings are listed in Table 3. Another reason is that the predictions are made as weighted sum of other agents' recommendations. Consequently, the increase of D and U can improve the prediction accuracy since if a larger number of agents with higher trustworthiness are discovered to give recommendations.

However, as Figure 3 shows, the accuracy of the trust-based community does not increase significantly when $D > 10$ and $U > 5$. The total number of the referrals that an agent encounters during the recommendation discovery is approximately D^U. When $D > 10$ and $U > 5$, $D^U >> N = 943$, a peer can almost reach all other peers. Most of the referrals are not visited in order to avoid duplicated recommendation, though there are a huge number of referrals returned. Hence, the increase in the number of referral with the increase in the

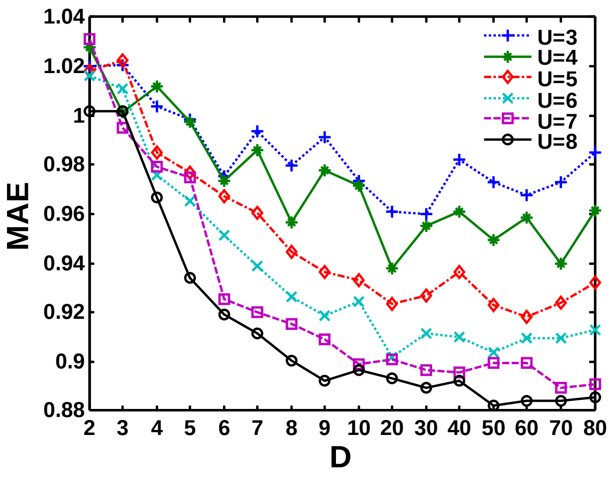


Figure 3: MAE achieved by the trust-based community

D	U	Trust-based	Similarity-based
$D = 5$	$U = 2$	0.68	0.79
	$U = 5$	1.98	2.13
	$U = 8$	5.00	5.89
$D = 50$	$U = 2$	1.54	1.01
	$U = 5$	7.37	3.70
	$U = 8$	34.00	12.26
$D = 100$	$U = 2$	1.61	1.03
	$U = 5$	8.68	4.11
	$U = 8$	37.63	16.44

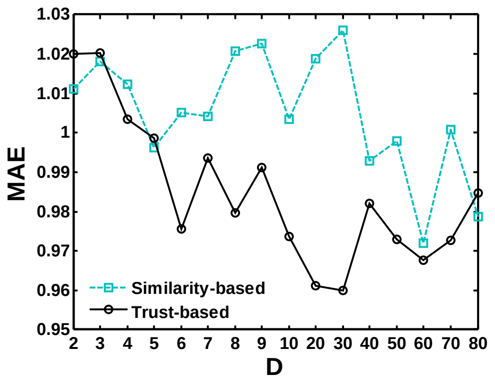
Table 3: Average number of agents giving recommendations

values of D and U does not increase the number of agents whose recommendations are taken into account in making predictions when $D > 10$ and $U > 5$. Consequently, the improvement produced by the increase in the values of D and U becomes insignificant when $D > 10$ and $U > 5$ since the predictions are made as weighted average of recommendations.

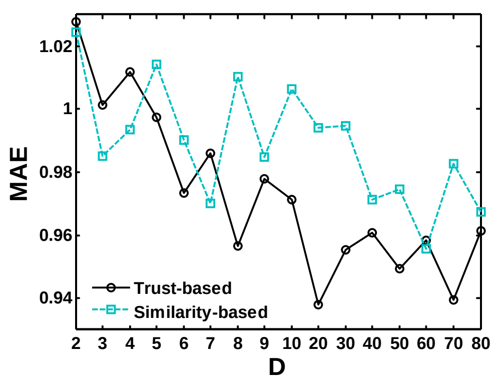
4.4 Results: Robustness

Inspired by the results of the first group of experiments, we choose $D = 10$ and $U = 3$ to run the second group of experiments. Robustness (measured by POA) in the presence of different ratios of malicious agent giving noisy ratings is shown in Table 4.

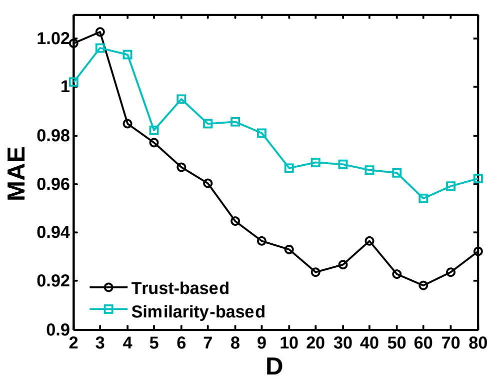
It is observed in Table 4 that with more maliciou agents giving noisy ratings, the predictions are biased toward the target ratings closer. Nevertheless, the trust-based community is more robust than its similarity-based counterpart



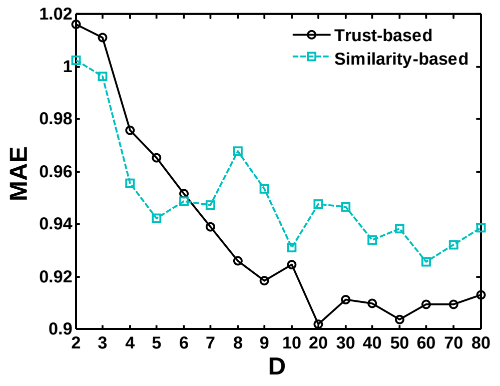
(a) Comparison of MAE ($U = 3$)



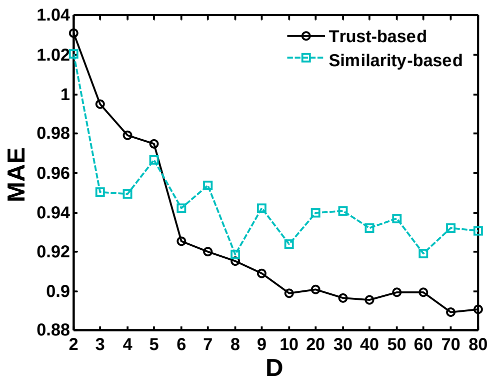
(b) Comparison of MAE ($U = 4$)



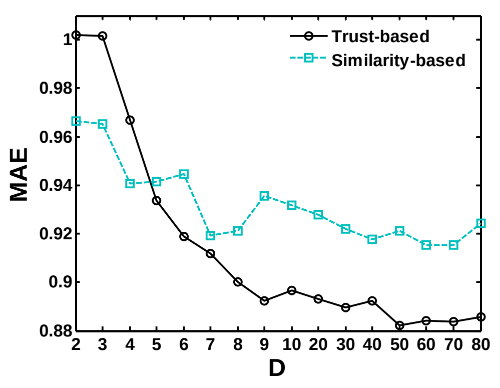
(c) Comparison of MAE ($U = 5$)



(d) Comparison of MAE ($U = 6$)



(e) Comparison of MAE ($U = 7$)



(f) Comparison of MAE ($U = 8$)

Figure 4: Coverage and Accuracy of the Similarity-based CF and the proposed method

Noisy rating ratios	POA (Push)		POA (Nuke)	
	Trust-based	Similarity-based	Trust-based	Similarity-based
25%	0.0688	0.0410	0.0722	0.0238
50%	0.1105	0.1880	0.1430	0.2145
75%	0.1375	0.2020	0.5305	0.6408
100%	0.1788	0.2211	0.6730	0.7617

Table 4: POA w.r.t. different ratios of agents giving noisy ratings

in the presence of noisy ratings. Since the predictions are made as weighted average of other agents' recommendations, a high weight is necessary for the malicious agent to successfully bias the predictions. In order to achieve high trustworthiness (i.e. weights in the trust-based community), an agent needs to contribute more recommendations to Ag_s in order to clear as many columns with dummy experience as possible. However, sharing recommendations across many columns does not necessarily increase the trustworthy score. If the non-zero cells distribute over different rows but within same column in Table 1, the uncertainty would raise too, which leads to the decrease of the trustworthiness. Consequently, it takes more effort for malicious agents to get high trustworthiness before they can successfully manipulate Ag_s's predictions. This helps to mitigate the malicious agents' influence. In contrast, agents are easier to achieve high similarities (i.e. weights in the similarity-based community) with high probability if they intentionally give recommendations on a small number of items.

Moreover, an agent does not make the prediction the new items' ratings by carrying out weighted average on the collected recommendations directly in the trust-based community. Instead, a recommendation is adjusted according to the agent's previous experience (if any) with the other agent who gives the recommendation. Even if a malicious agent manages to achieve high trustworthiness, the influence of its noisy ratings can still be mitigated by the recommendation adjustment. In contrast, the predictions are made with the recommendations directly in the similarity-based community. As a result, the trust-based community is more robust in the presence of noisy ratings than its the similarity-based community.

5 Related work

Work has been reported to introduce trust to overcome the limitations of traditional similarity-based recommendation as discussed in Section 1, e.g. [12, 15, 16]. In [12], it is shown that introduction of trust does improve the prediction accuracy and coverage of the recommendation community. Pouwelse. et al propose to establish trust through identifying real world friends [16]. This is not practical, however, since pseudonyms are usually used in cyberspace, and it is not easy to connect pseudonyms with real identities. Papagelis et al. focus on developing a computational model to establish trust between agents by exploiting the transitive nature of the trust [15].

All of these assume that an agent already has the ability to determine the degrees of trust it should place on others, and do not make clear how an agent can determine the degrees of trust it should place on the other agents. This paper goes beyond existing work in that it proposes a practical trust metrics, with which agents quantify the degrees of trust it places on others based on previous direct experience.

6 Conclusions and Future work

In this paper, we have proposed a trust-based recommendation community which incorporates trust into the domain of item recommendation. A trust metric is designed to quantify the degree of trust an agent should place on other agents. Then agents make prediction for new items' rating as a weighted average of trustworthy agents' recommendations, with agents' trustworthiness as weights. In order to mitigate the influence of noisy ratings, the other agents' recommendations are adjusted before being used to make predictions. The trust-based community possesses many advantages, as shown by the experimental results:

(1) the trust-based community is able to give more accurate predictions than the similarity-based community,
(2) the trust-based community manages to achieves a coverage that is much higher than the one achieved with similarity-based community,
(3) and the trust-based community is more robust in the presence of noisy ratings than the similarity-based community.

The prerequisite of trust-based community is that agents' experience with the items can be represented by discrete ratings. This prerequisite is rational and practical as most of the real world scenarios apply discrete ratings, e.g. MovieLens and Epinions. In scenarios where agents' experience are represented as continuous ratings, the trust-based community is still applicable if the continuous ratings are stratified into discrete bins.

Another concern is the storage overhead of the trust-based community. To compute the trust metrics, each agent needs to maintain and compile its previous experience with other agents in a table. Generally, the number of possible discrete ratings is usually less than 7, e.g. there are only 5 ratings available in both MovieLens and Epinions. In fact, studies have already shown that reliability of ratings collected does not increase substantially if the number of ratings is more than 7 [7]. Hence tables used by agents to compile the experience require small storage.

Our future work includes the following. Currently, the direct neighbors chosen when an agent joins the community are not changed during its whole life cycle in the community. It is possible that some direct neighbors become less trustworthy with more experience accumulated. In this case, those less trustworthy direct neighbors should be replaced by other agents that are more trustworthy. The performance of the trust-based community with neighbor replacement is to be further investigated.

In current work, we only study the trust in the context of opinion exchange [4] within the domain of item recommendation. In future work, trust in multi context within the domain of item recommendation is to be further studied, e.g. in the contexts of opinion exchange and reputation exchange [4, 21].

References

[1] Epinion.com FAQ. http://www.epinions.com/help/faq/?show=faq_wot.

[2] B. Barber. *The logic and limits of trust.* Rutgers University Press, 1983.

[3] G. Carenini. User-specific decision-theoretic accuracy metrics for collaborative filtering. In *Proceedings of Beyond Personalization 2005: Workshop on the Next Stage of Recommender Systems Research*, 2005.

[4] K. K. Fullam, T. B. Klos, G. Muller, J. Sabater, A. Schlosser, Z. Topol, K. S. Barber, J. S. Rosenschein, L. Vercouter, and M. Voss. A specification of the Agent Reputation and Trust (ART) testbed: experimentation and competition for trust in agent societies. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005)*, pages 512–518, 2005.

[5] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval Journal*, 4(2):133–151, 2001.

[6] L. A. Goodman and W. H. Kruskal. Measures of association for cross classifications. *Journal of the American Statistical Association*, 49(268):732–764, 1954.

[7] C. Jensen, J. Davis, and S. Farnham. Finding others online: reputation systems for social online spaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 447–454, 2002.

[8] A. Jøsang, E. Grayy, and M. Kinatederz. Simplification and analysis of transitive trust networks. *Web Intelligence and Agent Systems*, To Appear, 2005.

[9] A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support System*, To appear, 2006.

[10] S. K. Lam and J. Riedl. Shilling recommender systems for fun and profit. In *Proceedings of the 13th international conference on World Wide Web*, pages 393–402, 2004.

[11] S. P. Marsh. *Formalising Trust as a Computational Concept.* PhD thesis, Department of Mathematics and Computer Science, University of Stirling, 1994.

[12] P. Massa and P. Avesani. Trust-aware collaborative filtering for recommender systems. In *Proceedings of International Conference on Cooperative Information Systems 2004*, 2004.

[13] L. Mui. *Computational Models of Trust and Reputation: Agents, Evolutionary Games, and Social Networks.* PhD thesis, Department of Electrical Engineering and Computer Science, MIT, 2003.

[14] M. O'Mahony, N. Hurley, N. Kushmerick, and G. Silvestre. Collaborative recommendation: A robustness analysis. *ACM Transaction on Internet Technology*, 4(4):344–377, 2004.

[15] M. Papagelis, D. Plexousakis, and T. Kutsuras. Alleviating the sparsity problem of collaborative filtering using trust inferences. In *Proceedings of iTrust 2005*, pages 224–239, 2005.

[16] J. Pouwelse, M. van Slobbe, J. Wang, and H. Sips. P2P-based PVR recommendation using friends, taste buddies and superpeers. In *Proceedings of Beyond Personalization 2005: Workshop on the Next Stage of Recommender Systems Research*, Jan 2005.

[17] S. D. Ramchurn, D. Huynh, and N. R. Jennings. Trust in multi-agent systems. *The Knowledge Engineering Review*, 19(1):1–25, 2004.

[18] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, 1994.

[19] M. Richardson, R. Agrawal, and P. Domingos. Trust management for the semantic web. In *Proceedings of 2nd International Semantic Web Conference*, pages 351–368, 2003.

[20] J. Sabater-Mir. *Trust and Reputation for Agent Societies*. PhD thesis, Institut d'Investigació en Intel.ligència Artificial, 2003.

[21] W. T. L. Teacy, J. Patel, N. R. Jennings, and M. Luck. Coping with inaccurate reputation sources: Experimental analysis of a probabilistic trust model. In *Proceedings of The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005)*, pages 997–1004, 2005.