Distributed Collaborative Filtering for Robust Recommendations Against Shilling Attacks

Ae-Ttie Ji[1], Cheol Yeon[1], Heung-Nam Kim[1], and Geun-Sik Jo[2]

[1] Intelligent E-Commerce Systems Laboratory,
Department of Computer Science & Information Engineering, Inha University
{aerry13,entireboy,nami}@eslab.inha.ac.kr
[2] School of Computer Science & Engineering, Inha University,
253 Yonghyun-dong, Incheon, Korea 402-751
gsjo@inha.ac.kr

Abstract. Recommender systems enable a user to decide which information is interesting and valuable in our world of information overload. Collaborative Filtering (CF), one of the most successful technologies in recommender systems suffers from improper use of personal information and the incredibility of recommendations. To deal with these issues, we have been focusing on the trust relationships between individuals, i.e. *web of trust*, especially for protecting the recommender system against profile injection attack. Based on trust propagation scheme, we proposed *TCFMA* architecture which is added agent-based scheme obtaining attack resistance property as well as improving the efficiency of distributed computing. In *web of trust*, users' personal agents find a unique migration path made up of latent neighborhoods and reduce search scope to a reasonable level for mobile agents by using the *Advogato* algorithm. The experimental evaluation on *Epinions.com* datasets shows that the proposed method brings significant advantages in terms of dealing with profile injection attack without any loss of prediction quality.

1 Introduction

In a flood of information, a recommender system helps users to decide which items are most valuable and interesting to them. Collaborative Filtering (CF), one of the most successful technologies in recommender systems, has been applied to numerous commercial recommender systems. Even though they are popular, there are problems of improper use of personal information and the incredibility of recommendations especially in centralized CF recommender systems where all ratings by users are owned by system providers [12]. These problems can be partially improved by a distributed personal recommender, but the distributed systems might be vulnerable to a shilling attack, i.e. profile injection attack as similar as centralized ones. That is, an attacker may make many profiles with biased ratings with a malicious intent to influence the recommendations [1, 13, 14]. Because most of recommender systems are open Web services, a malicious attacker can easily inject manipulated profiles [13, 14]. One effective way to protect the system against such an attack is to build *web of trust* among the individuals in order to use only the profiles of trustworthy users [8, 11].

Z. Kobti and D. Wu (Eds.): Canadian AI 2007, LNAI 4509, pp. 14–25, 2007.

In this paper, we propose *TCFMA (Trust-based Collaborative Filtering with Mobile Agents)* architecture used a distributed CF method in peer-to-peer network using *web of trust* in order to effectively offer the corresponding user trustworthy recommendations. By using the *Advogato* trust metric [4], we propagate the trust information for overcoming the sparseness of *web of trust* as well as obtaining resistance from attacks by the malicious users [3, 4]. In addition, we employ mobile agents to increase the efficiency of distributed computing.

The rest of this paper is organized as follows: The next section contains a brief overview of some related work. In Section 3, we describe our proposed method in detail. Then, the performance evaluation compared with other P2P approaches is presented in Section 4. Finally, Section 5 draws some conclusion of this paper with a discussion of future work.

2 Backgrounds and Related Works

A Robustness Analysis of Collaborative Filtering. Most of Web-based CF recommender systems employ profiles which are made by anonymous unauthenticated users. That is, the systems can be vulnerable to manipulation due to profiles which are built and injected by an attacker. Many recent researches have shown that commonly used CF algorithms are significantly affected by modest attacks [14]. There are two formal types of profile injection attack that can be defined according to the intent of an attacker: a push attack and a nuke attack [13, 14]. The former makes particular items promoted and the latter makes them demoted in order to be more or less recommended. In the case of centralized CF systems where all profiles are owned by a merchant, a system provider can be a "push" attacker. Because a merchant always wants a customer to buy an item that maximizes profits. *PocketLens* [1], which we benchmark, described that this concern can be met by a distributed personal recommender because only a user can owns and controls his or her own profiles. Based on the credibility of recommendations, diverse distributed recommender system architectures and an incremental computing algorithm applicable to those architectures were proposed through this work [1].

Trust in Recommender Systems. Even though the distributed recommender can partially improve the effects of profile injection attacks from a system provider, it is still not safe from an anonymous attacker [1]. In order to overcome the vulnerabilities of CF systems to attacks, a number of recent studies focus on the notion of "trust" in recommendation [14]. Calculating explicit trust and reputation values of users or eliciting trust relationships between users, a system employs only the owner's profile guaranteed identity and trustworthiness [3, 4, 6, 8, 15]. In a more global view, "trust" of a recommender system has been studied in terms of automated attack detection schemes and robustness of recommendation algorithms in the face of malicious attacks [3, 4, 13, 14, 17]. Another perspective about trust focuses on correlation between trust and user similarity. Recent research of Sinha et al. confirms the fact that people tend to prefer recommendations from friends and acquaintances to those from online recommender systems [6]. Moreover, Ziegler et al. shows that people's preferences can be more similar to preferences of trusted users than those of arbitrary users [7]. That is, using

"trust" for recommender system can be effective in terms of attack-resistance and recommendation quality. However, *web of trust* tend to be sparse; so a mechanism of trust propagation is required [2]. R.Guha et al. define four atomic trust propagation methods that can be applied repeatedly to obtain a final matrix with trust and distrust information [2]. Ziegler et al. protect against massive attacks from malicious users propagating trust effectively in a social network by proposing the *Appleseed* algorithm based on the *Advogato* trust metric [3].

3 Trust-Based Collaborative Filtering for P2P Networks

Fig. 1 illustrates a brief overview of our proposed system with three steps; Firstly, a user agent finds the migration path along which a mobile agent migrates in *web of trust*. Along the path, the mobile agent finds trusted neighborhoods. Then the user agent builds a similarity model incrementally for its user by using the information that the mobile agent gathers from neighbors. Finally, the user agent provides its user with recommendations and updates the model with his or her feedback.

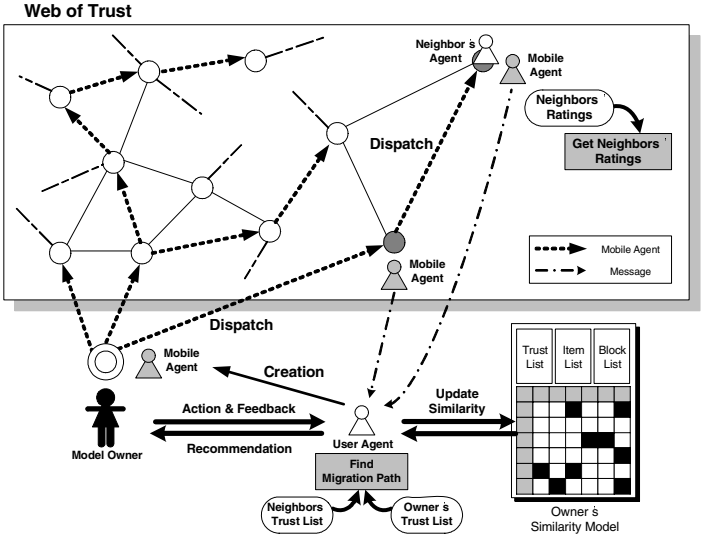

Fig. 1. Overview of trust-based collaborative filtering with mobile agents

3.1 Trust-Based User Selection

In this section we describe a scheme to finding a unique migration path, which consists of the trusted users, for mobile agent of a target user. Before describing the algorithms, some definitions of the notations used herein are introduced.

Let us assume a peer-to-peer network where each user *trusts* other users. The set of $\{TRUST_{Px}\}$, that is the list of users trusted by each user P_X, simply presented in the

Table 1. The meaning of notations

P_X	Arbitrary user included in web of trust
P_O	Target user, i.e. similarity model owner
P_C	Current user who P_O's mobile agent is visiting at the moment
$\{TRUST_{Px}\}$	List of users who are trusted by P_X
$\{BLOCK_{Px}\}$	List of users who are distrusted by P_X
$\{ITEMS_{Px}\}$	List of <*item, rating*> pairs, i.e. items which P_X already has expressed his or her own opinion and these preference ratings.
$\{PATH_{Px}\}$	Migration path which P_X's mobile agent migrates along
$AGENT_{Px}$	Personal agent of P_X
$AGENT^M_{Px}$	Mobile agent of P_X

same fashion as shown in *Advogato*[1] and *Epinions.com*[2]. *Web of trust* is constructed in the form of a bidirectional graph based on the set of $\{TRUST_{Px}\}$. In contrast with other systems, only a personal agent $AGENT_{Px}$ includes <*item, rating*> pairs, i.e. items in which P_X is interested and preference ratings for these items, as listed in $\{ITEMS_{Px}\}$.

$AGENT_{Px}$ finds the migration path $\{PATH_{Px}\}$ that includes users trusted by P_X for a mobile agent $AGENT^M_{Px}$. The *Advogato maximum flow algorithm* is exploited to obtain the migration path for a mobile agent. This algorithm, inspired by the *Ford-Fulkerson maximum flow algorithm*, was used to discover which users are trusted by credible members of an online community and which are not [3, 4, 15]. Because the bidirectional graph of trusts is restructured to form a tree-structure in the process of finding maximum flow through the edges, the algorithm makes it possible to find a unique migration path and to reduce search scope to reasonable levels for mobile agent.

The procedure to find $\{PATH_{Px}\}$ is as follows: Assume that P_O, who is the target user, is the trust source. The capacities C are assigned to every user in web based on the shortest-path distance from the source to P_X. The capacity of the source, which can be optionally chosen by P_O, is based on the number of all users expected to be visited and whose information is used for recommendations. Each successive level has a capacity equal to that of the preceding level L divided by the average number of trust edges extending from nodes of L. In order to apply the *Ford-Fulkerson maximum flow algorithm* to this single-source/multiple-target graph with capacity-constrained nodes, it has to be restructured to a single-source/single-target one with capacity-constrained edges rather than nodes. Each node P_X is split into P_X^+ and P_X^-, and the capacity that is the original $C(P_X)$ minus one is assigned to an edge between them. Then, a unit capacity edge is added from P_X^- to a virtual single target node. The original edge from P_X to P_Y is represented as the one from P_X^+ to P_Y^- with infinite capacity. $AGENT_{Po}$ applies the algorithm

[1] *Advogato* http://www.advogato.org/
[2] *Epinions.com* http://www.epinions.com/

to this converted graph tracing the shortest paths to the target first and adds the nodes reached by network flow to $\{PATH_{Po}\}$ [3, 4, 15].

Owing to the *bottleneck property* proposed as a common feature of attack-resistance trust metrics in [15], *Advogato* algorithm is useful to make profile injection attacks take no effects. The *bottleneck property* is that "the total trust quantity accorded to an $s \rightarrow t$ edge is not significantly affected by changes to the successors of t [3, 15]," i.e., the number of biased nodes accepted depends only on the number of precedence nodes, not on the number of biased ones. Assume that there is an attacker who intents to promote or demote a particular item, or just to make the overall system function poorly. Even though he builds many profiles (manipulated nodes t) with fraud ratings coincided with his purpose, he cannot make s trust t manipulated by the attacker. Therefore, the amount of trust accorded to t dose not increase even though the attacker injects more nodes [3, 4]. For this reason, even the least of profiles that make the attack succeeded is not included in the process of collaboration. More about attack-resistance properties of various trust metrics are discussed in detail in [15] and [17], it has been claimed that *PageRank* [16] possesses *bottleneck property* like *Advogato*.

3.2 Incremental Model Building

Recommender algorithm can be characterized by the neighbors they choose for each user, the model they build based on those neighbors, and the way they use the model to form recommendations [1]. In our research, the neighbors of target user P_O are chosen from the users included in $\{PATH_{Po}\}$. P_O's personal agent $AGENT_{Po}$ creates a mobile agent, $AGENT^M_{Po}$, to find neighbors and build a similarity model based on them incrementally. $AGENT^M_{Po}$ involves only the least amount of information for model building; $\{PATH_{Po}\}$ and $\{ITEMS_{Po}\}$. Searching for the users in $\{PATH_{Po}\}$ is done by the *Depth First Search* algorithm. Supposing that P_C is a currently visited neighbor, $AGENT^M_{Po}$ traces the path recursively until no users exist in $\{PATH_{Po}\} \cap \{TRUST_{Pc}\}$. Then $AGENT^M_{Po}$ is disposed of from the last node after visiting all users in $\{PATH_{Po}\}$.

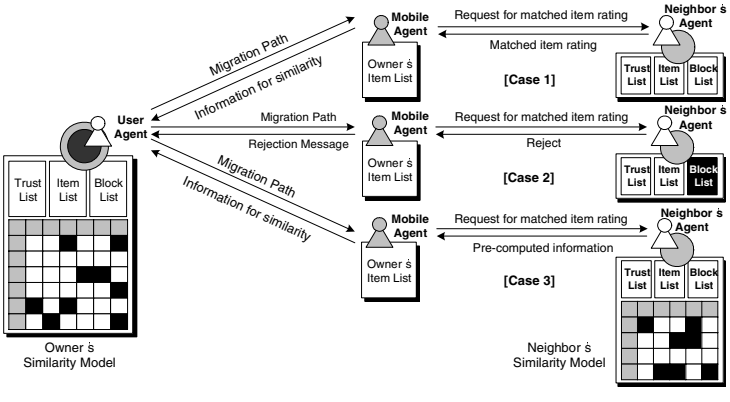

Fig. 2. Agents' tasks in each case

The similarity model is composed of a set of similarities between pairs of items and represented as a matrix [1]. For incremental computation of similarity relationships, each agent does as follows;

1. $AGENT^{M}_{Po}$ identifies IO and IP that are $\{ITEMS_{Pc}\} \cap \{ITEMS_{Po}\}$ and $\{ITEMS_{Pc}\} - \{ITEMS_{Po}\}$ respectively, by communicating with a neighbor's agent $AGENT_{Pc}$.
2. For each pair (IO_i, IP_j), which is $IO_i \in IO$ and $IP_j \in IP$, $AGENT^{M}_{Po}$ calculates values shown in Equation 1 and send the values to its own agent $AGENT_{Po}$.

$$\begin{aligned} W_{IO_i,IP_j} &= (Rating_{P_c,IO_i} \times Rating_{P_c,IP_j}) \\ W_{IO_i,IO_i} &= (Rating_{P_c,IO_i})^2 \\ W_{IP_j,IP_j} &= (Rating_{P_c,IP_j})^2 \end{aligned} \tag{1}$$

3. $AGENT_{Po}$ adds up these values incrementally until $AGENT^{M}_{Po}$ sends the values of all users in $\{PATH_{Po}\}$ except for those which don't have IO_i.

$$\begin{aligned} W_{Numer} &= W_{Numer} + W_{IO_i,IP_j} \\ W_{Denom1} &= W_{Denom1} + W_{IO_i,IO_i} \\ W_{Denom2} &= W_{Denom2} + W_{IP_j,IP_j} \end{aligned} \tag{2}$$

4. $AGENT_{Po}$ calculates the similarity of item pair (IO_i, IP_j).

$$sim(IO_i, IP_j) = \frac{\vec{IO_i} \bullet \vec{IP_j}}{|\vec{IO_i}| \times |\vec{IP_j}|} = \frac{W_{Numer}}{\sqrt{W_{Denom1}}\sqrt{W_{Denom2}}} \tag{3}$$

The above-mentioned procedure involving cosine-similarity metrics can have different versions simply by modifying the second process. For instance, in order to use adjusted cosine similarities between two items as a similarity metric [5], Equ.(1), (2) are modified as

$$\begin{aligned} W'_{IO_i,IP_j} &= (Rating_{P_c,IO_i} - AvgRating_{P_c}) \times (Rating_{P_c,IP_j} - AvgRating_{P_c}) \\ W'_{IO_i,IO_i} &= (Rating_{P_c,IO_i} - AvgRating_{P_c})^2 \\ W'_{IP_j,IP_j} &= (Rating_{P_c,IP_j} - AvgRating_{P_c})^2 \end{aligned} \tag{4}$$

In general cases, $AGENT_{Po}$ can update a similarity model owned by P_O incrementally according to the above procedures. However, assume that a neighbor P_C has the list $\{BLOCK_{Pc}\}$, the list of the users whom P_C distrusts, and P_O is included in this list. Not trusting P_O, P_C may not want to be open with P_O about his own information. In this case, $AGENT_{Pc}$ rejects the request for information from $AGENT^{M}_{Po}$.

If the similarity model has already been built for P_C to get recommendation, $AGENT^{M}_{Po}$ has no need to visit P_C's successive nodes and can get their values from this model. This model has been built based on $\{PATH_{Pc}\}$ setting P_C as a source. $\{PATH_{Pc}\}$ is much more likely to include the same users as $\{PATH_{Po}\}$; the closer the distance from the source to P_C, the more similar it is. In this case, P_O's similarity model might include the information of more neighbors than the users whom $AGENT^{M}_{Po}$ intended to visit at the beginning. Moreover, these users included in P_C's

model might overlap with the users who $AGENT^{M}{}_{Po}$ has already visited or is supposed to visit hereafter. However, "overlapping users" means that they are trusted by more preceding level users, which can have a positive influence on recommendations by reflecting their opinions more. By pruning the successive nodes, tracing costs can be decreased drastically.

3.3 Propagating User Feedback

Based on this similarity model, $AGENT_{Po}$ provides recommendations to P_O. Simply, the particular items, which obtain either the highest averages of each column or the highest prediction values, are recommended. Explicit prediction values of user P_O for item IP_j can be computed by the weighed sum of P_O's ratings about IO_i using the similarity $sim(IO_i, IP_j)$ as the weight and defined as Equ.(5) [1, 5].

$$p\_rating_{P_o,IP_j} = \frac{\sum_{IO_i \in IO}\{sim(IO_i, IP_j) \times Rating_{P_o,IO_i}\}}{\sum_{IO_i \in IO}\left|sim(IO_i, IP_j)\right|} \tag{5}$$

The idea is that the average rating of items that are similar to the selected item is a good estimate of the rating for the selected item [1].

One of the principle issues that we have to consider in a model-based approach is how the updated information can be reflected in original models during the term when they have not been re-built yet. When item IP_k is recommended to P_O, she can express her preference for this item as a rating. Whenever the P_O gives feedback, $AGENT_{Po}$ deletes IP_k's column from her model and adds it to $\{ITEMS_{Po}\}$ because it now has its own rating. Updated information that is, $\{ITEMS_{Po}\}$ including a pair $<IP_k$, *P_O's rating about* $IP_k>$, is propagated to personal agents of users in $\{TRUST_{Po}\}$. From P_C's point of view who is given this information by $AGENT_{Po}$, item IP_k has not been included in $\{ITEMS_{Po}\}$ before, so the similarities between IP_k and items in $\{ITEMS_{Po}\} \cap \{ITEMS_{Pc}\}$ cannot be included in P_C's model.

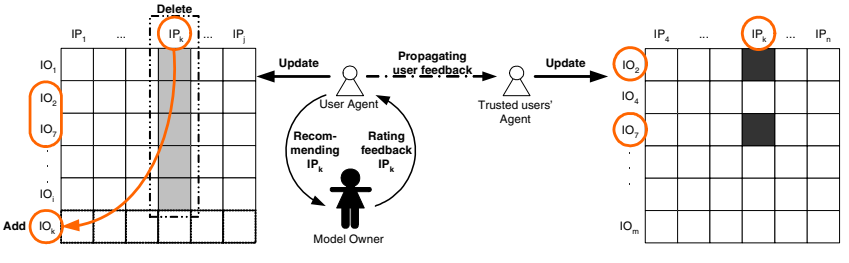

Fig. 3. Recommendations and propagation user's feedback

$AGENT_{Pc}$ now can compute similarities between them, update the model in patches; incremental updating can be achieved by propagating it to the users in P_C's own list $\{TRUST_{Pc}\}$. Fig.3 illustrates feedback and update process.

4 Experimental Results

In this section we present the results of applying *TCFMA* method for recommendation in a peer-to-peer environment. The prototype system is implemented using *IBM Aglet* Software with *JDK1.4.2* [10].

4.1 Data Sets and Evaluation Metrics

Epinions.com is an online community where users can review various items and rate them on a scale of 1 to 5. Judging whether the reviews of others are helpful to users themselves or not, users can express trust or distrust of these reviewers. We collected the dataset by crawling the *Epinions.com* site in May 2006. The collected dataset was too sparse to be used for experiments, so we selected a dataset including users who had rated at least 5 items and expressed a trust opinion of at least 25 users. In addition, the items had been rated by at least 10 users, i.e. the dataset contained 121,862 ratings for 2,955 items and 216,490 trust information presented by 4,751 users. The sparsity level of our dataset is $1-(121{,}862/4751 \times 2955)$, which is 0.9913. Then, this dataset was divided into two parts; training set contains all of each user's ratings except one rating used for testing. Testing set contains the only one rating for each user.

In order to measure performance, *Mean Absolute Error*, which is used as a measure of how accurate prediction of user's rating for an item can be, was performed [1, 5, 8]. *MAE* of all users in the testing set is defined as:

$$MAE = \frac{\sum_{i=1}^{M} | p\_rating_i - a\_rating_i |}{M} \tag{6}$$

where M is a list of all items and $<a\_rating_i , p\_rating_i>$ is the actual/predicted rating pairs of each user in the testing set.

Another metric *Absolute Prediction Shift* measured the distortion of prediction occurring due to an attack. While *p_rating* is the predicted rating computed before an attack, *p_rating′* means the predicted rating computed after an attack [14].

$$APS = \frac{\sum_{i=1}^{M} | p\_rating_i - p\_rating_i' |}{M} \tag{7}$$

The evaluation value of *Prediction Shift* originally has two meanings, in other words, a positive value has different meaning from a negative value. Each value means that the attack has succeeded in making the target item more positively or negatively rated [14]. However, *APS* measures the absolute value just to evaluate the influence from injected profiles regardless of attack-classification in our experiments.

4.2 Preliminary Experiments

Overall Performance of Prediction Quality. Increasing the number of users used for similarity model building, we compared our system with one of the methods proposed in *PocketLens*. The experiment was carried out in order to indicate that the proposed method performs as good as an existing work. Prior to experiments, the vector of ratings $\vec{r}$ for each user was normalized as $\|r\| = 1$ except for experiments by using

an adjusted cosine-based similarity metric; otherwise, users who had rated a large number of items had more influence than users who had only rated a few items [9]. In the process of model building, mobile agents find neighbor peers based on the converted trust graph and compute similarity relations by using a cosine-based similarity metric and an adjusted cosine-based one (see Equation 1 and 4). On the other hand, in the benchmarked one, the information of the peers randomly accessed regardless of *web of trust* is used for model building by using the cosine-based one.

Table 2. Overall performance of prediction quality

Neighbor peer size	10	30	50	70	100
Random	1.2866	1.2863	1.2859	1.2859	1.2859
TCFMA + cosine	1.2113	1.2114	1.2100	1.2101	1.2101
TCFMA + adjusted	1.2384	1.2480	1.2412	1.2415	1.2402

As shown in Table 2, all three methods provided nearly the same values for MAE. When the *TCFMA + cosine-based* scheme was used, the results moved from 1.2113 to 1.2101, which shown better prediction quality than other two methods. It can be observed that the proposed methods provide more accurate predictions than random model building at all neighborhood size levels. For example, when the neighborhood size is 50, the *TCFMA + cosine-based* scheme and the *TCFMA + adjusted cosine-based* one obtain an *MAE* of 1.2100 and an *MAE* of 1.2412 whereas the random scheme obtains an *MAE* of 1.2859.

In addition, the results show that even a small number of users can build relatively better model with our proposed methods, whereas the random scheme needs more users' information to obtain a similarity model of stable.

Positive Effect of Trust for Prediction. When the dataset including the users who have many trust opinions is used for building a similarity model, the model includes a larger number of trustworthy users. In order to determine the sensitivity of trust opinion size on the quality of the prediction, we assumed that each user have trust users of only the number of x. In each step of evaluations, the trust opinion size was selectively varied for building a similarity model of each testing user. According to the value of x, the prediction quality in cases of the proposed methods, i.e. similarity model building based on the converted trust graph, was evaluated.

Table 3. Sensitivity of trust on MAE (neighbor peer size = 50)

Trust x	Trust 5	Trust 10	Trust 15	Trust 25	Trust 45
TCFMA + cosine	1.4131	1.3338	1.3313	1.2867	1.1611
TCFMA + adjusted	1.5688	1.3028	1.2952	1.2512	1.2238

The conclusion drawn from these results shown in Table 3 is that the more trust opinions are included in each user, the better prediction quality is obtained. This means that the *direct* trust opinions have positive influence on the prediction quality.

4.3 Performance Evaluations

Robustness against Profile Injection Attack. To evaluate the robustness against a malicious attack, a set of manipulated user profiles including arbitrary 50 ratings each was inserted into the original training dataset, while the number of these profiles was increased from 100 to 2000. Each user in the manipulated set was made to have the trust edges to all users in the set while some users in the original set were made to present trust to some of manipulated users. Through the prior experiment and previous research, we selected 50 as a neighborhood size and set it up in this experiment [1, 9]. In both cases of *TCFMA + cosine-based* and random model building, the average number of manipulated users accessed for building each user's model was measured and the distortion of prediction occurring due to these data was compared by using *Absolute Prediction Shift*.

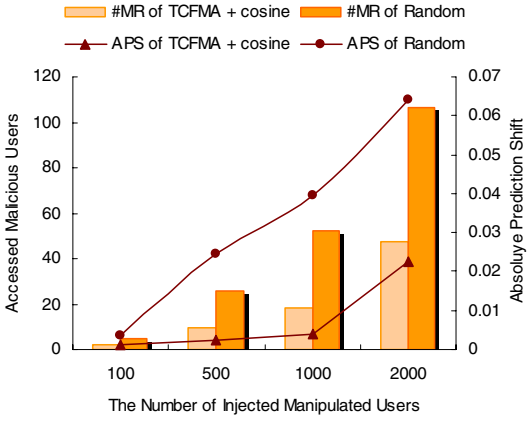

Fig. 4. Comparison of robustness on manipulated users

In accordance with increases in the number of manipulated users, we find that the proposed method showed better results than random model building in both measurements. Fig. 4 illustrates that the proposed method significantly outperforms others in resistance against the profile injection attack. For example, when the number of manipulated users was 2000, *TCFMA + cosine-based* scheme obtained a *#MR* of 47.71 and an *APS* of 0.0224, whereas the random scheme obtained a *#MR* of 106.56 and an *APS* of 0.0639.

Efficiency of similarity model building. Finally, we focused on the time required for model building. Forwarded along the path, a mobile agent only sends computed results to the personal agent of the model owner by the proposed methods whereas in the random scheme request/response messages have to be exchanged between the model owner and each random user for similarity model building. The experiment was conducted to evaluate the time and the number of accessed users that are required

Table 4. Comparison of required time and accessed users (neighbor user size = 50)

Model Owner		User 1	User 2	User 3	User 4	User 5	Average
TCFMA + cosine	Time(ms)	5786.81	11576.54	9776.97	12676.54	9425.59	9848.49
	# User	292.64	861.94	680.08	953.54	636.64	684.968
Random	Time(ms)	31590.24	30129.18	31966.27	23209.48	20977.24	27574.48
	# User	4379.48	4209.75	4505.89	3315.13	2962.29	3874.51

to build similarity models. For a <*item, rating*> pair of each 5 users in a testing set, the similarity model was built 30 times relatively to get the average performance.

The experimental results show that the average time of *TCFMA + cosine-based* model building is 9848.49 (msec.), which is reasonably shorter than 27574.48 (msec.) of the random one. In addition, the number of users who have to be accessed for similarity model building is also considerably smaller as shown in Table 4. These results demonstrate that the proposed method is far superior with respect to the effectiveness of similarity model building.

5 Conclusion and Future Work

In a peer to peer environment, a distributed recommender system is an ongoing area of diverse applications [1]. In the paper, we propose a novel *TCFMA* architecture to solve the problems that can occur in online collaborative filtering recommender systems related to an improper use of personal information and a profile injection attack. In order to obtain more trustworthy and accurate recommendations, we consider the trust relationships between users in *web of trust*. The *Advogato* trust metric is used as a trust propagation scheme required for overcoming sparseness of trust information. As noted in our experimental results, we obtain extraordinary robustness from malicious attacks without any degradation of prediction quality, compared to general peer-to-peer collaborative filtering recommender system. Moreover, we also achieve an efficiency of distributed computing for building item-item similarity model by employing trust-based collaborative filtering scheme which is added some useful functionality of mobile agents.

However, there still remains certain issue: trust decay. It means that the trust relationship becomes weaker as it forwards to its successors [2, 3]. In our system, although a mobile agent finds trust neighbors who are in the closest-distance from a model owner first, the neighbors, even on the different levels, are regarded as the same. It is essential to take this phenomenon into consideration for applying trust propagation algorithm to real-world application. Another interesting issue is about attack detection, which has been often shown in recent studies. Automated attack detection algorithms based on diverse types of attack model can lead more robust recommendation algorithms [13, 14, 17]. As the recommender systems have been more common in E-commerce application, these issues are becoming more interesting and important.

References

1. Miller, B., Konstan, J., Terveen, L., Riedl, J.: PocketLens: Towards a Personal Recommender System. In ACM Transactions on Information Systems 22 (2004) 437-476
2. Guha, R., Kumar, R., Raghavan, P., Tomkins, A.: Propagation of Trust and Distrust. In Proc. of the 13th Int. Conf. on World Wide Web (2004), ACM Press
3. Ziegler, C-N., Lausen, G.: Propagation Models for Trust and Distrust in Social Networks. In Information Systems Frontiers Vol. 7, Springer Netherlands (2005) 337 – 358
4. Levien, R., Aiken, A.: Attack Resistant, Scalable Name Service. Draft submission to the 4th Int. Conf. on Financial Cryptography (2000)
5. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based Collaborative Filtering Recommendation Algorithms. In Proc. of the 10th Int. Conf. on World Wide Web (2001)
6. Sinha, R., Swearingen, K.: Comparing Recommendations Made by Online Systems and Friends. In Proc. of the DELOS-NSF Workshop on Personalization and Recommender Systems in Digital Libraries (2001)
7. Ziegler, C-N., Lausen, G.: Analyzing Correlation between Trust and User Similarity in Online Communities. In Proc. of the 2nd Int. Conf. on Trust Management, LNCS, Vol. 2995 (2004) 251-265
8. Massa, P., Avesani, P.: Trust-aware Collaborative Filtering for Recommender Systems. In Proc. of Int. Conf. on Cooperative Information Systems (2004)
9. Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J.: An Algorithmic Framework for Performing Collaborative Filtering. In Proc. of the 22nd ACM SIGIR Conf. on Research and Development in Information Retrieval (1999)
10. Lange, D.B., Oshima, M.: Programming and Deploying Java Mobile Agents with Aglets. Addison-Wesley (1998)
11. Jung, J. J.: Visualizing recommendation flow on social networks. Journal of Universal Computer Science, Vol. 11, No. 11 (2005) 1780-1791
12. Kim, H.J., Jung, J.J., Jo, G.S.: Conceptual Framework for Recommendation System based on Distributed User Ratings. LNCS, Vol. 3032 (2003) 115-122
13. O'Mahony, M., Hurley, N., Kushmerick, N., Silvestre, G.: Collaborative Recommendation: A Robustness Analysis. ACM Transactions on Internet Technology, Vol. 4, No. 4 (2004) 344-377
14. Mobesher, B., Bruke, R., Bhaumik, R., Williams, C.: Towards Trustworthy Recommender Systems: An Analysis of Attack Models and Algorithm Robustness. to appear in ACM Transactions on Internet Technology, Vol. 7, No. 2 (2007)
15. Leivien, R.: Attack Resistant Trust Metrics. Ph.D thesis, UC Berkeley, Berkeley, CA, USA (2003)
16. Page, L., Brin, S., Motwani, R., Winograd, T.: The Pagerank Citation Ranking: Bringing Order to the Web. Technical Report, Stanford Digital Library Technologies Project (1998)
17. Twigg, A., Dimmock, N.: Attack-resistance of computational trust models. In Proc. of the 12th IEEE Int. Workshop on Enabling Technologies (2003) 275-280