Finding Reliable Recommendations for Trust Model

Weiwei Yuan, Donghai Guan, Sungyoung Lee, Youngkoo Lee*, and Andrey Gavrilov

Department of Computer Engineering, Kyung Hee University, Korea
{weiwei, donghai, sylee, avg}@oslab.khu.ac.kr, yklee@khu.ac.kr

Abstract. This paper presents a novel context-based approach to find reliable recommendations for trust model in ubiquitous environments. Context is used in our approach to analyze the user's activity, state and intention. Incremental learning based neural network is used to dispose the context in order to detect doubtful recommendations. This approach has distinct advantages when dealing with randomly given irresponsible recommendations, individual unfair recommendations as well as unfair recommendations flooding regardless of from recommenders who always give malicious recommendations or "inside job" (recommenders who acted honest previous suddenly give unfair recommendations), which is lack of consideration in the previous works. The incremental learning based neural network used in our approach also enables to filter out the unfair recommendations with limited information about the recommenders. Our simulation results show that our approach can effectively find reliable recommendations in different scenarios and a comparison is also given between previous works and our method.

1 Introduction

Computational models of trust have been proposed for ubiquitous environments because they are capable of deciding on the runtime whether to provide services to requesters who are either unfamiliar with service providers or do not have enough access rights to certain services. The basis for the trust model to make decision for unfamiliar service requesters are the recommendations given by recommenders who have past interaction history with the requesters. However, in the large-scale, open, dynamic and distributed ubiquitous environments, there may possibly exist numerous self-interested recommenders who give unfair recommendations to maximize their own gains (perhaps at the cost of others). Since recommendations given by recommenders are the key point for the trust model to make decision, finding ways to avoid or reduce the influence of unfair positive or negative recommendations from self-interested recommenders is a fundamental problem for trust model in ubiquitous environments. At the same time, because of the highly dynamic nature of ubiquitous environments, it is not always easy to get enough information about the recommenders. Hence the trust model is required to find the reliable recommendations with limited information about the recommenders.

* Corresponding author.

K. Aberer et al. (Eds.): WISE 2006, LNCS 4255, pp. 375 – 386, 2006.

The objective of this paper is to contribute to an approach which can find the reliable recommendations for the trust model in ubiquitous environments. This paper sets the stage by identifying a novel context-based approach using incremental learning algorithm. Context is used in our approach to analyze the user's activity, state and intention. The learning of context is incrementally increased by a Cascade-Correlation architecture neural network. By analyzing the context under which the recommendation was given, our method is able to filter out unfair recommendations in different scenarios. The advantages of our proposed approach are: (1) it can filter out randomly given irresponsible recommendations, individual unfair recommendations as well as unfair recommendations flooding no matter the recommendations are from recidivist (recommenders who always give malicious recommendations) or inside job (recommenders who acted honest suddenly give unfair recommendation on the benefit of themselves), (2) it can differentiate doubtful recommendations due to different reasons: the changing behaviors of service requesters in front of different recommenders, the incorrect observations by recommenders, as well as malicious intention of recommenders, (3) it is able to detect the malicious recommendations when limited information is available for the recommenders which is usually the case in a real scenario.

The rest of the paper is organized as follows. Section 2 gives the recommendation scenarios in ubiquitous environments. Section 3 gives our proposed approach in details. Section 4 gives the simulation results. Section 5 shows the comparison between our approach and previous works. The last section concludes the paper and points out the future work.

2 Recommendation Scenarios in Ubiquitous Environments

For the trust model in ubiquitous environments, the possible scenarios for the recommendations given by recommenders are as follows:

(1) Normal Recommendations.

a. Honest recommenders give accurate recommendations.

(2) Abnormal Recommendations.

b. Honest recommenders give inaccurate recommendations due to their incorrect observation.

c. Due to the changing behavior of service requester in front of different recommenders, honest recommenders give exceptional recommendations compared with recommendations given by other recommenders.

d. Recommenders give random recommendations at ease due to the lack of responsibility.

(3) Malicious Recommendations.

e. Recommenders who acted honest give unfair high or low recommendations individually. The past behaviors of these recommenders were always honest. However, they suddenly give unfair recommendations due to the relationship with the service requester or their own benefits. (Called Inside Job)

f. Recommenders who acted malicious give unfair high or low recommendations individually. Different from the recommenders in scenario e, these recommenders always give malicious recommendations in the past.

g. A number of recommenders who acted honest collude to give unfair recommendations (more than 50% of total recommendations), which causes the flooding of unfair recommendations. (Called Inside Job Flooding)

h. Unfair recommendations flooding similar as scenario g, but caused by recommenders whose past behaviors were always malicious.

A reliable trust model in ubiquitous environments should have the ability to filter out the recommendations in scenario b, d, e, f, g and scenario h, distinguish recommendations in scenario b and scenario d from recommendations in scenarios a, and differentiate scenario c apart from scenario b, d, e and scenario f.

3 The Proposed Approach

Trust is subjective since it is based on each user's own understanding of information. Hence it is relatively easy for the malicious recommender to pretend honest and for the honest recommender to be misunderstood as malicious, which makes it difficult to differentiate between the unfair and fair recommendations. Our key idea for the solution is that: different recommenders may give different recommendations due to their different understandings on the same information, however, from the view of psychology, one recommender will follow his own rule on recommendation giving, i.e., one recommender usually gives similar recommendations in similar context. In case one recommender gives exceptional recommendations compared with his previous ones in similar context, the reason lies in two aspects. One reason is that this exceptional recommendation is a malicious recommendation and should be filtered out. The other reason is that the recommender's rule on recommendation giving has changed, e.g. due the environments' effect on him, one recommender now only gives positive recommendation to the service requester whose past interaction with him are more than 80% successful in stead of 60%. In this case, our architecture will use incremental learning algorithm to catch up the recommender's new rule.

Table 1. Context used in our approach

ID	AB.	Context	Example
1	NA	User Name	Weiwei
2	RE	Relationship with other agents	Senior member
3	TD	Time/date of request	Weekday/daytime
4	CS	Current state	Busy working
5	PI	Past interaction history with service requester	57% successful
6	TL	Time of last communication	Within 3 days
7	CF	Confidence for the service requester in given time window	Number of total Communication is 27

To learn each recommender's rule on recommendation giving, we use incremental learning based neural network to learn the recommendations as well as the context under which the recommendations were given by the recommenders. The reason we use incremental learning is that the acquisition of a representative training data for the rule is time consuming and the rule is also possible to dynamically change from time to time. Consequently, it is not uncommon for such data to become available in small

batches over a period of time. In such setting, it is necessary to update an existing classifier in an incremental fashion to accommodate new data without compromising classification performance on old data [1]. The context which may relate to the learning of the rule is shown in Table1. Attributes 1, 3 and 4 of the context are specifically bounded to the recommender's activity or state, and the other attributes are supposed to hold regardless of the recommender's state since it depends on relationship of the external world where the recommender is currently situated.

We use the following architecture (shown in Fig.1.) to filter out the unfair recommendations.

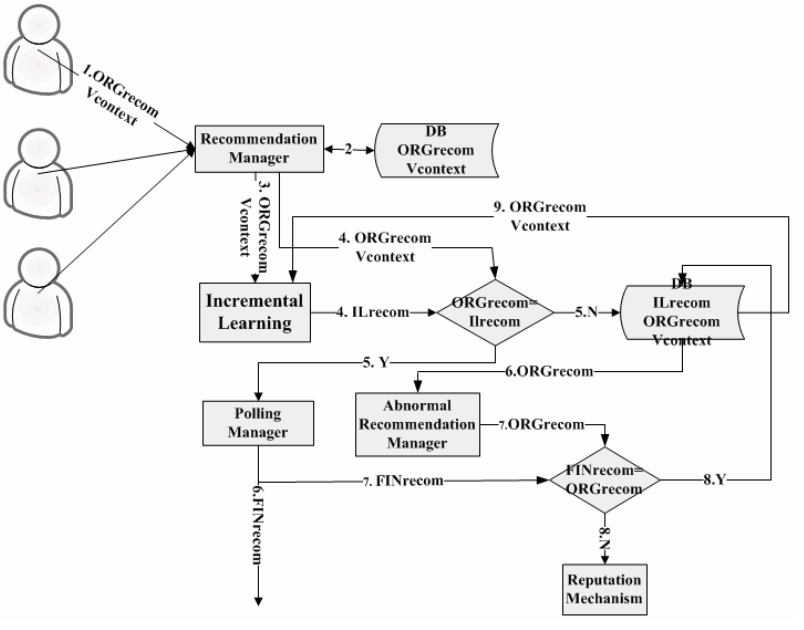


Fig. 1. Architecture Used for Filtering out Unfair Recommendations

Step 1: Recommendation Manager collects recommendations (REC_{org}) from all the recommenders, along with the context value $V_{context}[NA, RE, TD, CS, PI, TL, CF]$ under which recommendations were given by each recommender, where NA, RE, TD, CS, PI, TL, CF represent the context attributes mentioned in Table 1 from (1) to (7) respectively.

$$REC_{org} = \begin{cases} 1 & trusted \\ 0 & untrusted \end{cases}$$

Step 2: We use this step to find the doubtful recommendations from those gotten by step 1. To achieve this, we use incremental learning neural network, in particular, the Cascade-Correlation architecture.

Cascade-Correlation is useful for incremental learning, in which new information is added to an already-trained network. It is an architecture of neural network which begins with minimal network, then automatically trains and adds new hidden units one by one, creating a multi-layer structure. Once a new hidden unit has been added to the network, its input-side weights are frozen. This unit then becomes a permanent feature-detector in the network, available for producing outputs or for creating other, more complex feature detector [2]. Fig.2 gives the process of training Cascade-Correlation. In 1, we train the weights from input to output. In 2, we add a candidate unit and train its weights to maximize the correlation with the error. In 3, we retrain the output layer. We train the input weights for another hidden unit in 4. Output layer is retrained in 5, etc. The usage of Cascade-Correlation architecture has several advantages over others: it learns very quickly; the network determines its own size and topology; it retains the structures it has built even if the training set changes; and it requires no back-propagation of error signals through the connections of the network.

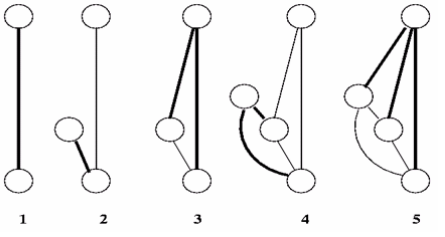


Fig. 2. Training Procedure of Cascade-Correlation Architecture

For each recommender who gives recommendation, the context $V_{context}[NA, RE, TD, CS, PI, TL, CF]$ under which he gave recommendation REC_{org} is used as the input of the Cascade-Correlation architecture. Using the Cascade-Correlation architecture trained by the previous context and corresponding recommendations, we will get the output REC_{IL}. REC_{IL} is the recommendation that the recommender will give due to his past behavior when given the input context $V_{context}[NA, RE, TD, CS, PI, TL, CF]$.

Compare REC_{org} and REC_{IL}:

$$REC_{com} = \begin{cases} REC_{IL} & REC_{IL}=REC_{org} \\ -1 & else \end{cases} \quad (1)$$

If $REC_{com} = REC_{IL}$, it means that the recommender gives the same recommendation as previous behavior in similar context, the possible scenarios are scenario a and scenario c. Otherwise if $REC_{com} = -1$, REc_{org} is regarded as a doubtful recommendation, which implies that the recommender's current behavior on

recommendation giving is different from previous. We will use following steps to judge whether this doubtful recommendation is malicious.

Step 3: We use this step to get the final recommendation. Since by using step 2 we have already found the doubtful recommendations, we use basic voting mechanism to calculate the final recommendation REC_{fin} . The voting is among the recommendations which are not doubtful ($REC_{com} = REC_{IL}$ in (1)).

$$REC_{fin} = \begin{cases} 1 & NUM\,[REC_{comi}=1 \,|\, REC_{comi} \neq -1] \geq \dfrac{NUM\,[REC_{comi} \neq -1]}{2} \\ 0 & else \end{cases}, \qquad (2)$$

where REC_{comi} is the REC_{com} of recommender i , $i \in N$. $NUM[REC_{comi} \neq -1]$ means the number of all the recommendations which are not doubtful. $NUM[REC_{comi} = 1 \,|\, REC_{comi} \neq -1]$ is the number of undoubtful recommendations which equal to 1.

Step 4: This step is used to find the malicious recommendations (scenario e, f, g and scenario h) and incorrect recommendations (scenario b and scenario d) from the doubtful recommendations gotten in step2.

In (1), if $REC_{com} = -1$, the possible situations are: A. the recommendation is malicious or incorrect, the possible scenarios are scenario b, d, e, f, g and scenario h, B. (1) the recommender's rule on recommendation giving has changed, i.e. the recommender gives different recommendation compared with previous one in similar context, however, this recommendation is also right. This kind of situation is reasonable since all the things in the world are always in movement, (2) the currently neural network is not enough to reflect the recommender's rule on recommendation giving since the Cascade-Correlation architecture begins with a minimal network and the knowledge on the recommender's rule is incrementally increased. We use the following method to differentiate between situation A and B.

$$result = \begin{cases} situationA & REC_{org} \neq REC_{fin} \,|\, REC_{com} = -1 \\ situationB & REC_{org} = REC_{fin} \,|\, REC_{com} = -1 \end{cases}, \qquad (3)$$

where $REC_{org} \neq REC_{fin} \,|\, REC_{com} = -1$ and $REC_{org} = REC_{fin} \,|\, REC_{com} = -1$ means $REC_{org} \neq REC_{fin}$ and $REC_{org} = REC_{fin}$ when given $REC_{com} = -1$ respectively.

If the result equals to situation B, $V_{context}[NA, RE, TD, CS, PI, TL, CF]$ as well as REC_{org} will be given back as retrain data to retrain the Cascade-Correlation architecture (as showed in Fig.1) to catch up the recommender's current rule on recommendation giving. Otherwise if the result is situation A, the record of this recommender will be given to a separated disposal unit and mark it as doubtful recommender. If one user appears several times as a doubtful recommender, it will be considered either as a malicious recommender or a recommender who is unable to

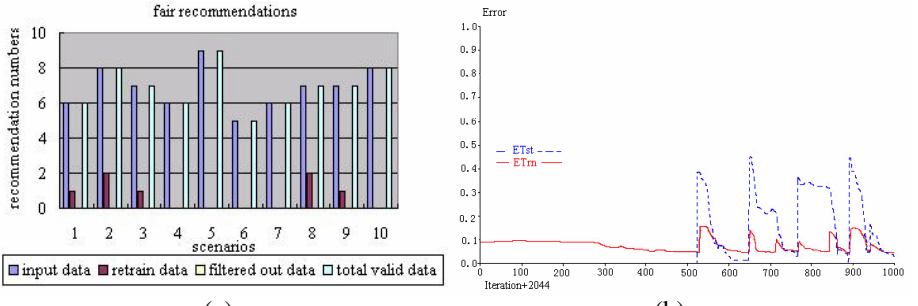
give correct recommendations. The recommendations given by this recommender later will be directly filtered out in step1.

4 Simulation Results

Our simulation is based on a ubiquitous computing supported smart office [3]. The possible recommenders in this smart office have different positions (in other words, they have different trust levels), such as professor, senior member. For the training of the original Cascade-Correlation architecture, we randomly choose 5 services requesters, the recommendations given by different recommenders as well as the context act as the training data. Each context is a vector including 7 attributes as shown in Table 1. The learning rate of the Cascade-Correlation neural network is set to 0.01 and the error tolerance is 0.05. The iterations for training the original neural network is 2252. As time goes by the Cascade-Correlation architecture will incrementally learn the new information (retrain data) which will make it closer to each recommender's current rule on recommendation giving. The following subsections give our simulation results in different scenarios mentioned in section 2.

4.1 Fair Recommendations

Fig.3 shows the simulation results when there is no unfair recommendation among all the recommendations (scenario a and scenario c). Since our approach compares the recommendations with the recommender's own previous behaviors, so there is no difference when dealing with scenario a and c in our architecture. The test data are the recommendations given by recommenders for different service requesters. The retrain data in Fig.3 (a) is the recommendations and the corresponding context which got $result = situationB$ in formula (3).



(a) (b)

Fig. 3. There is no unfair recommendation in total recommendations

As shown in Fig.3 (a), the total valid data is the same as input data, which means that when there is no unfair recommendation, even there are some recommendations have the result $REC_{com} = -1$ in formula (1), no data will be filtered out, in other

words, our approach does not have negative bias when the recommendations are all fair. Fig.3 (b) shows the simulation result of the Cascade-Correlation architecture using the data shown in Fig.3 (a). ETrn is the error of training data and ETst is the error of test data. When the error of training data is less than 0.05, the iteration is stopped. The sudden raise of ETrn is because of the adding of retrain data which were found as shown in Fig.3 (a). Due to the retrain of the Cascade-Correlation architecture, ETst also has the sudden raise along with the raise of ETrn. When ETrn is reduced to a reasonable level(less than 0.05 in our model), after a number of iterations, ETst is also reduced to a low level(less than 0.05), which means that the new neural network can fit the recommender's current rule.

4.2 Individual Unfair Recommendations

Using the input data as shown in Fig.3 (a), we randomly choose two recommenders and set their recommendations to be unfair. These data act as input data in this subsection. Since the recommenders are randomly chosen, they may have different trust levels. The possible scenarios are scenario b, e and scenario f.

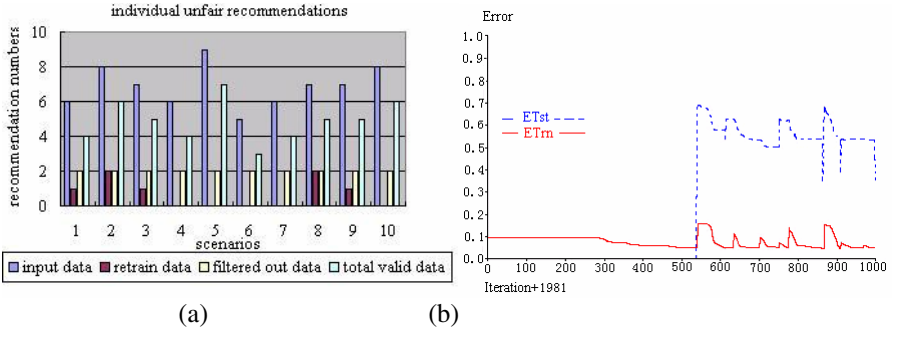


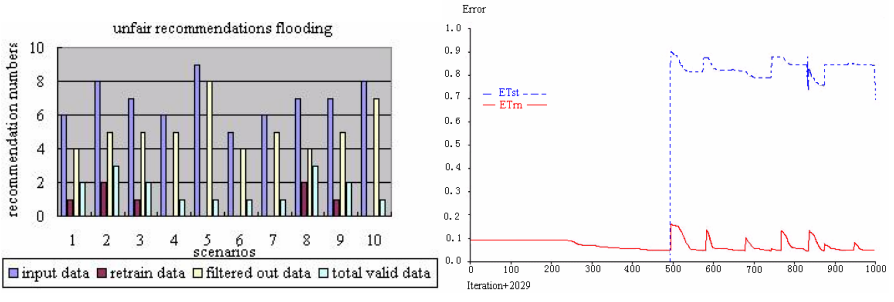
Fig. 4. There are individual unfair recommendations in total recommendations

The filtered out data as shown in Fig.4 (a) are those got *result* = *situationA* in formula (3), while if *result* = *situationB* in formula (3), these data will act as retrain data. Though $REC_{com} = -1$ in both case, only the data those act as retrain data can be valid data. Fig.4 (a) shows that our approach can accurately find the unfair recommendations (two for each input data as we set previous).

Using formula (2) and the input data showed in Fig.4 (a), we got the same final recommendation as section 4.1. Fig.4 (b) shows the simulation results for Cascade-Correlation architecture using the data in Fig.4 (a). The difference between Fig.4 (b) and Fig.3 (b) is that in Fig.4 (b), ETst is still big (more than 0.3) when ETrn is reduced to a reasonable level using the retrain data. The reason is that there are unfair recommendations among the test data (the input data in Fig.4 (a)). Since our approach can differentiate unfair recommendations from fair recommendations, ETst can not be reduced to a very low level along with ETrn.

4.3 Unfair Recommendations Flooding

Using the input data as shown in Fig.3 (a), we randomly choose the majority of recommenders and set their recommendations to be malicious. These data act as input data in this subsection. Since the recommenders are randomly chosen, they may have different trust levels. Therefore the possible scenarios are scenario g and scenario h. The result of the final recommendation using formula (2) in this case proves the correctness of our approach when dealing with unfair recommendation flooding since it is the same as the final decision in section 4.1 and 4.2.



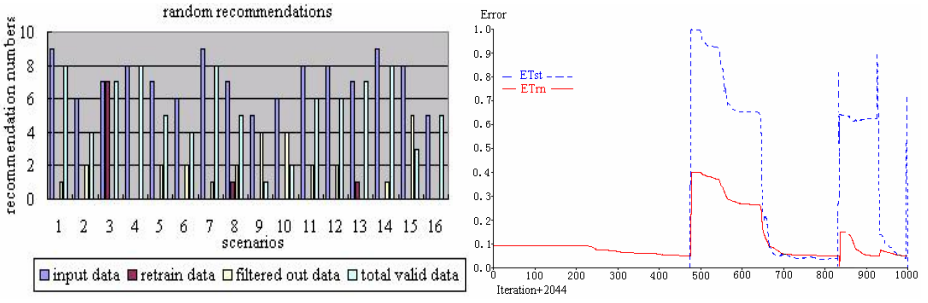
(a) (b)

Fig. 5. There are unfair recommendations flooding in total recommendations

If the unfair recommendations flooding can successfully influence the final recommendation, the fair recommendations will be regard as unfair. In this case since the unfair recommendations are majority, ETst can be reduced to a relatively low level (less than 0.5, the curve will be similar as Fig.4 (b)) along with ETrn. However, our simulation result shows that for ETst, when ETrn is reduce to a reasonable level, ETst in Fig.5 (b) is much higher than in Fig.4 (b) (above 0.7), which means that our approach can defend the unfair recommendations flooding.

4.4 Randomly Given Recommendations

The possible scenario in this case is scenario d. Since the recommendations are randomly given, there will be random number of filtered out data (as shown in Fig.6 (a)) as well as retrain data. Fig.6 (b) gives the error of Cascade-Correlation architecture given the data in Fig.6 (a). The curve of ETst does not have regularity. It can be up to a very high level (near 1.0) since most of recommendations for certain service requester are unfair and it can also be a very level (lower than 0.05) since the recommendations for certain service requester are fair. The reason our approach can filter out the randomly given recommendations is that if the recommendations are given randomly, it must be different with its own past behavior in similar context. Therefore, we will get $result = situationA$ in formula (3).



(a) (b)

Fig. 6. The recommendations are randomly given

5 Comparison with Related Works

There are some researches that gave helpful attempts on how to get reliable recommendations, especially for scenario f and scenario h. One method is to use polling method, e.g. in [4], the authors used basic polling as well as enhanced polling. The enhanced polling differs from the basic polling by requesting voters to provide their servent_id to prevent a single, possible malicious user to create multiple recommendations at a time. Another very popular method is to give weighted value to each recommender (also called a reputation-based method) to choose reliable recommendations. The reputation-based method had been used in a number of works, e.g. weighted majority algorithm is used in [5], and a Rating Reputation Feedback mechanism is used in [6] to train the weighted values. In [7] [8] [9] [10], the authors measure the reputation for each recommender and filter out unfair recommendations based on the usage of the reputation. Using the combination of different filters is also a reasonable method to filter out the unfair recommendations, as mentioned in [11] [12] [13] [14]. Their simulation results suggest that cluster filtering is suitable to reduce the effect of unfairly high recommendations and positive discriminations and frequency filtering can guarantee the calculation of trust not be influenced by the unfair raters flooding.

Table.2 gives the comparison between different methods when dealing with different scenarios as mentioned in section 2. The reason previous methods can not deal with some of the scenarios is that they took at least one of the following assumptions: (1) most recommendations are close in the range of the real quality of the product, (2) recommendations provided by different recommenders on a service requester will follow more or less the same probability distribution, (3) top ranked recommenders are the expert recommenders in the trust category, i.e., the higher rank recommender has, the more authority his recommendation will have. For example, as the most popular method, reputation-based method takes assumption (3), hence it is impossible for this method to filter out any of the inside job (scenario e and scenario g). What's more, if scenario e and scenario g happens, the higher the recommender's rank is, the more serious aftereffect there will be. Our approach is effective in different scenarios because the comparison used to filter out the unfair recommendations is not between different recommenders but between each recommender's own current behavior and previous behavior. Hence we do not take any of these assumptions.

Table 2. Comparison between different methods

Scenario	Polling	Weight-based (Reputation-based)	Combination of Filters	Our Approach
a	Y	Y	Negative Reputation Bias	Y
b	Y	High Rank User N Low Rank User Y	Y	Y
c	N	High Rank User Y Low Rank User N	N	Y
d	N	High Rank User N Low Rank User Y	Effective when variance is large	Y
e	Y	N	Y	Y
f	Y	Y	Y	Y
g	N	N	Y	Y
h	N	Y	Y	Y

However, the cost of our method is that it takes longer time for our approach to find the reliable recommendations. The reason is that to judge the validity of recommendations, each recommender's current behavior should be compared with his past behaviors. However, since these calculations take place in the service agent which has enough computing ability, we believe that it does not distinctly affect the efficiency of the whole trust model.

6 Conclusions

In this paper we propose a robust trust model for ubiquitous environments, in which a context-based approach is used to filter out the unfair recommendations. The learning of the context is based on incremental learning neural network. The filtered out recommendations may be the intended unfair recommendations as well as the mis-observation by the recommenders. We also focus on the flooding of unfair recommendations in this paper. Since our approach concentrates on the doubtful behavior of each entity, it has special advantages when dealing with inside job, which is lack of considerations in the current trust models.

In the future work, we plan to focus on the scalability of trust model since our trust model is merely used in a smart office now. And we also want to find the relationship between different attributes in the context shown in Table.1, and try to find the most important ones. We also plan to add risk analysis in our context-based trust model. Based on our comparison between our context-based approach and other methods, we believe that the usage of context-based trust model and incremental learning neural network within ubiquitous environments application presents a promising path for the future research.

Acknowledgment. This research was supported by the MIC (Ministry of Information and Communications), Korea under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment) in collaboration with SunMoon University.

References

1. Robi Polikar, Lalita Udpa, Satish S. Udpa and Vasant Honavar, "learn++: An Incremental Learning Algorithm for Supervised Neural Networks", IEEE transactions on systems, man, and cybernetics-Part C: Applications and Reviews, Vol. 31, NO.4, Nov.2001
2. Scott E. Fahlman, Christian Lebiere, "The Cascade-Correlation Learning Architecture". Technical Report CMU-CS-90-100, School of Computer Science, Carnegie Mellon University
3. Hung Q. Ngo, Anjum Shehzad, Saad Liaquat Kiani, Maria Riaz, Kim Anh Ngoc, Sungyong Lee.: Developing Context-aware Ubiquitous Computing Systems with a Unified Middleware Frame Work. The 2004 International Conference on Embedded & Ubiquitous Computing (EUC2004)
4. Damiani, Vimercati, Paraboschi, Samarati, and Violante, "A reputation-based approach for choosing reliable resources in peer-to-peer networks", 9th ACM CCS 2002
5. Bin Yu, Munindar P. Singh, and Katia Sycara, "Developing trust large-scale peer-to-peer systems", First IEEE Symposium on Multiagent Security and Survivability, 2004
6. Ping Xu, Ji Gao, Hang Guo, "Rating Reputation: a necessary consideration in reputation mechanism", Proceedings of 2005 International Conference on Machine Learning and Cybernetics
7. Whitby, A., Josang, A. and Indulska, J. "Filtering out unfair ratings in Bayesian reputation systems", AAMAS 2004, New York, USA
8. Weihua Song, Vir V. P hoha, and Xin Xu, "An adaptive recommendation trust model in multiagent system", IEEE/WIC/ACM IAT'04
9. Weihua Song, Vir V. Phoha, "Neural network-based reputation model in a distributed system", pp. 321-324, 2004 IEEE International Conference on E-Commerce Technology (CEC'04), 2004
10. Huang Baohua; Hu Heping; Lu Zhengding, "Identifying local trust value with neural network in p2p environment", The First IEEE and IFIP International Conference in Central Asia on Internet, 2005
11. C. Dellarocas, "The design of reliable trust management systems for electronic trading communities", MIT Working Paper
12. C. Dellarocas , "Building trust online: the design of robust reputation reporting mechanisms for online trading communities" A combined perspective on the digital era, Doukidis, G., Mylonopoulos, N. and Pouloudi, N. (Eds.), Idea Book Publishing (2004)
13. C. Dellarocas. "Immunizing online Reputation Reporting systems against unfair ratings and discriminatory behavior", In Proceedings of the ACM Conference on Electronic Commerce, pages 150--157, Minneapolis, Minnesota, USA, 2000
14. Chrysanthos Dellarocas , "Mechanisms for coping with unfair ratings and discriminatory behavior in online reputation reporting systems", In ICIS, pages 520--525, 2000