Trust and Reputation Framework for Social Web Platforms: Research Agenda and Preliminary Model

Thao Nguyen[1,2]*, Luigi Liquori[1], Bruno Martin[2], and Karl Hanks

[1] Institut National de Recherche en Informatique et Automatique, France
[2] Université de Nice Sophia Antipolis, France
{Thao.Nguyen,Luigi.Liquori}@inria.fr
Bruno.Martin@unice.fr
karl_hhanks@yahoo.co.uk

Abstract. Trust and reputation systems (TRSs) have recently seen as a vital asset for the safety of online interaction environment. They present in many practical applications, e.g., e-commerce and social web. A lot of more complicated systems in numerous disciplines also have been studied and proposed in academia. They work as a decision support tool for participants in the system, helping them decide whom to trust and how trustworthy the person is in fulfilling a transaction. They are also an effective mechanism to encourage honesty and cooperation among the users resulting in healthy online markets or communities. The basic idea is to let parties rate each other so that new public knowledge can be created from personal experiences. The greatest challenge in designing a TRS is making it robust against malicious attacks. In this paper, we provide readers an overview on the reasearch topic of TRSs, propose a consistent research agenda in studying and designing a robust TRS, and present an implemented reputation computing engine alongside simulation results, which is our preliminary work to acquire the target of a trust and reputation framework for social web applications.

1 Introduction

Since ancient times, reputation spread out via word-of-mouth networks has been used as an enabler of several economic and social activities. Nowadays, with the development of technology, in particular the Internet, reputation information can be spread out more easily and faster than ever. Trust and reputation schemes have attained attention of many information and computer scientists since the early 2000s. They have a wide range of application and are domainspecific. Multiple application areas of trust and reputation schemes can be named as social web platforms, e-commerce, peer-to-peer networks, sensor networks, ad-hoc network routing, etc. Among them, we are most interested in the field of social web platforms. We easily notice that trust and reputation has presented in many

* Corresponding author.

online systems, such as online auction and shopping websites eBay [1], where people buy and sell a broad variety of goods and services, and Amazon [2] who is a worldwide famous online retailer. Online services with trust and reputation systems (TRSs) could provide certain safety to their users compared to those without TRSs. A good TRS can also create an incentive for good behavior, threat for tricky ones, and mitigation of malicious actions to the main system. As the analysis of [9], markets with the support of reputation systems will be healthiest with a variety of prices and quality of service. TRSs are very important to an online community network with respect to safety for participants, robustness of the network against malicious behavior and fostering a healthy market.

From the functional view, a TRS can be split into three components [8]. The first function is to gather all feedbacks on a participants past behavior in transactions that were involved. This component includes storing feedbacks by users after each transaction they take part in. The second function is to compute reputation scores for participants through a reputation computing engine based on the gathered information. The third function is to process the reputations scores, having appropriate reward and punishment policies. A TRS can be centralized or distributed. In centralized TRSs, there is a central authority responsible for collecting ratings and computing reputation scores for users. Most of alive TRSs on the Internet now are centralized, for examples the feedback system on eBay [1] and customer reviews on Amazon [2]. Meanwhile, a distributed TRS has no central authority like that. Each user has to collect ratings and compute reputation scores for other users himself. Almost all proposed TRSs in the literature are distributed [**?**].

Some main unwanted behavior of users that TRSs should take into account are: *free riding* (where people are usually not willing to give feedback if they are not given an incentive to do so [9]), *untruthfully rating* (users give incorrect feedback because of either objective conditions or their intent to serve their malicious purposes), *colluding* (a group of users coordinate their behavior to inflate each other's reputation scores or bad-mouth other competitors. The colluding motives are only clear in a specific application), *whitewashing* (a user creates a new identity in the system to replace his old one when the reputation of the old one has gone bad), *milking reputation* (at first a participant behaves correctly to get high reputation and then turns bad to make profit from its high reputation score). The milking reputation behavior is more harmful to social network services and e-commerce than to the others and needs being detected as soon as possible.

2 Trust and Reputation System Design Agenda

Building a TRS is not a one-task job. Therefore, we find it necessary to have a concrete agenda to guide researchers through the process of studying and building a TRS which is robust to attacks. Our proposed agenda is depicted in Fig. 1. First, we characterize the system that we want to integrate a TRS into. Second, based on the characteristics of the application, we find suitable working mech-

anisms and processes for each component of the TRS. Corresponding to each component, there are key questions that need to be answered, like "What kinds of information need collecting and how?", "How to compute reputation scores using the collected information?", and "How to represent and process the reputation score to lead users to a correct decision?". According to [5], a reputation computing engine should meet these criteria: accuracy for long-term performance (distinguishing a newcomer with unknown quality with a low-quality participant who has stayed in the system for a long time), weighting towards recent behavior, smoothness (adding any single rating should not change the score significantly), and robustness against attacks. Third, we study the tentative design obtained after the second step in the presence of selfish behaviors. During the third step, we can repeatedly come back to step 2 whenever appropriate until the system reaches a desired or acceptable performance. The fourth step will refine the TRS and make it more robust against malicious attacks. Some of general selfish behaviors and malicious attacks are already mentioned in Sect. 1. If any modification is made, we need to come back to step 2 and check all the conditions in steps 2 and 3 before accepting the modification (Fig. 1).

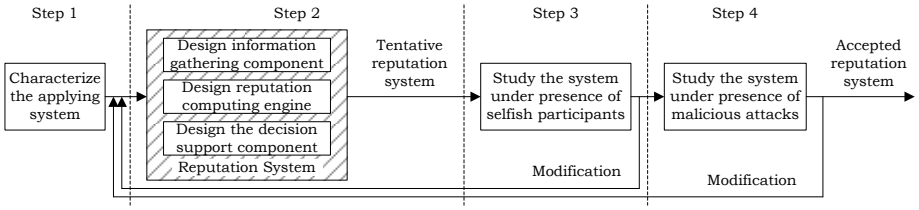

Fig. 1. Process of designing a robust reputation system

Most of the challenges for a TRS are induced by selfish and malicious behaviors. The problems arising from malicious behaviors are usually more complicated and difficult to cope with than those by normal selfish users. Therefore, the logic of our methodology is that we first design a reputation system that works for a community of all obedient members, and then the difficult level of designing and the sophistication of the system are increased to cope with selfish members and finally with malicious ones.

There is no single TRS which is suitable to all contexts and applications. In different applications, there are different kinds of available information and activities, hence different ways of computing reputation scores and different types of attacks. Accordingly, designing details of a TRS must be put into specific context of the application. However, in such circumstances, the methodology of our agenda is still useful and applicable anyway.

2.1 Related Work

In the literature, we have found a number of proposed agenda and guides in studying and building a TRS. But still, no one is as generic and comprehensive as ours. The proposed agenda is a complementary view contributed to the existing ones. Useful principles stated in [5] are finding and identifying new information elements substituting for traditional cues of trust and reputation in the physical world, and taking advantages of information technologies to collect vast amounts of necessary data. They fit into the task of designing the information gathering component in Step 2 of our agenda. In [4], the authors focus on robust reputation systems. Their research agenda is specially concerned about methods for evaluating the robustness. They propose three possible approaches including evaluating by implementing the system in reality, theoretical test by third parties so that the evaluation is more credible, and defining a comprehensive set of robustness evaluation methods and criteria. For social web applications, the work in [3] is an effective guide. It is applicable directly to step 2 of our agenda (Fig. 1) which are designing the components of gathering information and supporting user's decision.

3 Preliminary Trust and Reputation Framework

Among three components of a TRS, the information gathering one is the most dependent on the application system. The decision support component comes afterwards and then follows the reputation computing engine. Accordingly, the next step in our research will be building a robust reputation computing engine as more general as possible so that the engine is applicable to a variety of applications. We see that there is a large group of systems where a transaction is bilateral. In these systems, for each transaction there are two parties that we call consumer and provider. Consumer is the one who requests for a service, provider is the one who is capable of providing the service. When we add TRS to this kind of system, a user can have an additional role as a rater who has interacted with the provider before and now has an opinion about the provider's service. The following example scenario will give readers a clear view on how the system will work in the presence of a TRS.

In Fig. 2, the consumer $C1$ is looking for Service 1. The list of providers for the service is $\{P1, P2, P3, P4, P5\}$. Corresponding to each provider, there is a rater set like the one for provider $P1$ is $\{R1, R2, R3\}$. The rater set for $P4$ is empty because no one has yet been using the service provided by him. A rater can have ratings for several providers, like the case of $R3$ with providers $P1$, $P2$ and $P3$. In this example scenario, the consumer $C1$ already has personal experience of Service 1 provided by $P2$ and $P5$. To calculate the reputation score for $P1$, consumer $C1$ collects ratings by $R1$, $R2$ and $R3$. If the consumer has previous experience with the provider, like the case of provider $P2$, then $C1$'s personal experience is taken into account in computing reputation. After calculating reputation scores for all candidate providers, the consumer can rank

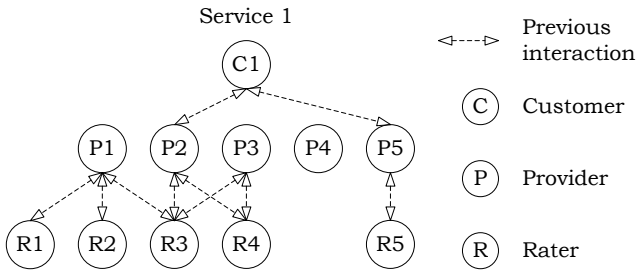

Fig. 2. Relationships between consumer, providers and raters.

them according to reputation scores and choose one of the top ranked providers to interact with.

4 Reputation Computing Engine

In this section we will elaborate on our preliminary implemented reputation computing engine. It works in a distributed manner because each user in the system collects ratings and computes reputation scores by himself when he has a request. The reputation scores of a provider vary correspondingly to each different consumer. Hence, they are local, but not global values, and recalculated every time the consumer has a query for the service.

4.1 Assumptions and Notations

Without losing generality, we consider the engine within the context of a service having one criterion to be judged. An example of service with multiple criteria is the one provided by an eBay user [1]. Upto May 2012, the service is judged by four criteria including "Item as described", "Communication", "Shipping time", and "Shipping and handling charges". The only one criterion in our context is called quality of service (QoS) whose values are normalized to the interval $[0, 1]$ and 1 is the optimal value. Correspondingly, the rating value is in $[0, 1]$ and the optimal value is 1 too. The following are main variables that a consumer will use for his computation. They are private information and accessible only to the owner consumer.

Rater Credibility (C_r): can be seen as reputation of a user in giving accurate feedback. It reflects how much the consumer should trust the rater r's ratings. C_r has a value in the range of $[0, 1]$, and is a private information kept by the consumer.

Usefulness factor (U_r): is the ratio of the number of times that the rater r submits useful ratings N_{useful} over the total number of submissions S. $U_r = N_{useful}/S$. After a transaction, if the difference between a rating and the outcome observed by the consumer is under a predefined threshold then the rating is useful. In our model, this threshold is set to 0.2.

Personal evaluation ($PerEval$): is the consumer's first-hand experience with the provider and has a value within $[0, 1]$. We set $PerEval$ as the experience of the last transaction with the provider. It might be not available if the consumer has never done any transaction with this provider.

Previously assessed reputation score (A): is the computed reputation score of a provider in the last time that the consumer interacted with him. If the consumer has never interacted with this provider, then A will be initiated as 0.5.

4.2 Adjusting Raters' Credibility

After collecting ratings on a provider's service, the consumer will adjust raters' credibility which will be used to calculate weights for the ratings. Two main criteria to adjust a rater's credibility are the consistency of his rating to other raters' and to the previous reputation score A of the provider. We use a modification of the k-mean clustering algorithm in [6] to find out the majority rating value among the raters. The main idea of this algorithm is to divide the rating set into clusters such that similar ratings are grouped into the same cluster, while different ones are separated into different clusters. The most crowded cluster is then labeled as the majority and its mean is the majority rating M. How close and how far of ratings to decide if they are in the same or different clusters are affected by distances C and R: $0 \leq C \leq R \leq 1$. The three parameters of the algorithm are coarsening and refinement distances (C and R), and the initial number of clusters (k). After having M, the consumer computes factor $M_r^f \in [0, 1]$ of rater r, which has an effect on the change of the rater's credibility due to its agreement/disagreement with the majority rating.

$$M_r^f = \begin{cases} 1 - |R_r - M|/\sigma_M \text{ if } |R_r - M| < \sigma_M \\ 1 - \sigma_M/|R_r - M| \text{ otherwise} \end{cases}$$

Where σ_M is the standard deviation of the received rating set.

$$\sigma_M = \sqrt{\sum_{r=1}^{N_R} R_r^2/N_R - (\sum_{r=1}^{N_R} R_r/N_R)^2}$$

with N_R is the total number of collected ratings. Factor $A^f = 1$ has an effect on the change of a rater's credibility due to its agreement/disagreement with the previous assessed reputation A. Thereafter, the credibility C_r of rater r is adjusted as the following:

1. If rating R is similar to both the *majority rating* M and the previous reputation A, i.e., ($|R - M| < 0.1$, $|R - A| < 0.1$), then the credibility of the rater is increased as: $C_r = Min(1, C_r + \aleph * (M_r^f + A^f)/\rho)$.
2. If ($|R-M| < 0.1$) and ($|R-A| \geq 0.1$), the rater's credibility is still increased, but less than the first case: $C_r = Min(1, C_r + \aleph * M_R^f/\rho)$.

3. If $(|R - M| \geq 0.1)$ and $(|R - A| < 0.1)$, the rater's credibility is decreased a little bit: $C_r = Max(0, C_r - \aleph * A^f/\rho)$.
4. If $(|R - M| \geq 0.1)$ and $(|R - A| \geq 0.1)$, the rater's credibility is decreased the most: $C_r = Max(0, C_r - \aleph * (M_R^f + A^f)/\rho)$.

Where $\aleph = C_r * (1 - |R - M|)$, and ρ is the consumer's pessimism factor which has a suggested minimum value of 2 ($\rho \geq 2$). The higher ρ is, the more difficultly the consumer trusts raters. A pessimistic consumer with high ρ will increase raters' credibility slowly for a rating consistent with M and A. Finally, the rater's credibility is adjusted by its *usefulness factor*: $C_r = C_r * u_r$.

4.3 Computing Assessed Reputation Score for Providers

To prepare weights for collected ratings, besides credibility of raters, we need to calculate *temporal factors* (f_r^t) for ratings and personal evaluation as well. The reason for using temporal factors is to stress more weight to the more recent information. Depending on the characteristics of the service and the discretion of designers, temporal factors can be calculated in several different ways. Our proposal here is one example. Ratings are arranged into chronological order, and then the temporal factor corresponding to each rating is the inverse of the number of ratings counted from it to the latest ratings (Fig. 3).

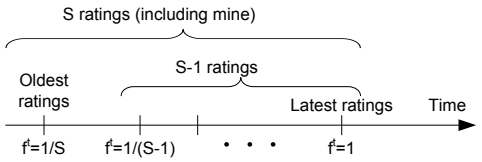

Fig. 3. A method to calculate temporal factors for ratings

Then the assessed reputation score of provider p is computed as in the following formula:

$$Rep_p = (\sum_{r=1}^{N_R}(R_r * C_r * f_r^t) + PerEval * f^t)/(\sum_{r=1}^{N_R} C_r + 1)$$

Where R_r is the rating by rater r, C_r is its credibility, f_r^t is the temporal factor put on the rating, $PerEval$ is the consumer's first-hand experience (if available) with the provider, f^t is the temporal factor for the consumer's $PerEval$, and N_R is the total number of collected ratings.

5 Decision Support

In many works [**?**], after the computation, the consumer opts to interact with the provider having the highest reputation score. This strategy is obvious and simple.

However, in our opinion, the chosen provider is not necessarily the one having the highest reputation score for two reasons. First, there might be errors or tolerance in collecting data or computation, leading to the fact that the provider having the highest reputation score is not really the best provider. Second, human decision-making is complex and based on not only reputation and personal experience but also many other elements like his aversion, bias, antecedents, mood, etc, which are impossible to model all in TRSs. Therefore, in reality, a user might choose the provider not having the highest reputation score. Being aware of that fact, we suggest TRSs in social web applications should stay at the level of supporting users, not make decisions for them.

For the purpose of examining a TRS, we model users' decision-making in selection strategies. Actually, these strategies can be part of a TRS when participants of the system are agents other than human such as computers, routers, devices, sensors, or software. Besides the traditional strategy of choosing the most reputable provider, we implement a strategy as following. The consumer ranks candidate providers based on their reputation scores, cuts off the ones having scores 0.5 lower than the score of the first ranked provider to get a short list of size N_{Size}, uses a Gaussian distribution having standard deviation $\sigma = \sqrt{N_{Size}}$ and mean $a = 0$ to calculate Gaussian random values for providers in the short list. Accordingly, Gaussian value of a provider depends on the size of the short list and its rank in the list, but not its absolute reputation score. Finally, the consumer opts a provider randomly with a probability proportional to its Gausian value.

6 Simulation results

We have implemented the above described reputation computing engine and tested it under multiple behaviors of users. The following is the details of our simulation.

6.1 Simulation Settings

We set up a population of N_u users providing the same service. A round of simulation includes N_t transactions. In each transaction, a random consumer is assigned to request the service. Other users then will be candidate providers for this request.

When a user plays the role of consumer, his behavior is modeled in $raterType$ attribute. In the simulator, we implement four types of raters including HONEST, DISHONEST and COLLUSIVE types. HONEST raters share their personal experience honestly, i.e. $R = PerEval$. DISHONEST raters provide ratings of 0.5 different from their true estimation, i.e. $R = PerEval \pm 0.5$. COLLUSIVE raters give highest ratings ($R = 1$) to users in their collusion and lowest ratings ($R = 0$) to the rest. Similarly, when a user acts as a provider, he can be one of the following types of providers: GOOD, NORMAL, BAD, or GOODTURNBAD. This type is denoted in $providerType$ attribute. The QoS of the service provided

by a GOOD, BAD or NORMAL provider has a value in the interval $(0, 0.4]$, $(0.4, 0.7]$, or $(0.7, 1]$ respectively. A GOODTURNBAD provider will change the QoS of his service in the meantime of a round of simulation. What a consumer will do to get a transaction done are getting a list of providers, computing reputation scores for them, choosing a provider to perform the transaction, updating his private information and publishing his rating for the provider. The quality of service that the consumer will experience depends on the $providerType$ of the chosen provider. The difference between the consumer's observation and his rating for the provider depends on the consumer's $raterType$.

For each user, we have the following measures. p_p: Percentage of transactions in which the user has performed as the provider over the total number of transactions of the simulation round. Apparently, p_p should be proportional to the user's QoS. a_d: Average absolute difference between the estimated reputation score Rep_p before the transaction and the personal evaluation $PerEval$ after the transaction. It reflects the correctness of the system in assessing a provider's QoS. a_d is captured when a user plays the role of the consumer.

We do at least five rounds of simulation and then take the average values to analyse. A round of simulation is denoted by a set of parameters as the following: ***Simulation(***N_u***,*** N_t***,*** $\%G$***,*** $\%N$***,*** $\%B$***,*** $\%GTB$***,*** $\%H$***,*** $\%D$***,*** $\%C$***,*** $\%dataLost$***)***. Where N_u is the number of users, N_t is the number of transactions. $\%G$, $\%N$, $\%B$, $\%GTB$ are percentage of GOOD, NORMAL, BAD, and GOODTURNBAD providers in the porpulation respectively, so that $\%G + \%N + \%B + \%GTB = 100\%$. $\%H$, $\%D$, $\%C$ are percentage of HONEST, DISHONEST, and COLLUSIVE raters respectively, so that $\%H + \%D + \%C = 100\%$. $\%dataLost$ is the percentage of ratings on a specific provider which are not available for the consumer at the moment he computes the reputation score for the provider.

6.2 Simulation Scenarios and Analysis

In this section, we are going to apply the methodology mentioned in Sect. 2 to study the implemented reputation computing engine. We examine the engine to see firstly if it works correctly when all users are obedient, secondly if it is robust against selfish users, thirdly if it is robust against malicious users.

Obedient Users. We consider obedient users the ones who provide a service truly as their capability and as stated, and share honestly their personal experience with the community. Injecting the thought into our simulation model, they are $GOOD$, $NORMAL$, or BAD providers and $HONEST$ raters. Therefore, we run a simulation of the following set of values: ***Simulation(200, 10000, 10, 20, 70, 0, 100, 0, 0, 0)***. The result shows that the engine works accurately in this scenario. The a_d of $GOOD$ and $NORMAL$ users are almost 0. BAD users represent for 70% of the population, but they occupy only 0.5% of total number of transactions. That means most of the times consumers make right decisions to avoid BAD providers.

Selfish Users. A selfish behavior in generic TRSs can be named as *free riding*. The consumer does not give feedback to the system after a transaction.

We roughly simulate that behavior via parameter $\%dataLost$, and run the following simulation: ***Simulation(200, 10000, 10, 20, 70, 0, 100, 0, 0, 60)***. Compared to the result when all users are obedient, the obtained one this time is almost the same. It proves that the engine still functions even when 60% of supposed-to-be-available ratings are not acquired.

Malicious Raters. As raters, malicious users can be categorized into DISHONEST and COLLUSIVE ones. DISHONEST raters act individually, while COLLUSIVE ones act in groups. It is difficult to identify COLLUSIVE users especially when they form into a large group. ***Simulation(200, 10000, 10, 20, 70, 0, 30, 70, 0, 0)*** 4 shows the result when 70% of users share dishonestly their personal experience. The error in computing reputation scores for BAD providers is 0.39 in average. And this error gives $70\% \times N_u$ BAD providers a chance to acquire $13\% \times N_t$ transactions. Similarly, the results for ***Simulation(200, 10000, 10, 20, 70, 0, 40, 0, 60, 0)*** where 60% of users collude into a group against the rest is in Fig. 5. From these results, we conclude that the engine is robust against dishonest behavior of upto 70% and colluding behavior of upto 60% of the population.

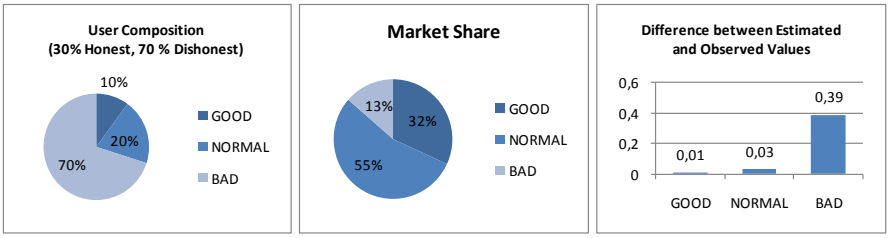

Fig. 4. Simulation results for scenario with DISHONEST raters

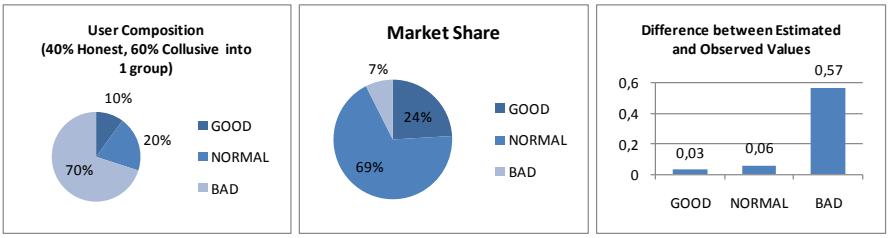

Fig. 5. Simulation results for scenario with COLLUSIVE raters forming a group

Malicious Providers. A malicious behavior of provider is *milking reputation* which can be modeled by GOODTURNBAD users: ***Simulation(200, 10000, 10, 10, 70, 10, 100, 0, 0, 0)***. We observe that when the users change

QoS of their service from high to low, they still get high reputation scores and high chance of being selected many times afterwards. Even the consumer who has experienced the bad service by a GOODTURNBAD provider, can still choose the provider again because the personal experience is not put enough weight while many other raters give high ratings for him. We conclude that the current engine is not responsive to dynamic behavior of users.

7 Related Work

After a thorough survey of current research, we find most interested in the model proposed in [7]. The service oriented environment analyzed in the paper fits into the bilateral transaction environment we are aiming at. The provided experimental results are appealing. However, the disadvantage of the model in [7] is that it is complicated to implement with unclear complexity of algorithms used in the model. Scalability of the system is not justified. Furthermore, a number of undecided algorithms which are left to the discretion of readers are still unspecified in conducted experiments. While we think that the variance of these algorithms can lead to subtle differences in experimental results. Such algorithms include the one for updating personal evaluation, and the one for recording and aggregating the provider's assessed reputations at previous time instances. Some other unclear points presenting in the paper are the thresholds to estimate if a rating is useful, to decide if a rating is similar to the majority rating and previously assessed reputations. These thresholds all affect the precision of assessed reputations. For the experiments with pessimism factor, the authors state in general that with high pessimism factor (pessimistic consumer), the assessed reputations are closer to the original performance and differences are within an acceptable threshold of 0.2, but not saying exactly what the value of the pessimism factor has been used.

In our work, we aim at a simple, yet robust, scalable and light framework so that it is applicable even in the environment of limited computing capacity devices like smartphones. We adopt the inspiring credibility update algorithm in RateWeb framework [7] and clarify a number of sub-algorithms which are left open by the authors.

The critical difference between experimental settings in [7] and our simulation settings is the existence of bootstrapping. While we assign a neutral default value to newcomers (initiating A to 0.5) as a simple bootstrapping mechanism integrated into the computing engine and let every user start from scratch. The work in [7] assumes that the bootstrapping has been done and the system is running with correct reputation scores and credibilities. That is a strong assumption and makes their results better than ours. Nevertheless, the results in Sect. 6 are still interesting to us and force us to modify the computing engine or look for a new one.

8 Conclusions and Future Work

References

1. http://www.ebay.com/.
2. http://www.amazon.com.
3. Chrysanthos Dellarocas. Online reputation systems: How to design one that does what you need. *MIT Sloan management review*, 51(3), spring 2010.
4. Audun Josang and Jennifer Golbeck. Challenges for robust trust and reputation systems. In *Proceedings of the 5th International Workshop on Security and Trust Management*, Saint Malo, France, September 2009.
5. Audun Josang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision support systems*, 43(2):618–644, March 2007.
6. J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.
7. Zaki Malik and Athman Bouguettaya. Rateweb: Reputation assessment for trust establishment among web services. *The international journal on very large data bases*, 18(4):885–911, August 2009.
8. Sergio Marti. *Trust and Reputation in Peer-to-Peer Networks*. PhD thesis, Stanford University, Stanford InfoLab, May 2005.
9. Paul Resnick, Richard Zeckhauser, Eric Friedman, and Ko Kuwabara. Reputation systems. *Communications of the ACM*, pages 45–48, December 2000.