Identifying and Exploiting Trustable Users with Robust Features in Online Rating Systems

Hyun-Kyo Oh[1] and Sang-Wook Kim[1]
[1] Department of Computer and Software, Hanyang University
Seoul – South Korea
[e-mail: rapkyo@agape.hanyang.ac.kr; wook@hanyang.ac.kr]
*Corresponding author: Sang-Wook Kim

Received September 25, 2016; revised December 23, 2016; accepted January 24, 2017; published April 30, 2017

Abstract

When purchasing an online product, a customer tends to be influenced strongly by its reputation, the aggregation of other customers' ratings on it. The reputation, however, is not always trustable since it can be manipulated easily by attackers who intentionally give unfair ratings to their target products. In this paper, we first address identifying trustable users who tend to give fair ratings to products in online rating systems and then propose a method of computing true reputation of a product by aggregating only those trustable users' ratings. In order to identify the trustable users, we list some candidate features that seem related significantly to the trustworthiness of users and verify the robustness of each of the features through extensive experiments. By finding and exploiting these robust features, we are able to identify trustable users and to compute true reputation effectively and efficiently based on fair ratings of those trustable users.

Keywords: Trust, False reputation, Robustness, Robust Features, Unfair ratings, Attackers

This research was supported by (1) Semiconductor Industry Collaborative Project between Hanyang University and Samsung Electronics Co. Ltd. and (2) the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. NRF-2014R1A2A1A10054151)

https://doi.org/10.3837/tiis.2017.04.019 **ISSN : 1976-7277**

1. Introduction

With a few clicks of a mouse, we can buy any products on the Internet. Behind these convenient purchases, there are considerable risks on uncertain products.

Customers can share their experiences on purchased products with other potential buyers via evaluation. The simplest way for consumers to express their levels of satisfaction with their purchases is to participate in online ratings. The overall buyers' satisfaction on a product can be quantified as the aggregated score of all the ratings given to the product and becomes available to other potential buyers. In this paper, we call this aggregated score for the product its *reputation*. The reputation is known to significantly influence other potential buyers to decide their final purchasing [5][7][14][16][18].

If a large number of buyers take part in giving ratings to a product with honesty, the reputation is trustable and really helps potential buyers to purchase products [8][11][13][20]. Unfortunately, the reputation does not seem that trustable when *reputation attackers* (*RAs*), who intentionally give high (or low) ratings to target products, participate in ratings [1][3][9][15][17][21][24].

False reputation, the reputation distorted by RAs, provides wrong information on products to potential buyers, thereby causing potential buyers to make a wrong decision on purchasing products [17]. Thus, we need to exclude as many RAs as possible but to include as many *trustable users* (*TUs*), who rate products with honesty, as possible in the process of reputation computation to get *true reputation* on products.

In order to identify TUs in online rating systems, we need to find good features to determine their trustworthiness by adopting the techniques from feature engineering [25], which is the process of analyzing domain specific data to extract some fine-tuned features. The features of user trustworthiness obtained by our feature engineering approach could be fundamental to practical applications such as user ranking, outlier detection, and reliable reputation. Because building feature models and suggesting good features in a specific domain is not only difficult but also time-consuming work [25], we focus on feature engineering to extract good features to determine trustworthiness of users in online rating systems.

Coming up with the features related to user trustworthiness, we analyze user rating behaviors to be able to identify their relationships to the features in question. For example, if the number of ratings given by a user is selected as such a feature, this may indicate that a user having more ratings is more likely to be considered as a TU than a user having less ratings. If we successfully find these features, we can identify TUs by using them and compute true reputations on products from only those TUs. As a result, the true reputation could be obtained by including TUs and excluding RAs.

The goal of RAs is to boost the reputation of a product by giving unfairly high ratings or to deteriorate the reputation of a product by giving unfairly low ratings. In the recommender systems area, RAs are called shilling attackers. Recommender systems are known vulnerable to shilling attackers who are malicious users to insert fake profiles [2][3][6][12][15][22][23] .

In this paper, we first exploit existing features used to identify RAs to measure the trustworthiness of users. For example, the rating deviation from mean agreement (RDMA), known as one of statistical features of RAs, increases when a user has few ratings and his

rating is deviated from the product reputation while that product has been evaluated by only a few users [3]. A user with high RDMA is regarded as an RA. Conversely, if a user has low RDMA, he should be regarded as a TU since he may have many ratings whose scores are closed to the product reputation evaluated by a number of users.

According to the result of social science studies obtained from analyzing the characteristics of online information, the reliability of online information causes user behaviors as follows [4][19]. The reliability tends to increase when an information producer has no bias but maintains an objective perspective (i.e., *objectivity*) and has a consistent viewpoint (i.e., *consistency*) when creating his information. Also, it tends to increase when an information producer frequently interacts with the users who access the information produced by him (i.e., *activity*). Thus, we first add the notions of activity, objectivity, and consistency as candidate features to recognize TUs.

In summary, we set candidate features that seem beneficial to determining the trustworthiness of users in online rating systems. The candidate features consist of the features which are used to identify RAs in the recommender systems area and the features of the activity, objectivity, and consistency. We normalize a value of each feature for a user to get a score expressing his trustworthiness. The effectiveness of each feature is measured by learning existing data to understand the correlation between the trustworthiness and the feature. To find the degree of relevance of a feature to the trustworthiness, we construct a variety of scenarios where false reputation occurs by injecting different types of RAs into our experimental data.

We select high ranked users with relatively high values of a feature and then compute the reputation on products by using only those high ranked users according to the feature. If the reputation thus obtained is closer to the ground truth, we consider the feature highly relevant to the user' trustworthiness and thus very robust to RAs. As ground truth, we use the reputation on products without injecting RAs.

The contributions of this paper are summarized as follows. First, we examine a variety of features extracted based on the results by analyzing user rating behaviors and from the social science studies. Second, we verify correlation between the trustworthiness and each of user features. Third, we suggest robust features to be used for true reputation against RAs and also the number of users necessary for computing true reputation through extensive experiments.

2. Related Work

In online rating systems, people do not want to listen to the opinions of attackers. Increasing is the demand to build the predictive models for distinguishing normal users from attackers as well as for identifying trustable users [17]. Creating domain-specific features is important to build predictive models and affects the prediction results. Feature engineering is the process of analyzing domain specific data to extract fine-tuned features [25]. The fine-tuned features help to build simpler and more representable predictive models, thereby often producing better prediction results [26][27]. Because of the importance of finding such features, this paper focuses on feature engineering to extract and evaluate those features related to the trustworthiness of users in online ratting systems.

Numerous studies have been conducted to improve the trustworthiness of information in online shopping malls by analyzing the rating behaviors of attackers who have participated in the rating system to manipulate the information. The "shilling attackers" are malicious users who try to insert fake profiles consisting of manipulated ratings into recommendation systems in order to increase the popularity of target items. Since the first introduction of shilling attacks [10], many different types of shilling attackers, such as random attackers, average attackers, segment attackers, bandwagon attackers, and love/hate attackers have been proposed in the literature [2][3][15]. Keeping pace with the researches on shilling attacks, researches on extracting features that characterize shilling attackers have also been performed [2][3][9][15][24].

Oh et al. [17] introduced dangerous situations where false reputation occurs and characterized dangerous users who cause false reputation. In order to solve the problem of false reputation, they proposed TRUEREPUTATION, an algorithm that iteratively adjusts the reputation based on the confidence of customer ratings. To determine the confidence of a rating, they exploited three key factors of activity, objectivity, and consistency. These three factors are well known in social science to be suitable for determining the confidence. They reinterpreted these factors in the context of online ratings.

Creating the features of user trustworthiness and computing true reputation can be used fundamentally in various practical applications. Hong et al. [28][29][30] studied reputation-based cooperation mechanism to enhance cooperative behaviors among self-interested individuals. By adopting our feature models to find trustable users, a certain reliable group among selfish individuals can be found where they can share information about their trustworthiness to improve their cooperation. Furthermore, adopting the concept of trustworthiness to practical applications such as collaborative recommender systems [31], personalized search engines [33], and discriminative classifiers [32] would be helpful to improve the performance and effectiveness of those applications.

3. User features

We introduce three equations on the activity, objectivity, and consistency of users. We also present various features that characterize RAs and propose a normalization method to transform the features of RAs into the features indicating the trustworthiness of users. We assume that the rating patterns of RAs are quite opposed to those of TUs. The descriptions and equations of all the features are shown as follows:

Activity [17]: A user who has more ratings could be considered more active. The user's activity, denoted by A_u, is quantified by the frequency of his ratings.

$$A_u = \Psi(|\mathbf{R}_u|, \alpha^{activity}, \mu^{activity})$$

where

$$\Psi(|\mathbf{R}_u|, \alpha^{activity}, \mu^{activity}) = \frac{1}{1+e^{-\alpha^{activity}(|\mathbf{R}_u|-\mu^{activity})}}$$

where $\mathbf{R}_u$ is a set of ratings by user u. The Ψ function is a sigmoid function for normalization of $|\mathbf{R}_u|$ to keep the returned value in the range of [0, 1]. Parameters $\alpha^{activity}$

and $\mu^{activity}$ determine the slope and adjust the midpoint of the curve of Ψ, respectively. In order that Ψ should be closer to 1 when $|\mathbf{R}_u|$ gets very large, $\alpha^{activity}$ should be a positive number. In order to distribute $|\mathbf{R}_u|$ evenly in the range of [0, 1], we use $\alpha^{activity}$ = 0.02, determined by our experiments. We set $\mu^{activity}$ to be the average number of ratings per user.

Objectivity [17]: The user's objectivity, denoted by O_u, is the normalized average of the objectivities of the ratings by that user. The objectivity of a rating, O_r, indicates that the user with a rating *r* has performed a more objective evaluation on item *m* when O_r is closer to 0. O_r is computed based on the reputation, denoted by $\bar{r}_m$, and the standard deviation, denoted by s_m, as follows:

$$O_r = |\frac{r - \bar{r}_m}{s_m}|$$

O_u is computed by the average of the objectivities in the ratings by that user. If it is closer to 0, the user is considered more objective.

$$O_u = \frac{1}{|\mathbf{R}_u|}\sum_{r\in\mathbf{R}_u} O_r$$

In order to regard O_u as the trustworthiness of a user, we need o_u^*, computed by normalizing O_u. We consider the user whose o_u^* is '1' to be the most objective user.

$$o_u^* = \Psi(O_u,\ \alpha^{objectivity}, \mu^{objectivity})$$

$\alpha^{objectivity}$ needs to be a negative number in order that Ψ should be close to 1 when O_u gets closer to 0. Based on our preliminary experiments, we set $\alpha^{objectivity}$ to be -2.5 and $\mu^{objectivity}$ to be the average of all users' objectivities.

Consistency: A user whose rating behavior conforms to the most common behavior in the online rating system could be considered consistent. The user's consistency, denoted by C_u, is defined by the variation in the objectivities of his ratings.

$$C_u = \frac{1}{|\mathbf{R}_u|}\sum_{r\in\mathbf{R}_u}(O_r - O_u)^2$$

A user who has many ratings is difficult to keep his consistency. The *Weighted Consistency* (*WC*) places a high weight on a user if he keeps his consistency despite many ratings. It differs from consistency only in that the number of his ratings is square rooted in the denominator outside the sum, thus relatively increasing the weight associated with users having many ratings.

$$C_u = \frac{1}{\sqrt{|\mathbf{R}_u|}}\sum_{r\in\mathbf{R}_u}(O_r - O_u)^2$$

Based on our preliminary experiments, we set $\alpha^{consistency}$ to be -10 and α^{WC} to be -8. $\mu^{consistency}$ is set as the average of all users' C_u and μ^{WC} is set as the average of all users' WC_u.

From now on, various features representing the rating behaviors of RAs are introduced. As explained earlier, we assume the behavior of TUs is directly opposed to that of RAs. When the values of features to detect RAs get closer to 0, we need to make the values to detect TUs closer to 1.

Rating Deviation from Mean Agreement (*RDMA*) [3]. RDMA measures the average disagreement of a rating of a user with those of other users for every item, weighted by the inverse of the number of ratings for that item. The feature is computed as follows:

$$RDMA_u = \frac{\sum_{m \in \mathbf{N}_u} \frac{|r_{u,m} - \bar{r}_m|}{l_m}}{|\mathbf{N}_u|}$$

where $\mathbf{N}_u$ is a set of items rated by user *u*, $r_{u,m}$ is the rating given by user *u* to item *m*, l_m is the number of ratings assigned for item *m* by all users, and $\bar{r}_m$ is the average of these ratings, the reputation of the item. In order to use $RDMA_u$, we need to normalize it as follows:

$$RDMA_u^* = \Psi(RDMA_u, \alpha^{rdma}, \mu^{rdma})$$

α^{rdma} needs to be a negative number in order that Ψ should be close to 1 when $RDMA_u$ gets closer to 0. Based on our preliminary experiments, we set α^{rdma} to be -250. μ^{rdma} is set as the average of all users' $RDMA_u$.

Weighted Degree of Agreement (*WDA*) [15]. WDA, derived from RDMA, measures the sum of the differences of a user's ratings from the items' reputations divided by the number of ratings on each item. The feature is computed as follows:

$$WDA_u = \sum_{m \in \mathbf{N}_u} \frac{|r_{u,m} - \bar{r}_m|}{l_m}$$

$$WDA_u^* = \Psi(WDA_u, \alpha^{wda}, \mu^{wda})$$

Based on our preliminary experiments, we set α^{wda} to be -1.5. μ^{wda} is set as the average of all users' WDA_u.

Weighted Deviation from Mean Agreement (*WDMA*) [15]. WDMA, derived from RDMA, places a higher weight on rating disagreement in parse items by squaring the denominator inside the sum, thus reducing the weight associated with the items rated by many users. The feature is computed as follows:

$$WDMA_u = \frac{\sum_{m \in \mathbf{N}_u} \frac{|r_{u,m} - \bar{r}_m|}{l_m{}^2}}{|\mathbf{N}_u|}$$

$$WDMA_u^* = \Psi(WDMA_u, \alpha^{wdma}, \mu^{wdma})$$

Based on our preliminary experiments, we set α^{wdma} to be -2,000 and μ^{wdma} to be the average of all users' $WDMA_u$.

Degree of Similarity with Top Neighbors (*DegSim*) [3]. DegSim captures the average similarity of a user's *k* nearest neighbors. DegSim intends that RAs are likely to have a higher similarity with their top-k closest neighbors than authentic users. The feature is computed as follows:

$$DegSim_u = \frac{\sum_{V \in neighbors(u)} W_{u,v}}{k}$$

where $W_{u,v}$ is the similarity between users *u* and *v* computed by Pearson's correlation and *k* is the number of neighbors.

$$degsim_u^* = \Psi(degsim_u, \alpha^{degsim}, \mu^{degsim})$$

Based on our preliminary experiments, we set α^{degsim} to be -5. μ^{degsim} is set as the average of all users' $DegSim_u$.

$DegSim'$[15], a variation of DegSim, penalizes the average similarity if a neighbor shares fewer than *d* ratings in common. For normalization, $DegSim'$ use different parameter settings, $\alpha^{degsim'}$ and $\mu^{degsim'}$ according to the value of *d*[1].

Length Variance (*LengthVar*) [15]. LengthVar is designed to detect those users who have abnormally lots of ratings than authentic users by capturing how much the length (i.e., the number of ratings) of a user's profile differs from the average length among all users in a database. The feature is computed as follows:

$$LengthVar_u = \frac{||\mathbf{N}_u| - |\bar{\mathbf{N}}||}{\sum_{p \in \mathbf{U}} (|\mathbf{N}_p| - |\bar{\mathbf{N}}|)^2}$$

$$LengthVar_u^* = \Psi(LengthVar_u, \alpha^{lengthvar}, \mu^{lengthvar})$$

where **U** is a set of all users and **N** is a set of all items. Based on our preliminary experiments, we set $\alpha^{lengthvar}$ to be -200,000 and $\mu^{lengthvar}$ to be the average of all users' $LengthVar_u$.

---

[1] The detailed parameter settings of $DegSim'$ are shown in the section of experimental results.

Filler Mean Variance (*FMV*) [15]. To measure FMV, $\mathbf{P}_{u,T}$ and $\mathbf{P}_{u,F}$ need to be defined. $\mathbf{P}_{u,T}$ is a set containing the items (suspected to be targets) rated by a user u, and $\mathbf{P}_{u,F}$ is a set containing all the other items except $\mathbf{P}_{u,T}$. FMV is designed to compute the mean variance between the non-target items in $\mathbf{P}_{u,F}$, the reputation for each item. The feature is computed as follows:

$$FMV_u = \frac{\sum_{m \in \mathbf{P}_{u,F}} (r_{u,m} - \bar{r}_u)^2}{|\mathbf{P}_{u,F}|}$$

FMV is computed twice, first when $\mathbf{P}_{u,T}$ consists of items assigned with the maximum rating, and second when $\mathbf{P}_{u,T}$ consists of items assigned with the minimum rating.

$$FMV_u^* = \Psi(FMV_u, \alpha^{fmv}, \mu^{fmv})$$

where parameters, α^{fmv} and μ^{fmv}, are assigned differently according to the goal of user u ("push" or "nuke"). "Push" means that the attacker gives a maximum rating to promote the item while "nuke" means that the attacker gives a minimum rating to demote the item. If we set the goal of a user as "push", we set α^{fmv} to be 3. If we set the goal of the user as "nuke", we set α^{fmv} to be 0.5. μ^{fmv} is assigned with the average of all users' FMV_u.

Filler Mean Difference (*FMD*) [15]. FMD is the average of the absolute differences between the user's rating and the average rating over all the items in $\mathbf{P}_{u,F}$ (rather than the squared value in FMV). The parameters, α^{fmd} and μ^{fmd}, are assigned differently according to the goal of a user u. α^{fmd} and u^{fmd} are assigned with exactly the same values as α^{fmv} and μ^{fmv}. u^{fmd} is assigned with the average of all users' FMD_u.

Profile Variance (*PV*) [15]. PV is simply the variance over all the ratings assigned to all the items in $\mathbf{P}_{u,F}$. If we set the goal of the user as "push", we set α^{PV} to be 2.5. If we set the goal of the user as "nuke", we set α^{PV} to be 2.5. μ^{PV} is assigned with the average of all users' PV_u.

Filler Mean Target Difference (*FMTD*) [15]. FMTD is designed to detect RAs who target a specific group of items. The attackers give the maximum ratings to the target items, $\mathbf{P}_{u,T}$ and randomly chosen ratings to other non-target items called the filler items, $\mathbf{P}_{u,F}$. FMTD is the difference between the average of the ratings on $\mathbf{P}_{u,F}$ and the average of the ratings on $\mathbf{P}_{u,F}$. The feature is computed as follows:

$$FMTD_u = \left|\left(\frac{\sum_{m \in \mathbf{P}_{u,F}} r_{u,m}}{|\mathbf{P}_{u,T}|}\right) - \left(\frac{\sum_{m \in \mathbf{P}_{u,F}} r_{u,m}}{|\mathbf{P}_{u,F}|}\right)\right|$$

where $r_{u,m}$ is the rating given by user u to item m. Based on our preliminary experiments, we set α^{fmtd} to be -1.5. μ^{fmtd} is assigned with the average of all users' $FMTD_u$.

Target Model Focus (*TMF*) [15]. A single attacker cannot actually influence the whole recommender system. A substantial group of attackers need to give the manipulated ratings to the target item to achieve what they aim at. To measure TMF for a user, we need to firstly define F_m (Focus value of item *m*). F_m is the degree of a given item *m* being targeted by attackers. TMF is the focus value of the item at which a given user has targeted. The item of the highest TMF would be attacked. The feature is computed as follows:

$$TMF_u = \max_{m \in \mathbf{P}_{u,T}} F_m$$

$$TMF_u^* = \Psi(TMF_u, \alpha^{tmf}, \mu^{tmf})$$

If we set the goal of the user as "push", we set α^{tmf} to be -0.8. If we set the goal of the user as "nuke", we set α^{tmf} to be -0.2. μ^{tmf} is assigned with the average of all users' TMF_u.

4. Performance evaluation

4.1 Experimental Setup

In online rating systems, it is important to provide true reputation of a product by aggregating trustable users' ratings. Our goal is to employ user features for including as few RAs and many TUs as possible in computing true reputation. The state of the art algorithm, TRUE-REPUTATION [17], uses all the users to calculate the reputations of products. Consequently, TRUE-REPUTATION can allow the ratings of RAs to be included in the calculation of a reputation of a product. Also, TRUE-REPUTATION performs the two steps (calculating the confidence of user ratings and adjusting the reputation of products) iteratively until all the reputations converge to a stable state. On the other hand, our approach uses only a small part of users that are TUs identified by our robust features. Thus, our approach enables to avoid the influence of unfair ratings of RAs. Also, our approach (just aggregating the ratings of TUs) does not require a complex process, thus saving a lot of time in computation.

To verify the effectiveness of our features identified, we generated various types of RAs and constructed scenarios that produce false reputation by injecting the RAs into the MovieLens [2] dataset. In our experiments, newly-released or unpopular movies having 90~110 are considered to be in a dangerous situation, where false reputation can easily occur. Among them, the movies with reputation higher than 3.53 (average reputation over all movies in MovieLens) are targeted for "nuke" and the others are for "push". We assume that users in MovieLens are ordinary users who rate the movies reasonably.

[2] The MovieLens dataset has 100,000 ratings with the scale from 1 to 5 for 1,682 movies by 943 users.

We generated six types of RAs, which can be further divided into "push" or "nuke" to make target items more or less likely to be recommended, respectively. The RA profile consists of the following components: a target item (m_T), a set of selected items ($\mathbf{M}_S$) that are selected differently depending on the attack types, a set of filler items ($\mathbf{M}_F$) that are chosen randomly, and a set of unrated items ($\mathbf{M} = \mathbf{M} - (m_T \cup \mathbf{M}_S \cup \mathbf{M}_F$). The six types of RAs are summarized in **Table 1**.

Table 1. Summary of RAs

Attack Model	Attack Type	$\mathbf{M}_S$	$\mathbf{M}_F$,	$\mathbf{M}_\emptyset$	m_T
Average	push/ nuke	Not used	filler items, ratings assigned with normal distribution around item mean	The items of $\mathbf{M}$ that are not in the union of m_T and $\mathbf{M}_F$, no ratings	The target item, r_{max}/r_{min}
Random	push/ nuke	Not used	filler items, ratings assigned with normal distribution around system mean	The items of $\mathbf{M}$ that are not in the union of m_T and $\mathbf{M}_F$, no ratings	The target item, r_{max}/r_{min}
Selected Popular	push	Popular items, r_{max}	filler items, ratings assigned with normal distribution around system mean	The items of $\mathbf{M}$ that are not in the union of m_T, $\mathbf{M}_S$, and $\mathbf{M}_F$, no ratings	The target item, r_{max}
Segment	push	Items chosen to define the segment, r_{max}	filler items, r_{min}	The items of $\mathbf{M}$ that are not in the union of m_T, $\mathbf{M}_S$, and $\mathbf{M}_F$, no ratings	The target item, r_{max}
Love/Hate	nuke	Not used	filler items, r_{max}	The items of $\mathbf{M}$ that are not in the union of m_T and $\mathbf{M}_F$, no ratings	The target item, r_{min}
Reverse Selected Popular	nuke	Widely disliked items, r_{min}	filler items, ratings assigned with normal distribution around system mean	The items of $\mathbf{M}$ that are not in the union of m_T, $\mathbf{M}_S$, and $\mathbf{M}_F$, no ratings	The target item, r_{min}

For generating RAs, we varied three factors: (1) the numbers of RAs added to MovieLens, (2) the types of RAs, and (3) the rating frequencies of RAs.

First, we varied the number of RAs attacking a target movie from 10% of its total number of ratings to 30% in increment of 10%. Second, we used six different types of RAs. Third, we chose 10 target movies and inserted each type of RAs to them. The number of ratings for each type of RA is set as 50, 100, and 150.

In order to verify the effectiveness of each feature, we computed the scores of all the features of users who gave ratings to the target item under the scenarios where false

reputation occurs. For each feature, users are ranked according to their score of the feature. By using the top-*k* percent users, the reputation of the target item can be temporarily computed. As this reputation is closer to the ground truth, we consider the feature to be more related to the user trustworthiness (i.e., more robust). As ground truth, we used the reputation of the target item computed before injecting RAs.

When computing the reputation only by the selected users, we will have a natural question "*how many users do we choose*?"

True reputation can be obtained by including as many TUs as possible but excluding as many RAs as possible. When computing the reputation on an item, we varied the portion of users used in computing reputation from 10% to 100% in step of 10%.

Basically, the effectiveness of each feature is evaluated by the difference between the reputation with RAs and the reputation without RAs, as shown in Equation 1.

$$Reputation\ Change\ Rate\ (RCR) = \frac{|R\ with\ RAs - R\ without\ RAs|}{R\ without\ RAs}, (R\text{: } reputation) \quad (1)$$

As RCR based on selected users gets smaller, we regard the feature to be more robust to RAs.

In order to find most robust features to get true reputation, features need to be compared with one another. As described above, the effectiveness of a feature is basically measured by RCR. However, the reputation can be varied according to the number of selected users.

When measuring the effectiveness of a feature, the smaller users participate in computing the reputation, the more weight we place on RCR. The effectiveness of a feature, *Feature Trustworthiness (FT)*, is defined as follows:

$$Feture\ Trustworthiness\ (FT) = RCR\left(R\ with\ top\ 10\%\ of\ \mathbf{U}_{m,f}^{rank}\right) + \frac{1}{2}\ RCR\left(R\ with\ top\ 20\%\ of\ \mathbf{U}_{m,f}^{rank}\right) + \ldots + \frac{1}{10}\ RCR\left(R\ with\ top\ 100\%\ of\ \mathbf{U}_{m,f}^{rank}\right) \quad (2)$$

where FT is computed based on a set of users who have rated item *m*, $\mathbf{U}_{\mathrm{m}}$. $\mathbf{U}_{m,f}^{rank}$ is a set of users in $\mathbf{U}_{\mathrm{m}}$ ranked according to the score of feature *f*. The lower FT of a feature is, the more robust the feature is.

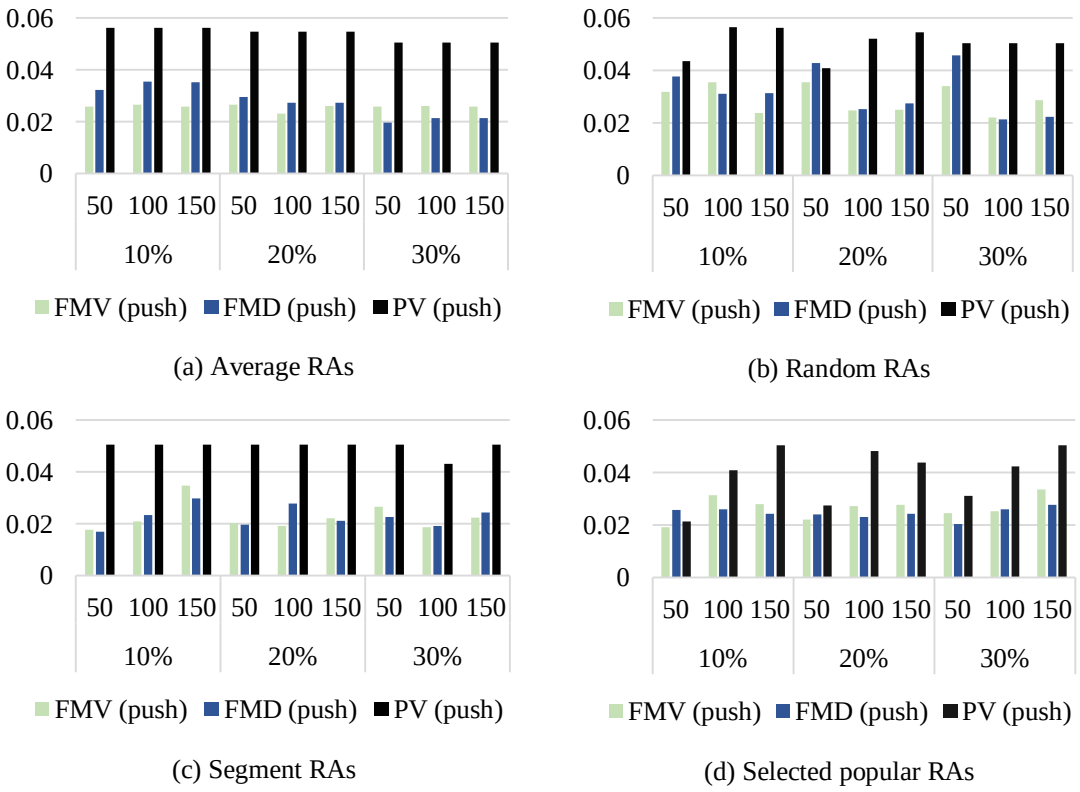
4.2 Results and Analyses

Through experiments, we try to answer the following questions:

- How much the effectiveness of features varies according to the change of the number of RAs, the rating frequency by RAs, the purpose of RAs, and the type of RAs?
- In order to get trust reputation, how many users need to be involved in the process of computing the reputation?

- What are robust features?
- How much are the combinations of multiple features beneficial to getting true reputation?

The parameter values for RAs are set as follows: the number of RAs is set to be 10%, 20%, and 30% of the total number of ratings on the item; the rating frequency by RAs is set to be 50, 100, and 150. **Fig. 1** shows the effectiveness of some features (FMV, FMD, and PV selected as robust features via our experiments in Section 4) according to the number of RAs and the rating frequency under attacks by different types of push RA (i.e., Average RA, Random RA, Segment RA, and Selected Popular RA). The *y*-axis represents RCR measured by top 50% of $\mathbf{U}_{m,f}^{rank}$, where the feature is FMV, FMD, or PV on an item. The *x*-axis represents combinations of the number of RAs (10%, 20%, 30%) and the rating frequency (50, 100, 150). According to these experimental results in **Fig. 1**, the effectiveness of each feature is shown similar in the case of the same type of RAs regardless of different parameter settings. For each RA, we thus show the average of RCRs obtained from nine experimental results performed by different parameter settings.



(a) Average RAs

(b) Random RAs

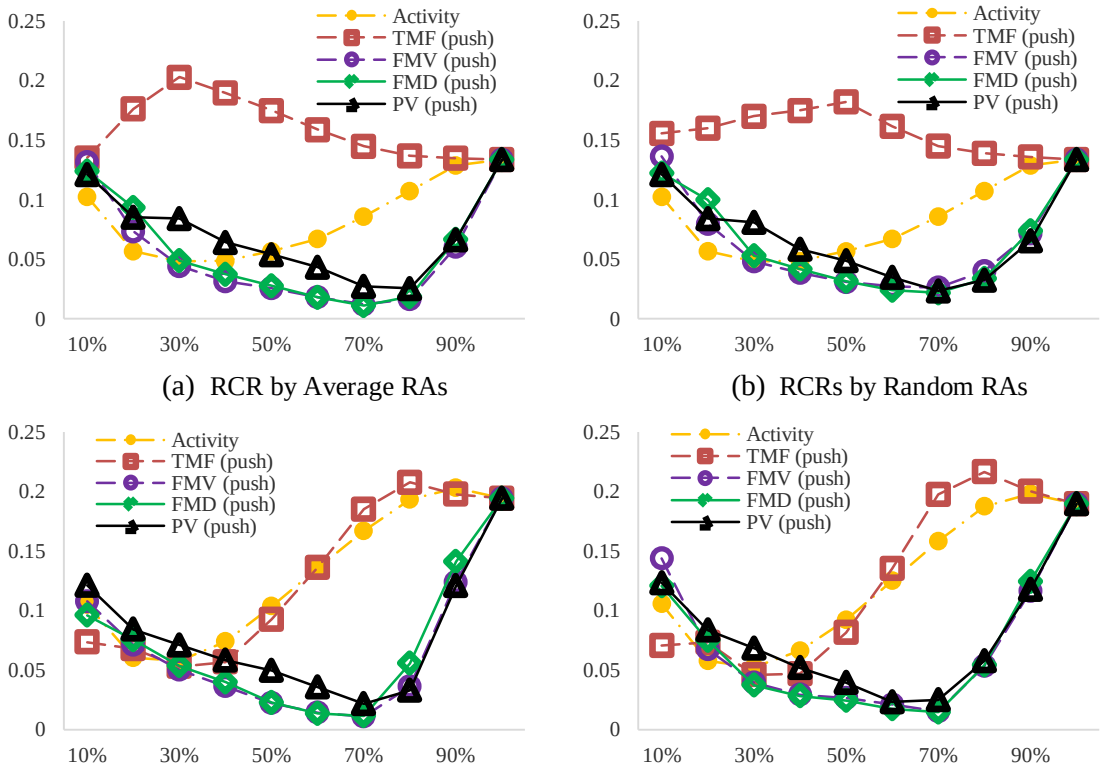
(c) Segment RAs

(d) Selected popular RAs

Fig. 1. Reputation change by the number of RAs and the rating frequency

The figures in **Fig. 2**, **Fig. 2-(a)** to **Fig. 2-(d)**, show the effectiveness of features under the attacks by different types of push RA, i.e., Average RA, Random RA, Segment RA, and Selected Popular RA, respectively. In the case of Degsim, we used k = 25 and in the case of $DegSim'$, we also used k = 25 and set d to be 10, 20, 30, 40, and 50[3]. The figures show the top 5 features whose FTs are smaller than those of other features. The x-axis represents the ratio of users used to the total number of users on an item and the y-axis represents RCR.

Fig. 2-(a) to **Fig. 2-(d)** show that overall FMV, FMD, and PV perform reasonably. The RCRs of the three features increase up to 70%, however, sharply deteriorate after that. In the cases of FMV, FMD, and PV, RCR gets higher when more than 70% users are used in computing reputation because RAs are involved in reputation computation. Overall, when online rating systems face attacks by push RAs, we can get the reputation close to true reputation by using the top 70% users ranked by FMV.



(a) RCR by Average RAs (b) RCRs by Random RAs

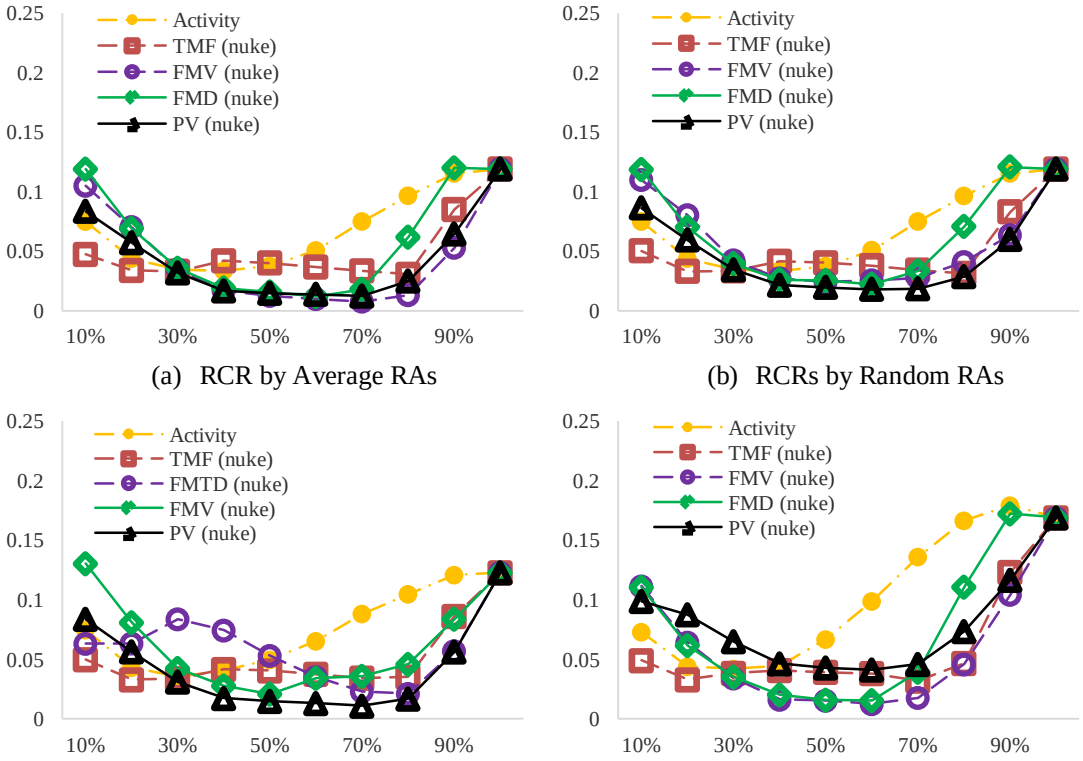
(c) RCR by Segment RAs (d) RCRs by Selected Popular RAs

Fig. 2. Reputation change by push RAs

[3] Based on our preliminary experiments, we set $\alpha^{degsim'}$ to be -5, -7.5, and -10 according to different values of d. We assign $\mu^{degsim'}$ with the average of all users' $DegSim'_u$.

The figures in **Fig. 3**, **Fig. 3-(a)** to **Fig. 3-(d)**, show the effectiveness of the top 5 features under attacks by different types of nuke RA, i.e., Average RA, Random RA, Love/ Hate RA, and Reverse Selected Popular RA, respectively.

Fig. 3-(a) to **Fig. 3-(d)** show that, overall, the top 5 features work well. In the case of TMF, it shows a very low RCR even though a small number of users are used in computing the reputation. In the cases of FMV, FMD, and PV, RCR increases when more than 70% users are used in computing the reputation because RAs start to be involved in computation.



(a) RCR by Average RAs

(b) RCRs by Random RAs

(c) RCR by Love/Hate RAs

(d) RCRs by Reverse Selected Popular RAs

Fig. 3. Reputation change by nuke RAs

By comparing FTs of all features, we examine which features are robust under attacks by push RAs and nuke RAs. **Table 2** shows the changes of RCRs, under attacks by push RAs, according to the percent of selected users & the used feature and the ranks of features according to FT. We shade the cells whose values are smaller than *0.1627*, which is RCR when the reputation is computed by using all the users. FMD, FMV, PV, Activity, and TMF perform reasonably in most cases and Degsim, $DegSim'$, RDMA, and WDMA show poor RCRs in most cases.

Table 2. Effectiveness of all the features by push RAs

	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%	*FT*	RANK
Activity	0.1038	0.0578	0.0517	0.0593	0.0772	0.0987	0.1241	0.1487	0.1646	0.1627	0.0268	4
Objectivity	0.6954	0.6042	0.4847	0.3825	0.3121	0.2623	0.2262	0.2000	0.1834	0.1627	0.1455	11
Consistency	0.6671	0.5637	0.4553	0.3692	0.3129	0.2721	0.2440	0.2049	0.1848	0.1627	0.1367	9
WC	0.5425	0.4647	0.3786	0.3095	0.2663	0.2340	0.2800	0.1902	0.1783	0.1627	0.1135	7
RDMA	0.8143	0.7303	0.5531	0.4172	0.3286	0.2718	0.2229	0.1911	0.1740	0.1627	0.1670	16
WDMA	0.8153	0.7320	0.5538	0.4109	0.3285	0.2709	0.2293	0.1956	0.1776	0.1627	0.1673	17
WDA	0.6809	0.6744	0.5597	0.4350	0.3464	0.2786	0.2336	0.1990	0.1739	0.1627	0.1523	13
DegSim (k=25)	0.8112	0.7363	0.5609	0.3978	0.3099	0.2565	0.2229	0.1958	0.1773	0.1627	0.1663	15
DegSim' (d=10, k=25)	0.8115	0.7450	0.5901	0.4383	0.3538	0.3026	0.2627	0.2207	0.1831	0.1627	0.1713	19
DegSim' (d=20, k=25)	0.7941	0.7234	0.5773	0.4412	0.3652	0.3026	0.2567	0.2205	0.1865	0.1627	0.1683	18
DegSim' (d=30, k=25)	0.7087	0.6486	0.5401	0.4421	0.3710	0.3106	0.2569	0.2237	0.1875	0.1627	0.1551	14
DegSim' (d=40, k=25)	0.6441	0.6058	0.5418	0.4480	0.3674	0.3132	0.2593	0.2224	0.1878	0.1627	0.1467	12
DegSim' (d=50, k=25)	0.5880	0.5584	0.5221	0.4384	0.3614	0.3088	0.2581	0.2170	0.1865	0.1627	0.1376	10
LengthVar	0.1993	0.2882	0.3629	0.3733	0.3292	0.2808	0.2401	0.2075	0.1858	0.1627	0.0767	6
TMF (push)	0.1087	0.1193	0.1179	0.1170	0.1327	0.1479	0.1680	0.1749	0.1671	0.1627	0.0369	5
FMTD (push)	0.5551	0.5185	0.4354	0.3525	0.2839	0.2423	0.1964	0.1743	0.1721	0.1627	0.1230	8
FMV (push)	0.1298	0.0729	0.0456	0.0340	0.0264	0.0203	0.0162	0.0366	0.0931	0.1627	0.0232	2
FMD (push)	0.1158	0.0859	0.0482	0.0369	0.0267	0.0183	0.0148	0.0406	0.1015	0.1627	0.0227	1
PV (push)	0.1216	0.0843	0.0761	0.0582	0.0480	0.0343	0.0244	0.0371	0.0926	0.1627	0.0254	3

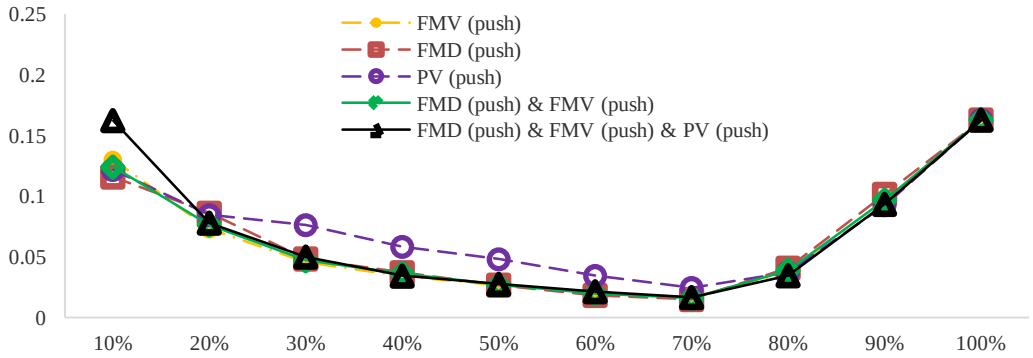
Table 3 shows the changes of RCRs, under attacks by nuke RAs, according to the percent of selected users & the used feature and the ranks of features according to FT. We shade the cells whose values are smaller than *0.1324*, which is RCR when the reputation is computed by using all the users. TMF, PV, Activity, FMV, and FMD work well in most cases and Degsim, $DegSim'$, RDMA, and WDMA show poor RCRs in most cases.

Table 3. Effectiveness of all the features by nuke RAs

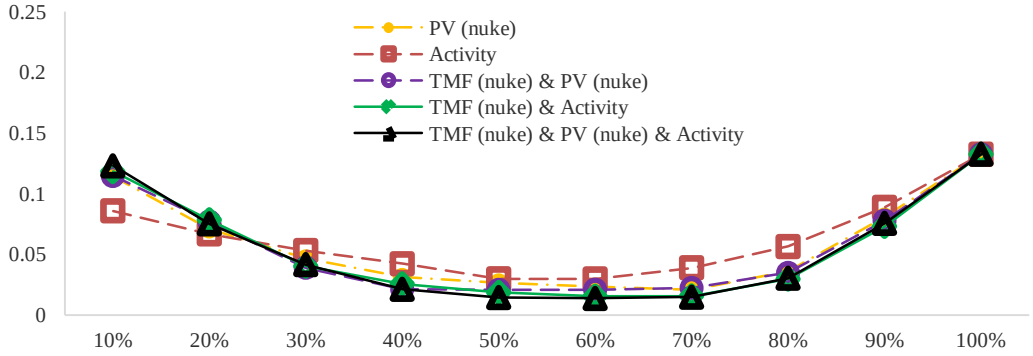
	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%	*FT*	RANK
Activity	0.0744	0.0437	0.0367	0.0380	0.0479	0.0664	0.0935	0.1160	0.1327	0.1324	0.0194	3
Objectivity	0.5300	0.4499	0.3713	0.2933	0.2395	0.2052	0.1797	0.1546	0.1360	0.1324	0.1107	10
Consistency	0.5795	0.4051	0.2944	0.2271	0.1821	0.1582	0.1388	0.1352	0.1286	0.1324	0.1050	9
WC	0.4226	0.3109	0.2386	0.1979	0.1735	0.1543	0.1219	0.1338	0.1330	0.1324	0.0815	8
RDMA	0.7158	0.6062	0.4436	0.3254	0.2546	0.2121	0.1785	0.1622	0.1431	0.1324	0.1409	16
WDMA	0.7215	0.6064	0.4395	0.3245	0.2540	0.2080	0.1834	0.1664	0.1441	0.1324	0.1414	17
WDA	0.5930	0.5587	0.4327	0.3295	0.2612	0.2157	0.1830	0.1609	0.1441	0.1324	0.1263	13
DegSim (k=25)	0.7249	0.6110	0.4501	0.3399	0.2712	0.2240	0.1895	0.1645	0.1439	0.1324	0.1434	19
DegSim' (d=10, k=25)	0.7213	0.6066	0.4453	0.3398	0.2802	0.2330	0.1980	0.1701	0.1529	0.1324	0.1433	18
DegSim' (d=20, k=25)	0.6742	0.5908	0.4410	0.3293	0.2641	0.2261	0.1997	0.1735	0.1515	0.1324	0.1370	15
DegSim' (d=30, k=25)	0.6407	0.5511	0.4276	0.3293	0.2617	0.2221	0.1964	0.1737	0.1536	0.1324	0.1311	14
DegSim' (d=40, k=25)	0.5881	0.5249	0.4244	0.3309	0.2614	0.2170	0.1918	0.1729	0.1538	0.1324	0.1243	12
DegSim' (d=50, k=25)	0.4914	0.4695	0.4068	0.3283	0.2585	0.2156	0.1903	0.1693	0.1544	0.1324	0.1110	11
LengthVar	0.2008	0.2404	0.3091	0.3069	0.2595	0.2221	0.1898	0.1639	0.1446	0.1324	0.0667	7
TMF (nuke)	0.0491	0.0332	0.0348	0.0415	0.0400	0.0376	0.0335	0.0360	0.0943	0.1324	0.0135	1
FMTD (nuke)	0.0663	0.0699	0.0986	0.1063	0.1051	0.1067	0.1087	0.1120	0.1227	0.1324	0.0256	6
FMV (nuke)	0.1141	0.0739	0.0384	0.0220	0.0183	0.0206	0.0223	0.0364	0.0762	0.1324	0.0206	4
FMD (nuke)	0.1217	0.0712	0.0474	0.0332	0.0318	0.0278	0.0388	0.0865	0.1366	0.1324	0.0237	5
PV (nuke)	0.0881	0.0652	0.0406	0.0255	0.0231	0.0216	0.0219	0.0360	0.0743	0.1324	0.0178	2

We next examine whether the combination of robust features is more effective in getting true reputation. **Fig. 4-(a)** shows the top 5 features selected by comparing the effectiveness of FMD, FMV, and PV (the best three features under attacks by push RAs) and all the possible combinations of the three features (FMV&FMD, FMV&PV, FMD&PV, FMV&FMD&PV). **Fig. 4-(a)** shows the top 5 features selected by comparing the

effectiveness of TMF, PV, and Activity (the best three features under attacks by nuke RAs) and the combinations of the three features (TMF&PV, TMF&Activity, PV&Activity, and TMF&PV&Activity). **Fig. 4** shows the effectiveness of the combinations of features slightly increases. However, it is difficult to say the combinations are really effective in getting true reputation.



(a) RCR using feature combinations by push RAs



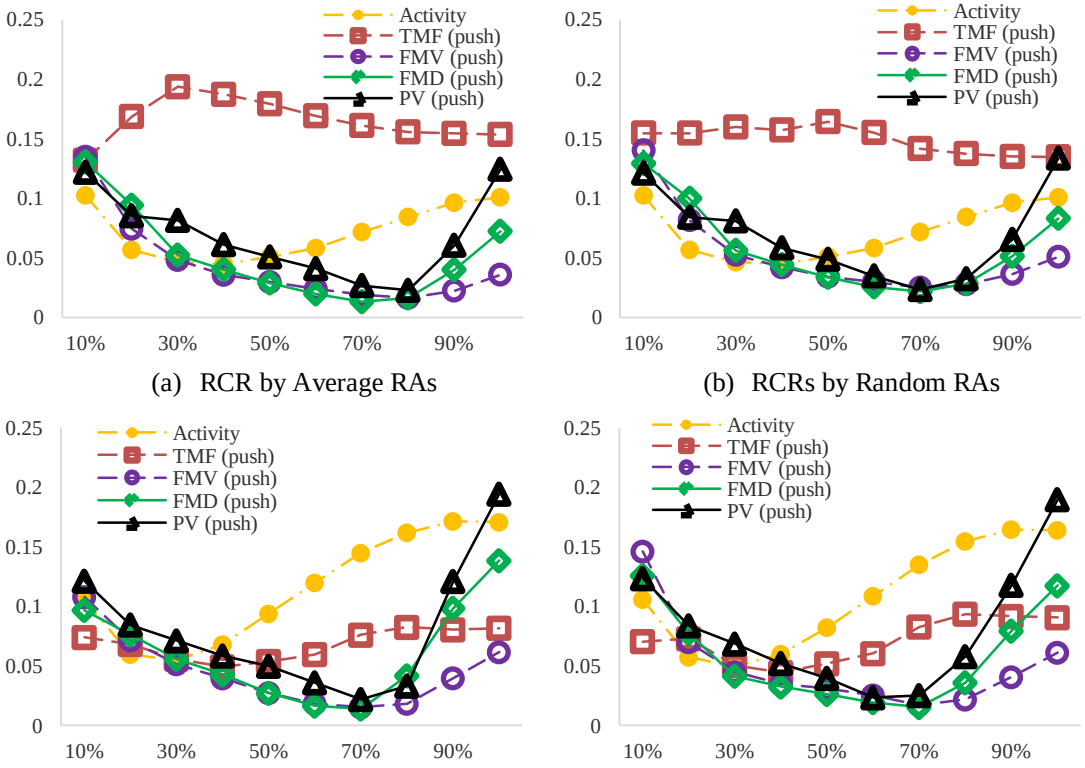
(b) RCR using features combinations by nuke RAs

Fig. 4. Reputation Change using feature combinations

In all previous experiments, we use the average (i.e., to assign the same weight to each rating) when aggregating ratings of (high ranked) users with relatively high values of a feature. When RAs are selected as the high ranked users, using the average may result in a false reputation. Therefore, we perform a series of experiments not using the average but giving a different weights to ratings (assigning a more weight to users with high values of a feature) in computing reputation. All the parameters for RAs and features are set as the same as in previous experiments. We also show the average RCRs obtained from nine experimental results performed by different parameter settings.

The figures in **Fig. 5**, **Fig. 5-(a)** to **Fig. 5-(d)**, show the effectiveness of features when using the weighted sum method under the attacks by different types of push RA. Comparing them with the results by using the average, the biggest difference is the result of RCR when all the users are counted (i.e., the maximum value in the *x*-axis of **Fig. 5**). While using the average where all the ratings are equally considered to compute the reputation cannot distinguish normal users from RAs, using the weighted sum is likely to distinguish normal users from RAs, especially when using the features of FMV, FMD, and Activity.

Fig. 5-(a) to **Fig. 5-(d)** show that overall FMV, FMD, and PV perform well. Similar to the results obtained by using the average, the RCRs of the three features increase up to 70%. After that, the RCRs of FMV and FMD gradually deteriorate while that of PV abruptly deteriorates. In the case of FMV, even though more than 70% users are used in computing reputation (i.e., RAs are involved in reputation computation), RCR does not vary greatly with the ratio of used users. Overall, when online rating systems are attacked by push RAs, we can get the reputation close to true one by using the weighted sum with only a small number of users ranked by FMV.



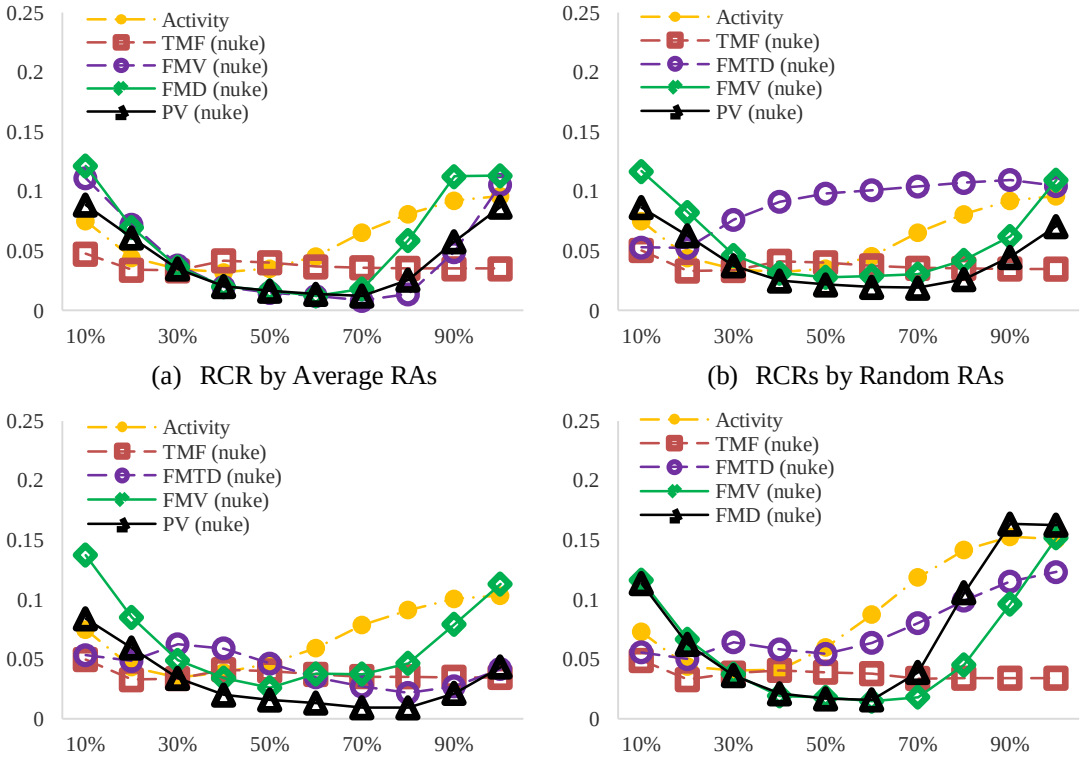
(a) RCR by Average RAs (b) RCRs by Random RAs

(c) RCR by Segment RAs (d) RCRs by Selected Popular RAs

Fig. 5. Reputation Change by push RAs (using the weighted sum)

The figures in **Fig. 6**, **Fig. 6-(a)** to **Fig. 6-(d)**, show the effectiveness of features when using the weighted sum under the attacks by different types of nuke RA. Using the weighted sum seems to distinguish normal users from RAs, especially when using the features of TMF.

Fig. 6-(a) to **Fig. 6-(d)** show that overall, the top 5 features work reasonably. Similar to the results by using the average, the RCRs of FMV and PV increase up to 70%. However, after the point, the RCRs of FMV and PV gradually deteriorate. In the case of TMV, it shows a very low RCR even though a small number of users are used in computing reputation. Also, even though more than 70% users are used in computing reputation, the RCR is still reasonable. Overall, when online rating systems are attacked by nuke RAs, we can get the reputation close to true one by using the weighted sum with only a small number of users ranked by TMV.



(a) RCR by Average RAs (b) RCRs by Random RAs

(c) RCR by Love/Hate RAs (d) RCRs by Reverse Selected Popular RAs

Fig. 6. Reputation change by nuke RAs (using the weighted sum)

Table 4 shows the changes of RCRs when using the weighted sum, under attacks by push RAs, according to the percent of selected users & the used feature and the ranks of features according to FT. We shade the cells whose values are smaller than *0.1627*, which is RCR by using the average when aggregating ratings of all users. Comparing them with RCRs in **Table 2**, the RCRs of Activity, TMF, FMV, FMD, and PV with 100% users show lower than 0.1627, the average value.

Table 4. Effectiveness of all the features by push RAs (using the weighted sum)

	10%	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**	***FT***	**RANK**
Activity	0.1038	0.0574	0.0491	0.0544	0.0693	0.0861	0.1057	0.1212	0.1320	0.1339	0.0249	3
Objectivity	0.6983	0.6110	0.4990	0.4047	0.3397	0.2939	0.2610	0.2381	0.2236	0.2102	0.1501	10
Consistency	0.6695	0.5711	0.4761	0.4047	0.3596	0.3284	0.3031	0.2860	0.2770	0.2739	0.1479	9
WC	0.5433	0.4667	0.3842	0.3184	0.2779	0.2483	0.2241	0.2105	0.2027	0.1991	0.1182	7
RDMA	0.8146	0.7362	0.5789	0.4637	0.3904	0.3453	0.3118	0.2941	0.2863	0.2844	0.1769	15
WDMA	0.8153	0.7331	0.5586	0.4208	0.3429	0.2903	0.2556	0.2333	0.2246	0.2236	0.1703	14

WDA	0.6803	0.6759	0.5709	0.4648	0.4001	0.3606	0.3432	0.3364	0.3349	0.3348	0.1627	12
DegSim (*k*=25)	0.8135	0.7518	0.6268	0.5154	0.4522	0.4118	0.3842	0.3625	0.3460	0.3321	0.1858	17
DegSim' (*d*=10, *k*=25)	0.8142	0.7653	0.6639	0.5655	0.5017	0.4580	0.4213	0.3858	0.3554	0.3364	0.1918	19
DegSim' (*d*=20, *k*=25)	0.7952	0.7414	0.6418	0.5501	0.4960	0.4519	0.4175	0.3900	0.3644	0.3465	0.1875	18
DegSim' (*d*=30, *k*=25)	0.7088	0.6627	0.5978	0.5534	0.5164	0.4884	0.4642	0.4479	0.4318	0.4222	0.1775	16
DegSim' (*d*=40, *k*=25)	0.6441	0.6132	0.5703	0.5215	0.4817	0.4554	0.4332	0.4184	0.4055	0.3978	0.1642	13
DegSim' (*d*=50, *k*=25)	0.5880	0.5618	0.5426	0.5002	0.4684	0.4509	0.4384	0.4305	0.4251	0.4221	0.1550	11
LengthVar	0.1982	0.2834	0.3529	0.3658	0.3325	0.2996	0.2781	0.2651	0.2603	0.2595	0.0793	6
TMF (push)	0.1077	0.1160	0.1148	0.1095	0.1123	0.1110	0.1153	0.1172	0.1153	0.1151	0.0328	5
FMTD (push)	0.5377	0.5122	0.4392	0.3659	0.3056	0.2689	0.2297	0.2070	0.1912	0.1792	0.1235	8
FMV (push)	0.1322	0.0743	0.0492	0.0379	0.0304	0.0243	0.0191	0.0210	0.0347	0.0523	0.0220	1
FMD (push)	0.1205	0.0868	0.0513	0.0398	0.0289	0.0202	0.0159	0.0305	0.0672	0.1027	0.0224	2
PV (push)	0.1218	0.0843	0.0754	0.0575	0.0472	0.0337	0.0242	0.0365	0.0911	0.1604	0.0253	4

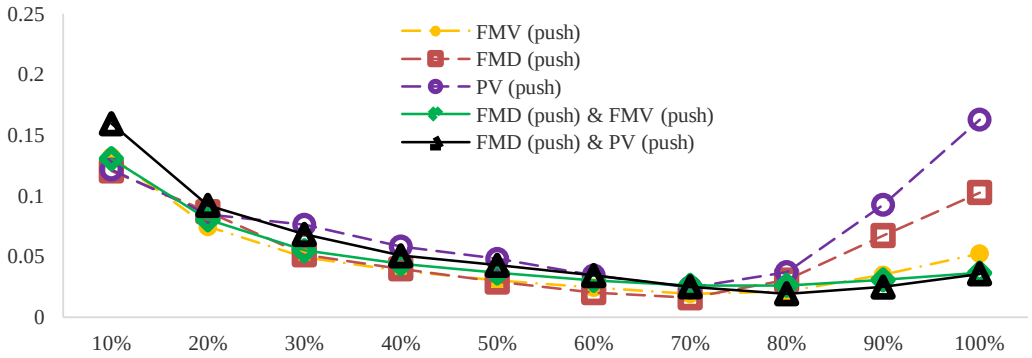
Table 5 shows the changes of RCRs when using the weighted sum, under attacks by nuke RAs, according to the percent of selected users & the used feature and the ranks of features according to FT. We shade the cells whose values are smaller than *0.1324*, which is RCR by using the average of ratings of all users. Comparing them with RCRs in **Table 3**, the RCRs of Activity, TMF, FMTD, FMV, FMD, and PV with 100% users are lower than 0.1324. In most cases, Activity, TMF, FMTD, FMV, FMD, and PV show reasonable RCRs while DegSim, $DegSim'$, RDMA, and WDMA show poor RCRs.

Table 5. Effectiveness of all the features by nuke RAs (using the weighted sum)

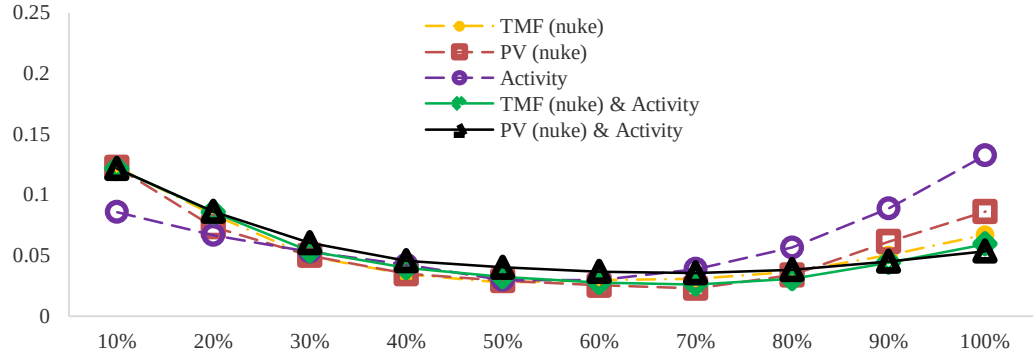
	10%	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**	***FT***	**RANK**
Activity	0.0745	0.0439	0.0362	0.0360	0.0442	0.0594	0.0821	0.0986	0.1094	0.1114	0.0184	3
Objectivity	0.5318	0.4558	0.3816	0.3088	0.2584	0.2261	0.2022	0.1802	0.1642	0.1590	0.1139	10
Consistency	0.5838	0.4196	0.3172	0.2560	0.2163	0.1954	0.1849	0.1777	0.1739	0.1738	0.1125	9
WC	0.4236	0.3136	0.2428	0.2033	0.1798	0.1621	0.1479	0.1437	0.1432	0.1431	0.0844	8

RDMA	0.7160	0.6118	0.4638	0.3601	0.2989	0.2630	0.2375	0.2266	0.2193	0.2173	0.1478	15
WDMA	0.7215	0.6074	0.4433	0.3320	0.2651	0.2226	0.2010	0.1890	0.1786	0.1771	0.1436	14
WDA	0.5925	0.5607	0.4440	0.3555	0.3047	0.2776	0.2643	0.2592	0.2578	0.2578	0.1342	12
DegSim (*k*=25)	0.7263	0.6327	0.5158	0.4366	0.3855	0.3501	0.3236	0.3032	0.2864	0.2744	0.1603	18
DegSim' (*d*=10, *k*=25)	0.7241	0.6383	0.5282	0.4531	0.4030	0.3618	0.3292	0.3027	0.2846	0.2667	0.1617	19
DegSim' (*d*=20, *k*=25)	0.6759	0.6170	0.5172	0.4392	0.3882	0.3557	0.3313	0.3085	0.2895	0.2744	0.1549	17
DegSim' (*d*=30, *k*=25)	0.6413	0.5761	0.5050	0.4556	0.4167	0.3916	0.3748	0.3604	0.3487	0.3395	0.1531	16
DegSim' (*d*=40, *k*=25)	0.5881	0.5390	0.4738	0.4243	0.3899	0.3657	0.3509	0.3395	0.3297	0.3221	0.1422	13
DegSim' (*d*=50, *k*=25)	0.4914	0.4775	0.4443	0.4097	0.3851	0.3721	0.3631	0.3566	0.3523	0.3488	0.1290	11
LengthVar	0.2003	0.2361	0.2993	0.3014	0.2641	0.2376	0.2204	0.2112	0.2069	0.2063	0.0687	7
TMF (nuke)	0.0491	0.0332	0.0348	0.0413	0.0399	0.0373	0.0352	0.0351	0.0347	0.0347	0.0119	1
FMTD (nuke)	0.0536	0.0501	0.0698	0.0795	0.0820	0.0850	0.0881	0.0907	0.0943	0.0960	0.0196	4
FMV (nuke)	0.1204	0.0765	0.0426	0.0261	0.0217	0.0234	0.0237	0.0369	0.0717	0.1200	0.0216	5
FMD (nuke)	0.1246	0.0720	0.0485	0.0343	0.0327	0.0285	0.0389	0.0836	0.1304	0.1284	0.0240	6
PV (nuke)	0.0896	0.0682	0.0434	0.0289	0.0249	0.0225	0.0217	0.0319	0.0549	0.0814	0.0175	2

We also examine whether the combination of robust features is more effective in getting true reputation when using the weighted sum. **Fig. 7-(a)** shows the top 5 features selected by comparing the effectiveness of FMD, FMV, and PV (the best three features under attacks by push RAs) and all the possible combinations of the three features (FMV&FMD, FMV&PV, FMD&PV, FMV&FMD&PV). **Fig. 7-(b)** shows the top 5 features selected by comparing the effectiveness of TMF, PV, and Activity (the best three features under attacks by nuke RAs) and the combinations of the three features (TMF&PV, TMF&Activity, PV&Activity, and TMF&PV&Activity). **Fig. 7** shows the effectiveness of combinations of features. Even though all the users are involved in computing reputation, the combination of robust features show very low RCRs. This indicates that the weighted sum method using robust features and the combination of robust features can calculate reputation reducing the influence of unfair ratings of RAs without the risk of omitting ratings of normal users.



(a) RCR using feature combinations by push RAs



(b) RCR using features combinations by nuke RAs

Fig. 7. Reputation Change using feature combinations (using the weighted sum)

5. Conclusions

For trustable aggregation of online ratings, we have identified TUs by finding robust features, which are directly opposite to the features of RAs and activity, objectivity, and consistency. The features are normalized into the values expressing the trustworthiness of users. We have employed user features for including as few RAs and many TUs as possible in computing true reputation. We have verified the effectiveness of each feature through extensive experiments. The experimental results show that some features are very robust to RAs. Regardless of being attacked by push RAs or nuke RAs, features of Activity, TMF, FMV, FMD, and PV are very robust. These robust features help to get true reputation by using only a small number of trustable users rather than whole users. In particular, if we use the average (i.e., to assign the same weight to each rating) in aggregating ratings of users, by using the top 70% users ranked by FMV, we could obtain reputation close to ground truth. When we use more users, RAs are likely to be involved in the process of the reputation computation. On the other hands, if we use the weighted sum method in aggregating ratings of users, by using the top 20% to 100% users ranked by FMV&PV and FMD&FMV, we could obtain reputation close to ground truth. These results show that the weighted sum method can distinguish TUs from RAs by assigning the different weights to different ratings.

References

[1] M. Brennan, S. Wrazien, and R. Greenstadt, "Using Machine Learning to Augment Collaborative Filtering of Community Discussions," in *Proc. of the 9th Int'l Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1569-1570, 2010. Article (CrossRef Link)

[2] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik, "Classification Features for Attack Detection in Collaborative Recommender Systems," in *Proc. of the 12th Int'l Conf. on Knowledge Discovery and Data Mining (KDD)*, pp. 542-547, 2006. Article (CrossRef Link)

[3] P. Chirita, W. Nejdl, and C. Zamfir, "Preventing Shilling Attacks in Online Recommender Systems," in *Proc. of the 7th Annual ACM Int'l Workshop on Web Information and Data Management (WIDM)*, pp. 67-74, 2005. Article (CrossRef Link)

[4] M. Eisend, "Source Credibility Dimensions in Marketing Communication-A Generalized Solution," *Journal of Empirical Generalisations in Marketing Science*, vol. 10, no. 2, 1-33, 2006. Article (CrossRef Link)

[5] S. Grazioli and S. L. Jarvenpaa, "Perils of Internet Fraud: An Empirical Investigation of Deception and Trust with Experienced Internet Consumers," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 30, no. 4, pp. 395-410, 2000. Article (CrossRef Link)

[6] I. Gunes, C. Kaleli, A. Bilge, and H. Polat, "Shilling Attacks against Recommender Systems: A Comprehensive Survey," *Artificial Intelligence Review*, pp. 1-33, 2012. Article (CrossRef Link)

[7] G. Häuubl and V. Trifts, "Consumer Decision Making in Online Shopping Environments: The Effects of Interactive Decision Aids," *Marketing Science*, vol. 19, no. 1, pp. 4-21, 2000. Article (CrossRef Link)

[8] J. Howe, "The Rise of Crowdsourcing," *Wired*, 2006. Article (CrossRef Link)

[9] N. Hurley, Z. Cheng, and M. Zhang, "Statistical Attack Detection," in *Proc. of the 3rd ACM Conf. on Recommender Systems (RecSys)*, pp. 149-156, 2009. Article (CrossRef Link)

[10] S. Lam and J. Riedl, "Shilling Recommender Systems for Fun and Profit," in *Proc. of the 13th Int'l Conf. on World Wide Web (WWW)*, pp. 393-402, 2004. Article (CrossRef Link)

[11] C. Leadbeater, "WE-THINK: Mass Innovation, not Mass Production," *Profile Books*, 2008. Article (CrossRef Link)

[12] J.-S. Lee and D. Zhu, "Shilling Attack Detection–A New Approach for a Trustworthy Recommender System," *INFORMS Journal on Computing*, vol. 24, no. 1, pp. 117-131, 2012. Article (CrossRef Link)

[13] P. Levy, "L'Intelligence Collective: Pour Une Anthropologie Du Cyberspace," *La Découverte*, 1997. Article (CrossRef Link)

[14] M. Limayem, M. Khalifa, and A. Frini, "What Makes Consumers Buy from Internet? A Longitudinal Study of Online Shopping," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 30, no. 4, pp.421-432, 2000. Article (CrossRef Link)

[15] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, "Towards Trustworthy Recommender Systems: An Analysis of Attack Models and Algorithm Robustness," *ACM Transactions on Internet Technology*, vol. 7, no. 2, pp. 1-40, 2007. Article (CrossRef Link)

[16] Nielsen, "Trends in Online Shopping," *a global Nielsen consumer report*, Feb. 2008. Article (CrossRef Link)

[17] H.-K. Oh, S.-W. Kim, S. Park, and M. Zhou, "Can You Trust Online Ratings? A Mutual Reinforcement Model for Trustworthy Online Rating Systems," *IEEE Transactions on Systems, Man and Cybernetics: Systems*, vol. 45, no. 12, pp. 1564-1576, 2015. Article (CrossRef Link)

[18] H.-K. Oh, S.-W. Kim, S. Park, and M. Zhou, "Trustable Aggregation of Online Ratings," in *Proc. of the 22nd ACM Int'l Conf. on Information and Knowledge Management (CIKM)*, pp. 1233-1236, 2013. Article (CrossRef Link)

[19] S. Y. Rieh and D. Danielson, "Credibility: A Multidisciplinary Framework," *Annual Review of Information Science and Technology*, vol. 41, no. 1, pp. 307-364, 2007. Article (CrossRef Link)

[20] J. Surowiecki, "The Wisdom of Crowds: Why the Many are Smarter than the Few and how Collective Wisdom Shapes Business, Economies, Societies, and Nations," *NY: Doubleday*, 2004. Article (CrossRef Link)
[21] A. Whitby, A. Jøsang and J. Indulska, "Filtering Out Unfair Ratings in Bayesian Reputation Systems," *The Icfain Journal of Management Research*, vol. 4, no. 2, pp. 48-64, 2005. Article (CrossRef Link)
[22] Z. Wu, J. Cao, B. Mao, and Y. Wang, "Semi-Sad: Applying Semi-Supervised Learning to Shilling Attack Detection," in *Proc. of the 5th ACM Conf. on Recommender Systems (RecSys)*, pp. 289-292, 2011. Article (CrossRef Link)
[23] Z. Wu, J. Wu, J. Cao, and D. Tao, "HySAD: A Semi-Supervised Hybrid Shilling Attack Detector for Trustworthy Product Recommendation," in *Proc. of the 18th Int'l Conf. on Knowledge Discovery and Data Mining (KDD)*, pp. 985-993, 2012. Article (CrossRef Link)
[24] S. Zhang, A. Chakrabarti, J. Ford, and F. Makedon, "Attack Detection in Time Series for Recommender Systems," in *Proc. of the 12th Int'l Conf. on Knowledge Discovery and Data Mining (KDD)*, pp. 809-814, 2006. Article (CrossRef Link)
[25] A. Ng, *Machine Learning and AI via Brain simulations*, Stanford University, 2015. Article (CrossRef Link)
[26] "Feature Engineering: How to transform variables and create new ones?" *Analytics Vidhya*. 2015. Article (CrossRef Link)
[27] J. Brownlee, "Discover Feature Engineering, How to Engineer Features and How to Get Good at It," *Machine Learning Mastery*, 2015. Article (CrossRef Link)
[28] H. Ding, L. Cao, Y. Ren, K.-K. R. Choo, and B. Shi, "Reputation-Based Investment Helps to Optimize Group Behaviors in Spatial Lattice Networks," *PLoS ONE*, 2016. Article (CrossRef Link)
[29] H. Ding, Y. Zhang, Y. Ren, B. Shi, and K.-K. R. Choo, "Heterogeneous Investment in Spatial Public Goods Game with Mixed Strategy," *Soft Computing*, 2016. Article (CrossRef Link)
[30] H. Ding, J. Huang, Y. Chen, Y. Ren, "Don't Speak to Strangers: The Suspicious Strategy Can Help to Improve Cooperation in Spatial Donation Game," in *Proc. of The 13rd IEEE Int'l Conf. on Dependable, Autonomic and Secure Computing (DASC2015)*, pp. 1954–1959, 2015. Article (CrossRef Link)
[31] T. Ma, J. Zhou, M. Tang, Y. Tian, A. Al-Dhelaan, M. Al-Rodhaan, and S. Lee, "Social Network and Tag Sources based Augmenting Collaborative Recommender System," *IEICE transactions on Information and Systems*, vol. E98-D, no.4, pp. 902-910, Apr. 2015. Article (CrossRef Link)
[32] B. Gu, X. Sun and V. S. Sheng, "Structural Minimax Probability Machine," *IEEE Transactions on Neural Networks and Learning Systems*, 2016 Article (CrossRef Link)
[33] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling Personalized Search over Encrypted Outsourced Data with Efficiency Improvement," *IEEE Transactions on Parallel and Distributed Systems*, 2015 Article (CrossRef Link)

Hyun-Kyo oh received the B.S., M.S. and Ph.D. degree in Electronics and Computer Engineering from Hanyang University, Seoul, Korea at 2008, 2010 and 2016. Currently, he is a senior engineer at S/W development team in Memory Division of Samsung Electronics. He visited the Department of Computer Science of Carnegie Mellon University as a Visiting Scholar in 2013. He worked with Microsoft Research Asia, China as a Research Intern from 2014 to 2015. His research interests include machine learning, deep learning, NAND flash memory, and computational trust.

Sang-Wook Kim received a B.S. degree in Computer Engineering from Seoul National University at 1989, and earned M.S. and Ph.D. degrees in Computer Science from KAIST at 1991 and 1994. From 1995 to 2003, he served as an Associate Professor of Division of Computer, Information, and Communications Engineering at Kangwon National University. In 2003, he joined Hanyang University, where he currently is a Professor at Department of Computer Science and Engineering. His research interests include databases, data mining, social network analysis, and recommendation. From 2009 to 2010, Dr. Kim visited Computer Science Department at CMU as a Visiting Professor. From 1999 to 2000, he also worked with IBM Watson Research Center as a Post-Doc. He is now an Associate Editor of Information Sciences and is a member of ACM and IEEE.