Improving the True-Reputation Algorithm by Age Parameters

July 2018

https://github.com/itaygal/RS_TrueReputation

Itay Gal
Software and Information
Systems Engineering
Ben Gurion University
Beer Sheva, Israel
itag@post.bgu.ac.il

Yefim Faiterberg
Software and Information
Systems Engineering
Ben Gurion University
Beer Sheva, Israel
faiterbe@post.bgu.ac.il

ABSTRACT

Collaborative filtering techniques have been successfully employed in recommender systems, but such systems have been shown to be vulnerable to the reputation injections attacks. Recently published True-Reputation algorithm is a novel algorithm in the field of robustness recommender systems that has proven to be superior to existing state-of-the-art algorithms in the field. In this paper we introduce an improved version of True-Reputation algorithm that is taking into account age parameters of users and items. These are indicative parameters based on well-founded theories that new items are the most attractive targets for reputation attacks and that reputation attackers will use new fictitious profiles for executing the attacks. Through various and extensive experiments, we manifest models of reputation attacks where our improved algorithm outperforms the original True-Reputation, and in some models the improved algorithm even achieved a totally superiority.

KEYWORDS

Recommender Systems, Collaborative Filtering, Robust Collaborative Filtering, Trustable Rating, Reputation Attackers, True Reputation, Age Parameter

I. INTRODUCTION

Recommender Systems (RSs) are software tools and techniques providing predictions of user's interest in an item. One of the most common approach for recommender systems is a Collaborative Filtering (CF) [14]. Collaborative Filtering is a method of predicting the interests of a user by collecting and analyzing its preferences and the preferences of its similar users (nearest neighbors). The underlying assumption of collaborative filtering is that if a user U1 has the same interest as a user U2 (for example both of them liked the same movie), then U1 is more likely to have U2's interest in an unknown yet item for U1. For example - if U2 liked a new movie that he recently watched, then most likely U1 will like this movie too, and it is a suitable item to recommend for him [16].

The most common way for consumers to express their satisfaction with the item is through online ratings. The overall users' satisfaction on an item can be quantified as the wisdom of crowds – the aggregated score of all ratings given to the item (Item's Reputation). The reputation of an item plays an important role for recommender systems to recommend the item for the user and also significantly psychological influence on the user to consume the item [7][9]. Obviously, the trustworthiness of a reputation can be achieved automatically only when the users honestly rates the item, otherwise - the reputation could be cunningly manipulated with the injection of unfair ratings (False Reputation). Moreover, the magnitude of false reputation is influenced by the ratio between the trustworthy users and the reputation attackers (RAs). As a result, in a high-risk situation are located new or unknown products with small number of reliable ratings. For example, in the case of prerelease movie, the production company may hire a bunch of people to provide high ratings for the movie and consequently increase the movie's reputation.

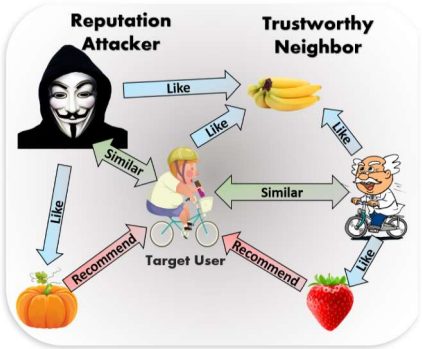

Figure 1. Attack on CF Recommendations Model

Figure 1 illustrates by toy-example the vulnerability of collaborative filtering methods to reputation attacks. On the right part of the figure – recommender system based on collaborative filtering methods founds similarity between the target-user and the trustworthy-neighbor (both of them love bananas). Afterwards, recommender system detects that in addition to bananas, the trustworthy-neighbor likes strawberry, and consequently recommends the strawberry to the target-user. This is a classical win-win situation - the target-user buys a strawberry after a trustworthy recommendation of a recommender system and happily discovers that strawberry and banana are a really good combination in preparing a juice. But in a meanwhile, a company that produce pumpkins vegetables detects a decline in their sales and hires a reputation attack firm to promote the sales. The reputation attack firm creates fictive user profile and attempts to achieve a high similarity to the target user in a recommender system (for example by liking the bananas). This process will cunningly cause the recommender system to recommend the pumpkin to the target-user and create a lose-lose situation for the relationship between the target-user and the recommender system. According to the biased recommendation, the target user gets a product that he does not really like and predictably became embittered and loses the trust in a recommender system. The only winner of this situation is the reputation attackers. Of course, it is a simplified and abstract toy example – in a real-life scenario, recommender system will not define similarity between two users, only on the basis of one common item.

Also the reputation attackers in most cases will need to create a bit more than one fictitious profile for making a significant biased effect.

In this paper, first we describe the scenarios in which a false reputation could occur and we depict the potential damage it could cause to the recommender system. Second, we review the existing algorithms that dealing with the detection of false reputation with emphasis on True-Reputation algorithm that is in the leading position in this field. Third, we propose an improvement for the True-Reputation algorithm by utilizing the user and item age parameters. Fourth, we verified the superiority of our improved algorithm by comparing it with the original True-Reputation algorithm through variety extensive experiments.

This paper is organized as follows. Section II (Related Work) introduces related work in the field of robust recommender systems and the attempts of age parameters improvement made on various statistical algorithms. Section III (Method) describes our proposal for improvement of True-Reputation algorithm by interpolating the age parameters in the confidence function of a rating. Section IV (Evaluation) verifies the superiority of the improved algorithm by comparing it with the original True-Reputation algorithm through variety extensive experiments. Section V (Results) manifests the results of the evaluation experiments through the various scenarios of reputation attacks. Section VI (Discussion) discusses about the experiments results and their implications. Section VII (Conclusion) concludes this paper with directions for future research.

II. RELATED WORK

The theoretical basis for the vulnerabilities in collaborative recommender systems has been well established. Much of these works relates to an earlier research on the impact of biased noise [13]. Since the first introduction of rating attacks in 2004 [8], many different models of rating attacks have been proposed in the literature [2][4][11]. There are classical models like Random and Average attack models that were introduced originally in 2004 [8], and there are more innovative models like Bandwagon and Segment attack models. This is only a partial list of attack models that is growing up frequently. It seems like a "cat and mouse game" – when attackers develop a new kind of rating attack model, there are significant efforts of recommender systems researchers to characterize the signatures of the attacks and exclude the fictive ratings from the reliable ones. The attackers in turn do not give up and they restart the game with a new and cunning approach to attack the recommender systems from another angle. It looks like an endless game that will accompany the researchers in this field for a long time.

Profile of rating attackers can be categorized based on three parameters: (a) the knowledge required by the attacker to mount the attack; (b) the intent of a particular attack; (c) the size of the attack [11]. Knowledge required by the attackers can be defined as high or low. Low level of knowledge is information that does not require much effort to achieve - like recognizing popular items that might be obtained by deriving public information sources. On the other hand, high level of knowledge is information that require significant effort to achieve or even insider information – like the average rating and standard deviation distribution of the items in the system. There are two main intents of the rating attacks – push and nuke. Push attack means that the attackers give an unfairly high rating to promote the item, and on the other hand, nuke attack means that the attackers give an unfairly low rating to demote the item. The size of the attack is measured by the amount of attack profiles and by the amount of the ratings from these profiles. From the perspective of the attacker, the best attack against a recommender system is of course, one that yields the biggest impact for the least amount of effort. We used five push and five nuke common attack models to evaluate the various algorithms in this study. Section IV (Evaluation) depicts more details and formal definitions of our used attack models.

Numerous studies have been conducted to improve the trustworthiness of recommender systems. The main idea is to detect the abusers who participated in the rating attacks and exclude their ratings when calculating item's reputation. Studies on the field of robustness of recommender systems began from strategies based on multiagent systems. Main approach was the principle of majority rule – the system considers as fair rating only the majority ratings (more than half the ratings), and excludes the minority ratings when calculating the reputation [3][15][17]. A prominent drawback of this approach is that it puts in high risk new or unpopular items with a relatively small amount of ratings. For them every serious attack model with significant amount of rating attacks might turn the table upside down and change the item's reputation radically.

Major research nowadays in a rating attacks detection concentrates on three main stages: (a) classifying reputation attacks according to different attack models; (b) extracting attributes that represent the characteristics of each model; (c) developing algorithms based on classification or clustering that used quantified attributes to detect the rating attacks [11][10]. A cutting-edge research published in 2015 [18] represents a new True-Reputation algorithm, that demonstrates through extensive experiments its superiority over the existing classification algorithms. Its superiority can be explained by the fact that classification algorithms are not extremely accurate on their classification. They may face situations where rating attackers cannot be detected or where trustworthy users are mistakenly considered as attackers. As a result, it can lead to situation when a reputation is calculated without the ratings of trustworthy users or worse when the calculation includes fictitious ratings. Additionally, a significant amount of time is required to collect training data and extract attributes related to the rating attackers.

True-Reputation algorithm first introduced in [18], uses all ratings to calculate the reputations of items, without the risk of losing the ratings of trustworthy users. The main idea of the algorithm is to reduce the influence of unfair ratings by assigning a confidence score to them. Furthermore, True-Reputation is a graph-based algorithm and does not require time for machine learning training process. To compute the confidence score, True-Reputation uses three values: (a) user activity; (b) user objectivity; (c) rating consensus score.

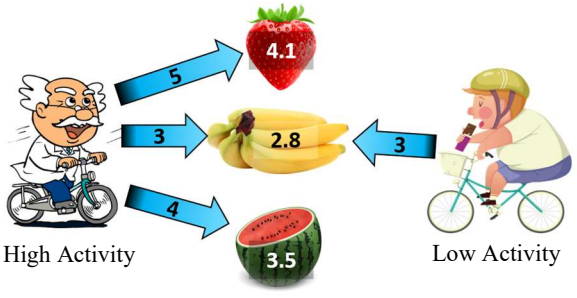

Figure 2. High vs Low Activity Users

Figure 2 illustrates by toy-example the difference between high and low activity users. The user with more ratings is considered a more active user. User activity attribute based on assumption that high activity users are likely to be more trustworthy. User activity formal defined as follows [11]:

$$a_u = \Psi\left(\left|\mathbf{R}^u\right|, \alpha, \mu\right) \quad (1)$$

where

$$\Psi\left(\left|\mathbf{R}^u\right|, \alpha, \mu\right) = \frac{1}{1 + e^{-\alpha(|\mathbf{R}^u| - \mu)}}. \quad (2)$$

The user's activity, denoted by a_u, is quantified by the frequency of his ratings, where R_u is a set of ratings by user u. The Ψ function is a sigmoid function for normalization of $|R_u|$ to keep the returned value in the range of [0, 1]. Parameters α∈Z and μ∈Z determine the slope and adjust the midpoint of the curve of normalization function and should be selected in a way that evenly distribute $|R_u|$ in the range of [0, 1] (Generally, μ is the average value).

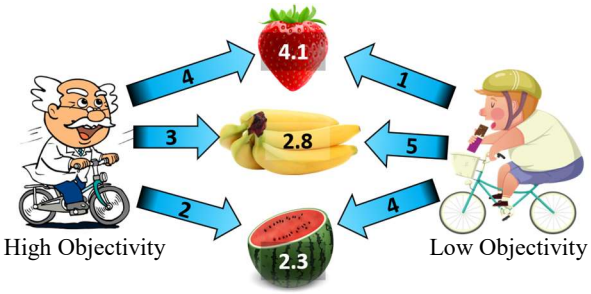

Figure 3. High vs Low Objectivity Users

Figure 3 illustrates by toy-example the difference between high and low objectivity. From the illustration it can be noticed that the ratings of the left user in the figure, are more closely to the reputation of the items, and therefore he is considered to be a more objective. The objectivity of rating is computed based on its deviation from the aggregated score of the item's reputation. The objectivity of a rating denoted by o_r is higher when the rating is closer to the item's reputation. o_r is calculated based on the reputation, denoted by $\overline{r_m}$ and the standard deviation, denoted by s_m as follows [11]:

$$o_r = \left|\frac{r - \bar{r}_m}{s_m}\right|. \quad (3)$$

Equation (3) indicates that rating r is a more objective when o_r is closer to 0. User objectivity, denoted by o_u, is calculated as the average of the objectivities of the ratings by that user. If this value is closer to 0, the user is more objective. The formal definition of o_u is as follows [11]:

$$o_u = \frac{1}{\left|\mathbf{R}^u\right|} \sum_{r \in \mathbf{R}^u} o_r. \quad (4)$$

In order to use o_u in calculating the confidence of a rating, we need to normalize it. The computational model defines the user whose normalized objectivity is "1" as the most objective user. The transformation function ψ for normalizing o_u is defined as follows [11]:

$$o_u^* = \Psi\left(o_u, \alpha', \mu'\right). \quad (5)$$

o_u^* is normalized by the Ψ function with the α' and μ' parameters in (5). The Ψ function in (5) is the sigmoid function for normalization of o_u to keep the value in the range of [0, 1] and to reduce the influence of a user who has a very large o_u value.

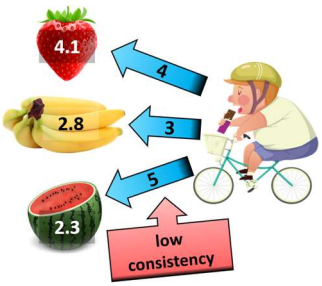

Figure 4. Rating Consistency

Figure 4 illustrates by toy-example the intuition of rating consistency. The user ratings of strawberry and of banana are closely to their reputation and at the other hand, the rating of watermelon is extremely high relatively to its reputation. This causes the rating of watermelon to be a low consistent rating of the user. It is a very useful attribute especially against a sophisticated attack's models, when an attacker trying to simulate a reliable user and rates most of the items with a rating closely to their reputation, and only the target item gets unfairly high or low rating from him.

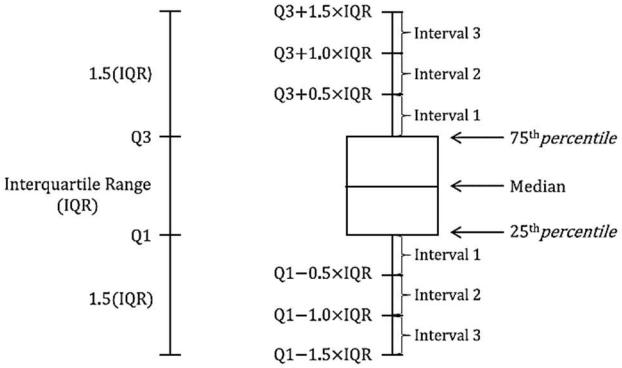

Figure 5. Box-plot for user's consistency adopted from [11].

Consensus score c_r represents the abnormality of a rating with respect to user consistency in ratings item. An abnormal rating r is given a low consensus score, while a rating consistent with the user's normal behavior is given a high consensus score. Figure 5 illustrate box-plot analysis is used by True-Reputation to analyze the distribution of o_r for each user and to detect abnormal ratings. Values within the IQR are considered normal behavior with a consensus score 1, values outside the IQR are more highly penalized as o_r moves farther from the median. The outliers considered abnormal behavior, with a consensus score 0 [11]:

$$c_r = \begin{cases} 0, & \text{if } o_r > Q3 + 1.5\text{IQR} \\ & \text{or } o_r < Q1 - 1.5\text{IQR} \\ 0.5, & \text{if } o_r \le Q3 + 1.5\text{IQR} \quad \text{and } o_r > Q3 + 1.0\text{IQR} \\ & \text{or } o_r \ge Q1 - 1.5\text{IQR} \quad \text{and } o_r < Q1 - 1.0\text{IQR} \\ 0.7, & \text{if } o_r \le Q3 + 1.0\text{IQR} \quad \text{and } o_r > Q3 + 0.5\text{IQR} \\ & \text{or } o_r \ge Q1 - 1.0\text{IQR} \quad \text{and } o_r < Q1 - 0.5\text{IQR} \\ 0.9, & \text{if } o_r \le Q3 + 0.5\text{IQR} \quad \text{and } o_r > Q3 \\ & \text{or } o_r \ge Q1 - 0.5\text{IQR} \quad \text{and } o_r < Q1 \\ 1, & \text{otherwise.} \end{cases} \quad (6)$$

True-Reputation is an iterative algorithm. The initial reputation of an item is set to be the arithmetic mean of all ratings provided. Thus, all ratings have the same initial confidence scores. At each iteration, True-Reputation recomputes the confidence of each rating and adjusts the reputation of each item based on the recalculated confidence scores of all ratings. The algorithm stops when the computation converges to a stable state. The confidence of a rating t_r is computed as follows [11]:

$$t_r = a_u \times o_u^* \times c_r, \quad r \in \mathbf{R}^u. \tag{7}$$

The reputation of each item is then adjusted based on the confidence of the ratings on the same item as follows [11]:

$$\bar{r}_m = \frac{\sum_{r \in \mathbf{R}_m} (r \times t_r)}{\sum_{r \in \mathbf{R}_m} t_r}. \tag{8}$$

The process of computing t_r and adjusting $\overline{r_m}$ continues until the values reach a stable state. The stability of True-Reputation is measured by the marginal change in reputations between the iterations. Let the vector $\vec{r}$ represents the reputation of all items. The difference between the old vector $\overrightarrow{r^*}$ and the new vector $\vec{r}$ is measured as one minus the cosine similarity between the vectors, as follows [11]:

$$d(\vec{r}^*, \vec{r}) = 1 - \frac{\vec{r}^* \cdot \vec{r}}{\| \vec{r}^* \| \| \vec{r} \|}. \tag{9}$$

If $d(\overrightarrow{r^*}, \vec{r})$ is less than some preset value δ, True-Reputation stops.

True-Reputation is an efficient algorithm that has been already demonstrated its superiority over leading algorithms of the field. The main drawback we identified in True-Reputation is its lack of reference to age parameters of users and items. We compute the age parameter of the user using the date of his first rating, and the age parameter of the item using its year of production. We believe there are substantial technical reasons for attackers to use new fictitious profiles in their attacks on the recommender system, and there are considerable economic reasons to prefer to attack new items than older (our assumptions are fully described in the next section). Based on our assumptions we believe that it is inaccurate to treat old users the same as new users, and in the same way it is not accurate to treat the old items the same as new items. In this paper we propose an improvement to the True-Reputation algorithm, that takes into computation the age parameters of the users and the items, and we demonstrate its superiority in various experiments over the True-Reputation algorithm. Note that we have already found positive precedents with improving recommender systems statistical algorithms using age parameters [1].

III. METHOD

True-Reputation algorithm is based on three key factors – user activity, user objectivity and rating consistency. We propose two additional factors to improve the algorithm – user age and item age.

User Age. We believe that there are substantial technical reasons for the reputation attackers to use new fictitious profiles in their attacks on the recommender system. First, when a reputation attacker's firm receives an attack task, it is not certainly that it already has profiles in the requested recommender system or that it has profiles that suitable to the requested task. Second, it takes too much effort to properly maintain fictitious profiles over time. Without properly maintenance, at any given moment the recommender system will be able to identify the fictitious profile and remove it from the system. There are many possibilities to suspect the fictitious profile, for example – suspicious profile activity, a long time without activity, suspicious traffic signatures or failure in verification of personal information. Third, in light of the fact that the interests of users are constantly changing, there are many collaborative recommender systems [19][5][6] that use various types of decay functions to normalize the ratings and to adapt the recommendations to the users' current interests. In this situation, there is a significant interest of reputation attackers to use new fictitious profiles for gaining a maximal effect for their attacks.

In our proposal, a user with higher age in the system should be considered a more senior user and as a result, get a higher user age score. We denote user age as t_u and it is quantified as the user registration date or the first user rating date (depending on what is giving by the system). In our experiments we use Unix timestamp data and to give t_u a more meaningful scale, we change Unix timestamp from seconds to months (compute number of months from 01/01/1970 UTC). This scale gave us best results in the experiments:

t_u=registration Unix timestamp/(60*60*24*30) (10)
60 seconds, 60 minutes, 24 hours, 30 days

We normalize t_u by applying the transformation sigmoid function. This not only keeps the value in the range of [0, 1] but also reduces the influence of extreme user age values. The user whose t_u^* is closely to 1 is the most senior user, he is at a low probability to be an attacker, so we do not want to significantly modify its reputation. The user whose t_u^* is closely to 0 is the most junior user, and we need to treat his ratings with greater suspicion. In order to distribute t_u^* evenly in the range of [0, 1], the values of α and μ should be determined appropriately. The transformation function should be closer to 1 when t_u is small (user is more senior) so α should be a negative number. Based on our experiments best results were achieved with using α = -0. 2 and μ set to be the average user age:

$$t_u^* = \Psi(t_u, \alpha, \mu) = \frac{1}{1+e^{-\alpha(t_u-\mu)}} \tag{11}$$

Item age. Our assumptions are that reputation attack firms do not work for free and must be compensated for their work. This means that their target items should be attractive enough for production companies to pay for their promotion. It is hard for us to find a logical reason for the production companies, that they would want to pay for promoting products created in 1940, for example. This is especially true if we concentrate on entertainment products such as movies, books or songs. In addition to the obvious intuition, according to [12] around the world more than six in ten respondents say they like to get offer of new products and more than half say that they purchased a new product during their last shopping. All these facts reinforce our assumptions that real reputation attackers will aim to attack recently created items, and we need to treat these items in a special way.

In our proposal, an item with higher age in the system should be considered a more senior item and as a result, get a higher item age score (less relevant for the reputation attack). We denote item age as t_i and it is quantified as the item release date or the first rating date for that item.

Unlike user age, here we do not use Unix timestamp value – in our experiments the item release year gave the best results:

$$t_i = \text{item release year} \quad (12)$$

We normalize t_i by applying the transformation sigmoid function. This not only keeps the value in the range of [0, 1] but also reduces the influence of extreme item age values. The item whose t_i^* is closely to 1 is the most senior item and is less relevant for reputation attack. The item whose t_i^* is closely to 0 is the most junior item and is very relevant for reputation attack, which makes us look at him suspiciously. In order to distribute t_i^* evenly in the range of [0, 1], the values of α and μ should be determined appropriately. The transformation function should be closer to 1 when t_i is small (small release year means more seniority) so α should be a negative number. Based on our experiments best results were achieved with using α = -0. 2 and μ set to be the average item age:

$$t_i^* = \Psi(t_i, \alpha, \mu) = \frac{1}{1+e^{-\alpha(t_i-\mu)}} \quad (13)$$

Same as the True-Reputation algorithm we use an iterative approach to compute the confidence score of a rating. First, we compute the a_u and t_u^* for all users and t_i^* for all items. Since a user activity level, user age and item age do not change, they are computed only once. The initial reputation of an item is set to be the arithmetic mean of all ratings provided; thus, all ratings have the same initial confidence score. At each iteration, we recompute the confidence of each rating and adjust the reputation of each item based on the recalculated confidence scores of all ratings.

The algorithm stops when the computation converges to a stable state. Next, we compute o_r for all ratings of all items, which is then used to compute both o_u and c_r. Afterward, the o_u is normalized to o_u^* using the transformation sigmoid function. The consensus score c_r of a rating r is also computed. The confidence of a rating, denoted as t_r is computed as follows:

$$t_r = a_u \times o_u^* \times c_r \times t_u^* \quad (14)$$

After the iterative algorithm reaches convergence, we balance the processed reputation $\bar{r}_m$ by item age normalized parameter t_i^*. The closer the t_i^* is to 1, the more likely it is that the item is not attractive for an attack (old item) and we want in this case to reduce the influence of True-Reputation processing on the item reputation, and try to keep it as close as possible to the original item reputation, denoted as $m_{avg\ item\ rating}$:

$$\bar{r}_m = \frac{\sum_{r \in \mathbf{R}_m} (r \times t_r)}{\sum_{r \in \mathbf{R}_m} t_r} \times (1 - t_i^*) + t_i^* \times m_{avg\ item\ rating}$$

Our improved algorithm is described in Algorithm 1 below. Black lines are original True-Reputation pseudo code, and the blue lines are our proposed improvements:

Algorithm 1: True-Reputation Improved

Input: set of users ***U***, set of items ***M***, set or Ratings ***R***
Output: reputations of all items

for u ∈ ***U*** do:
  Compute a_u
  Compute t_u
end for
for m ∈ ***M*** **do:**
  Compute t_i
end for
repeat:
  for m ∈ ***M*** **do:**
    for r ∈ R_m **do:**
      Compute o_r
    end for
  end for
  for u ∈ ***U*** **do**:
    Compute o_u
    Compute o_u^* from o_u
  end for
  for u ∈ ***U*** **do**:
    for r ∈ R_u **do:**
      Compute c_r
      Compute t_r
    end for
  end for

  for m ∈ ***M*** **do:**

$$\bar{r}_m = \frac{\sum_{r \in \mathbf{R}_m} (r \times t_r)}{\sum_{r \in \mathbf{R}_m} t_r}$$

  end for
  $\vec{r}^* = \vec{r}$
  $\vec{r} = [\bar{r}_1, \ldots, \bar{r}_l]$
until cosine similarity of $\vec{r}$ and $\vec{r}^*$ is less than δ

for m ∈ ***M*** **do:**

$$m_{avg\ item\ rating} = \frac{\sum r \in R_m}{|R_m|}$$

end for
for m ∈ ***M*** **do:**

$$\bar{r}_m = \frac{\sum_{r \in \mathbf{R}_m} (r \times t_r)}{\sum_{r \in \mathbf{R}_m} t_r} \times (1 - t_i^*) + t_i^* \times m_{avg\ item\ rating}$$

end for

Figure 5. Flow chart of our method

Our method is divided into four sections. First section is data collection – top priority task is collecting right, tidy and full data of users, items and relations between them (ratings). In order for the method to run properly, there is a need to collect meta data as well, such as item release year and timestamp of ratings executions. Second section is data processing – in this section we compute attributes that need to be computed only ones, such as user and item age. Third section is running algorithm – in this section we execute the iterative algorithm of improved True-Reputation until the values converge into a stable state. Fourth and last section is modified ratings – in this section we actually get processed rating, that should be much more trustworthy and adequate without the malicious influence of reputation attackers.

IV. EVALUATION

In our experiments, we used the MovieLens 100k dataset, which consists of 100 000 ratings using a scale from 1 (bad) to 5 (excellent) for 1682 movies submitted by 943 users. Each user rated at least 20 movies. The

dataset includes meta data about movies, from which we extracted the movie release year (used to calculate item age). It also contains a Unix timestamp for each rating (used for user age by taking the first user rating).

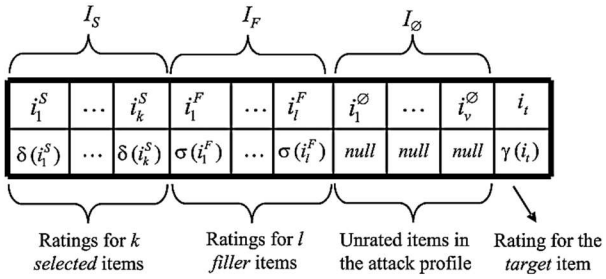

Figure 6. General form of the rating attack profile adopted from [11]

We generated ten various types of reputation attacks based on five different models (for every model we generated a push and a nuke approach). We use a formal framework to describe the rating attack models as defined in a classical study in the field of trustworthy recommender systems in [11]. The generic form of a rating attack profile is illustrated in Figure 2. Let I be a set of items in the dataset. The rating attack profile consists of four item sets: a set of target items (I_T) which an attacker aims to promote or demote; a set of selected items (I_S) which are selected differently depending on the attack models; a set of filler items (I_F) which are chosen randomly, and a set of unrated items $(I_\emptyset = I - (I_T \cup I_S \cup I_F))$.

(1) **Target Only** - the simplest strategy to cause false reputation. The attacker generates unfair ratings to target movies only (maximum ratings on push attack and minimum on nuke attack). The selected and the filler sets on this model are empty.
(2) **Love / Hate** – more intensive attack strategy. Besides unfair ratings to target movies, the attacker chooses random movies for filler set and generates for them the opposite rating from the target movies (minimal ratings on push attack and maximum ratings to nuke attack). Selected set on this model is empty.
(3) **Random** – on this strategy the attacker tries to camouflage himself as an ordinary user. Besides unfair ratings to target movies, the attacker chooses random movies for filler set and generates for them the average rating of all movies in the dataset. Selected set on this model is empty and push or nuke attack different only for target movies.
(4) **Average** – more sophisticated strategy to camouflage an attack. Besides unfair ratings to target movies, the attacker chooses random movies for filler set and generates for them their original average rating. This is high-knowledge model, because it needs an internal information. Selected set on this model is empty and push or nuke attack different only for target movies.
(5) **Popular** – the only model of our choices that uses the selected set. On push attack, the attacker chooses popular movies and generates for them maximum ratings as same as for target movies. The nuke attack is the opposite, the attacker chooses unpopular movies and generates for them minimum ratings as same as for target movies. The filler set on this model is the same as Random model – choose random movies and generate rating for them as the average rating of all movies in the dataset.

The release year of movies in MovieLens dataset distributes between 1922 to 1998. In our experiments, new movies with small amount of ratings (release year between 1997 and 1998 and amount of ratings between 60 and 140) are considered as movies in the dangerous situation and are selected as target movies (the definition of dangerous situations is adopted from experiments in [18]). In our experiments we varied three factors: (1) the percentage of attacked ratings out of the total number of ratings per movie. We varied the percentage from 5% to 30%; (2) the amount of ratings per attacker profile. We varied the amount from 2 to 32 on target only models, and from 50 to 100 in other models; (3) the attacking model. We generated ten attacker models on all spectrum of variations of the percentage of attacked ratings and amount of ratings per attacker profile. On target-only model we attacked 32 target movies, on other models we attacked 10 movies (we tried to stick as closely as possible to the experiments made in [18]). In light of the fact that we wanted to simulate a recent attack, we gave each rating a random date of the last 30 days from the last date in the dataset (24 April 1998).

We generated all the experiments on three algorithms: (1) arithmetic mean (red line in the graphs) – it is our baseline algorithm; (2) True-Reputation algorithm (blue line in the graphs); (3) Our improved True-Reputation algorithm (orange line in the graphs). The performance of each algorithm is evaluated as the difference between the reputation after attacks and the reputation before the attacks. The difference between the vectors of reputation is as shown in equation (9). We expect when using our improved algorithm, the Reputation Change Rate (RCR) will be smaller than from the True-Reputation change rate due to the reduced influence of the attacks. We adopted the RCR formula from [18]:

$$\text{RCR} = \frac{|Reputation\ with\ RA - Reputation\ witho\quad RA|}{Reputation\ witho\quad RA} \qquad (15)$$

V. RESULTS

The results (RCR values of each algorithm) are shown in format of four graphs per model. Two top graphs are for push attack and two bottom graphs are for nuke attack. Two left graphs are for lower amount ratings of the attackers (2 on target-only model and 50 on other models), two right graphs are for higher amount ratings of the attckers (32 on target-only model and 100 on other models).

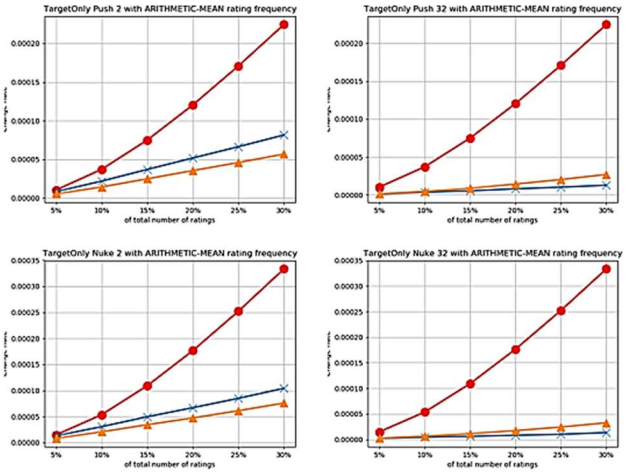

Figure 7. Target-Only Model.

Figure 7 illustrates the superiority of our improved algorithm over True-Reputation both on push and nuke models, but only when the amount of attack ratings is small.

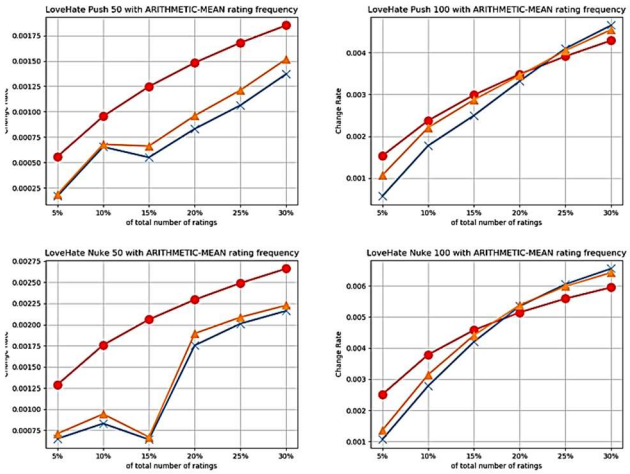

Figure 8. Love/Hate Model.

Figure 8 does not illustrate a clear advantage of our improved algorithm over the True-Reputation algorithm on Love/Hate model.

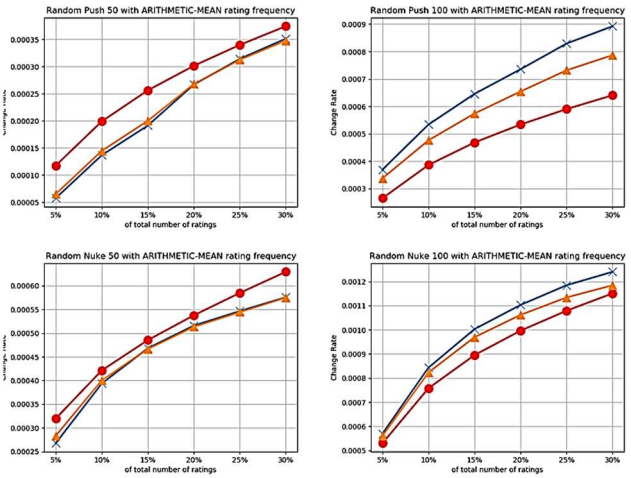

Figure 9. Random Model.

Figure 9 illustrates an equality between our improved algorithm and True-Reputation algorithm when the amount of rating attacks is small, although when the amount is increasing, our improved algorithm clearly outperforms the True-Reputation algorithm.

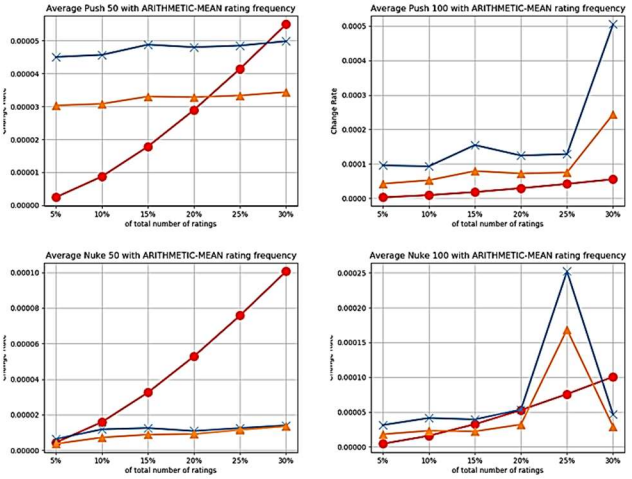

Figure 10. Average Model.

Figure 10 illustrates totally superiority of our improved algorithm over the True-Reputation. Average model is the most sophisticated attacking model. It uses an internal information to camouflage itself as an ordinary user. Moreover, the more intense the attack was, the more impressive our algorithm was.

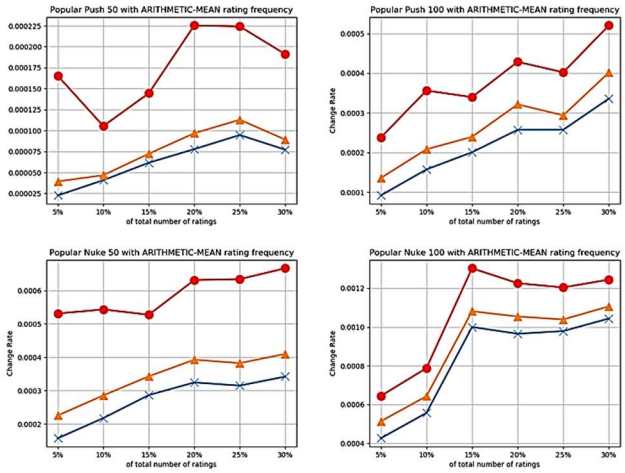

Figure 11. Popular Model.

Figure 11 illustrates superiority of True-Reputation algorithm over our improved algorithm on the popular model.

Attack Model	Arithmetic Mean Average RCR	True-Reputation Average RCR	Our Improved Algorithm Average RCR	Percentage of Improvement
Average Nuke 100	0.00005643	0.00009291	0.00005872	58.24%
Average Nuke 50	0.00005643	0.00001366	0.00001082	26.20%
Average Push 100	0.00003089	0.00021987	0.00011261	95.24%
Average Push 50	0.00003089	0.00005719	0.00003896	46.79%
LoveHate Nuke 100	0.00550944	0.00519719	0.00533871	-2.65%
LoveHate Nuke 50	0.00251142	0.00161080	0.00170752	-5.66%
LoveHate Push 100	0.00371695	0.00338241	0.00364170	-7.12%
LoveHate Push 50	0.00155482	0.00092805	0.00104422	-11.13%
Popular Nuke 100	0.00128185	0.00099422	0.00108750	-8.58%
Popular Nuke 50	0.00070707	0.00032878	0.00040801	-19.42%
Popular Push 100	0.00045723	0.00026029	0.00032021	-18.71%
Popular Push 50	0.00021117	0.00007495	0.00009146	-18.05%
Random Nuke 100	0.00108220	0.00118909	0.00114697	3.67%
Random Nuke 50	0.00059554	0.00055346	0.00055646	-0.54%
Random Push 100	0.00057707	0.00080144	0.00071238	12.50%
Random Push 50	0.00031787	0.00026378	0.00026769	-1.46%
TargetOnly Nuke 2	0.00018800	0.00007001	0.00004962	41.09%
TargetOnly Nuke 32	0.00018800	0.00000900	0.00001897	-52.55%
TargetOnly Push 2	0.00012767	0.00005341	0.00003656	46.08%
TargetOnly Push 32	0.00012767	0.00000839	0.00001536	-45.35%
Summary (Average)	0.00096643	0.00083300	0.00080500	6.93%

Figure 12. Summary RCR values from the all experiments.

Percentage of Improvement = $\frac{Improved\,TR - Original\,TR}{Improved\,TR}$

Figure 12 illustrates bird's- eye view on results from the all experiments. Summarizing all experiments, our improved algorithm, have superiority over the original True-Reputation algorithm with about 7%. We have achieved decisive superiority in the most difficult models of reputation attack's detection, such as Random and Average attack models. Moreover, the more intense the attack was in these models, the more impressive our algorithm was.

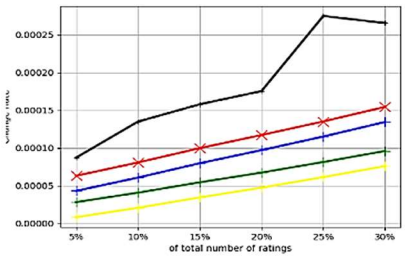

Figure 13. Experiments on various configurations.

True-Reputation with const & percentile cutoff (Black)
Original True-Reputation (Red line)
True-Reputation with user age (Blue line)
True-Reputation with item age (Green line)
True-Reputation with user and item age (Yellow line)

Note that during the research we examined additional parameters for improving the algorithm, such as const and percentile cutoff. For example, we used 20th percentile in order to cut off the less 20% of confident ratings (on assumption that they are intensively biased). In our experiments, the configuration with the best performance was the combination of user age with item age. We tried also various another configurations but they did not meet the expectations.

VI. DISCUSSION

On the one hand, we did not achieve inconclusive results in the experiments. There have been experiments with our total and impressive superiority, were experiments with our minor superiority, and there have been experiments with equal performance or even with a superiority of a True-Reputation algorithm. However, on the other hand we gained a total superiority of all experiments of about 7% over the original True-Reputation algorithm. Moreover, we have achieved decisive superiority precisely in the most sophisticated attacks models - such as Random and Average attack models. In these models, the reputation attacker tries to camouflage himself as an ordinary user by generating lots of ratings for random movies with close ratings to their reputation. During the experiments, the more intense the camouflage attack was, the more impressive our algorithm was.

In our opinion, we obtained superiority over the original True-Reputation algorithm by focusing on the more attractive movies for the attack (new movies released in recent years). We believe it is inaccurate to treat all users and items equally regardless of their seniority. Our assumptions are that reputation attack firms do not work for free and they must be compensated for their work. This means that their target items should be attractive enough for production companies to pay for their promotion, and there is no logical reason to pay for reputation attack on ancient movie.

In addition, we focused on users who are more likely to be used as fictitious profiles by the reputation attackers (new users that registered in the last month). We believe that there are substantial technical reasons for the reputation attackers to use new fictitious profiles in their attacks on the recommender system. First, when a reputation attacker's firm receives an attack task, it is not certainly that it already has profiles in the requested recommender system or that it has profiles that suitable to the requested task. Second, it takes too much effort to properly maintain fictitious profiles over time. Without properly maintenance, at any given moment the recommender system will be able to identify the fictitious profile and remove it from the system. There are many possibilities to suspect the fictitious profile, for example – suspicious profile activity, a long time without activity, suspicious traffic signatures or failure in verification of personal information. Third, in light of the fact that the interests of users are constantly changing, there are many collaborative recommender systems that use various types of decay functions to normalize the ratings and to adapt the recommendations to the users' current interests. In this situation, there is a significant interest of reputation attackers to use new fictitious profiles for gaining a maximal effect for their attacks.

VII. CONCLUSION

This paper describes the scenarios in which a false reputation could occur and depicts the potential damage it could cause to the recommender system. The understanding of why and when a false reputation occurs helps us establish experimental situations and improving the existing algorithms. In order to find an optimal solution to the false reputation problem, we surveyed social and technical researches in the field of rating's attack and examined various attributes with the potential to detect the attacks. Eventually, we based our algorithm on the sketch of the True-Reputation algorithm and we focused in our research on two leading additional attributes – user age and item age. In the end, the combination of the two attributes gave us the best performance. Through various and extensive experiments, we show that our improved algorithm can reduce the influence of various rating's attacks models, especially the most sophisticated ones.

We believe that future work in this area consists of two main directions: the first is the examination of the existing attributes into additional domains (it is possible in the first stage to expand into nearby entertainment domains such as books and songs). The main goal of expansion into new domains is to deeper base our theories and to make sure we have not created over-fitting problem around a movies domain. Second direction is to continue a demanding research for new attributes that characterize the constantly changing rating's attackers. We believe that a great boost can be achieved on researching the real attacked dataset. Reality is beyond imagination and the real data never resemble the simulated ones.

ACKNOWLEDGMENTS

We would also like to express our gratitude to Prof. Bracha Shapira and Mrs. Liat Antwarg of Ben-Gurion University for sharing with us the pearl of their wisdom during this research and throughout the course of Advanced Topics in Recommender Systems.

REFERENCES

[1] S. Aygün and S. Okyay, ''Improving the pearson similarity equation for recommender systems by age parameter,'' in Proc. AIEEE, Riga, Latvia, Nov. 2015.
[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.
[3] M. Brennan, S. Wrazien, and R. Greenstadt, "Using machine learning to augment collaborative filtering of community discussions," in Proc. 9th Int. Joint Conf. Auton. Agents Multiagent Syst. (AAMAS), Toronto, ON, Canada, 2010, pp. 1569–1570.
[4] P. Chirita, W. Nejdl, and C. Zamfir, "Preventing shilling attacks in online recommender systems," in Proc. 7th Annu. ACM Int. Workshop Web Inf. Data Manage. (WIDM), Bremen, Germany, 2005, pp. 67–74.
[5] Y. Ding and X. Li, "Time weight collaborative filtering," in Proc. ACM CIKM '05, Bremen, Germany, Oct. 2005, pp. 485–492
[6] Y. Ding, X. Li, and M. E. Orlowska, "Recency-based collaborative filtering," in Proc. ADC '06, Hobart, Australia, Jan. 2006, pp. 99–107
[7] G. Häuubl and V. Trifts, "Consumer decision making in online shopping environments: The effects of interactive decision aids," Market. Sci., vol. 10, no. 1, pp. 4–21, 2000.
[8] 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.
[9] M. Limayem, M. Khalifa, and A. Frini, "What makes consumers buy from Internet? A longitudinal study of online shopping," IEEE Trans. Syst., Man, Cybern. A, Syst., Humans, vol. 30, no. 4, pp. 421–432, Jul. 2000.
[10] S. Liu, J. Zhang, C. Miao, Y. Theng, and A. Kot, "iCLUB: An integrated clustering-based approach to improve the robustness of reputation systems," in Proc. 10th Int. Joint Conf. Auton. Agents Multiagent Syst. (AAMAS), Taipei, Taiwan, 2011, pp. 1151–1152.

[11] 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.
[12] The Nielsen Company (2015). Nielsen report of June 2015, Looking to achieve new product success, New York, USA.
[13] O'MAHONY, M., HURLEY, N., KUSHMERICK, N., AND SILVESTRE, G. 2004. Collaborative recommendation: A robustness analysis. ACM Trans. Inter. Tech. 4, 4, 344–377.
[14] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors. Recommender Systems Handbook. Springer, 2011.
[15] E. Santos and D. Li, "On deception detection in multiagent systems," IEEE Trans. Syst., Man, Cybern. A, Syst., Humans, vol. 40, no. 2, pp. 224–235, Mar. 2010.
[16] Sarwar, B., et al. 2001. Item-based collaborative filtering recommendation algorithms, in Proceedings of the 10th international conference on World Wide Web, ACM: Hong Kong, Hong Kong. p. 285-295.
[17] A. Whitby, A. Jøsang, and J. Indulska, "Filtering out unfair ratings in Bayesian reputation systems," Icfain J. Manage. Res., vol. 4, no. 2, pp. 48–64, 2005.
[18] Oh HK, Kim SW, Park S, Zhou M (2015) Can you trust online ratings? A mutual reinforcement model for trustworthy online rating systems. IEEE Trans Syst Man Cyber Syst 45(12):1564–1576
[19] P. Wu, C.H. Yeung, W. Liu, C. Jin, Y.-C. Zhang, Time-aware collaborative filtering with the piecewise decay function, 2010.