Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 490261, 9 pages
http://dx.doi.org/10.1155/2015/490261

Research Article

Abnormal Profiles Detection Based on Time Series and Target Item Analysis for Recommender Systems

Wei Zhou,[1] Junhao Wen,[1,2] Min Gao,[2] Haijun Ren,[2] and Peng Li[3]

[1]*College of Computer Science, Chongqing University, Chongqing 400030, China*
[2]*School of Software Engineering, Chongqing University, Chongqing 400030, China*
[3]*Chongqing Health and Family Planning Commission, Chongqing 400000, China*

Correspondence should be addressed to Junhao Wen; wjhcqu@cqu.edu.cn

Received 12 December 2014; Accepted 18 May 2015

Academic Editor: Aime' Lay-Ekuakille

Collaborative filtering (CF) recommenders are vulnerable to shilling attacks designed to affect predictions because of financial reasons. Previous work related to robustness of recommender systems has focused on detecting profiles. Most approaches focus on profile classification but ignore the group attributes among shilling attack profiles. Attack profiles are injected in a short period in order to push or nuke a specific target item. In this paper, we propose a method for detecting suspicious ratings by constructing a time series. We reorganize all ratings on each item sorted by time series. Each time series is examined and suspected rating segments are checked. Then we use techniques we have studied in previous study to detect shilling attacks in these anomaly rating segments using statistical metrics and target item analysis. We show in experiments that our proposed method can be effective and less time consuming at detecting items under attacks in big datasets.

1. Introduction

With the development of information technology, people come from era of lack of information into information overload (information overload) era. In this era, consumers and information producers are experiencing a great challenge: For information consumers, it is difficult to find the information they are interested in from mass information; for information producers, it is also difficult to make their own information stand out. Recommended system is an important tool to resolve this contradiction. It is the recommender systems' task to contact the users and information; on the one hand, they help users find valuable information and on the other hand information can be displayed in front of the users who are interested in it. Finally, win-win information consumers and information producers relationship can be achieved. In recent years, recommender systems have become extremely common and are applied in a variety of applications. Recommender systems have been developed to recommend movies, music, news, books, research articles, social tags, and other products. Examples of recommender systems include Amazon.com and taobao.com. Recent research has shown that traditional CF algorithms are vulnerable to attacks [1–3]. Attackers influence systems by introducing biased profiles into the rating matrix so that their items are recommended to users more often. Shilling attacks can be divided as push and nuke attacks according to their intent, making a target item more or less likely to be recommended, respectively. These attacks can affect the quality of predictions and recommendations for many users, resulting in users that do not trust the system.

Recommender systems, which are built using data from existing users, are vulnerable to the injection of biased data otherwise known as attacks. It is difficult to prevent unscrupulous users from injecting fake data (profiles) into a system. There are several reasons for an attack to be carried out. A major reason is that there may be monetary incentive when an item is rated highly on a recommendation list. Individuals may be interested in promoting or demoting an item, which is known as a target item, by manipulating the recommender system. Most attacks can be implemented as follows. The attackers take on different identities within

the system and create a user profile for each identity. We refer to such profiles as attack profiles. Within each of the profiles created, the attacker would then manipulate the recommendation by rating or recommending a particular target item. In order to obfuscate themselves as genuine users in the system the attack profiles will contain ratings for nontarget items. These ratings can be selected in different ways either randomly or more intelligently if the attacker has prior knowledge of the ratings in the system. The attacker can make the system produce the recommendation behavior they desire by using this. Recent work has shown that even modest attacks are sufficient to manipulate the behavior of the most commonly used recommendation algorithm [1, 4].

To ensure the trustworthiness of recommender systems, attack profiles need to be detected and removed accurately. In this paper, we propose a novel technique based on statistical metrics and rating time stamps, called *TS-TIA* (target item analysis). The main contribution of this technique is that we divide the rating matrix into rating segments (windows) and find suspicious rating segments. We examine the suspected rating segments instead of the whole rating matrix. This reduces the algorithm complexity and time consumed to detect shilling attacks.

The paper is organized as follows. In next section we look at previous work in the area. In Section 3 we discuss preliminary knowledge used in this paper, including CF, attack models, detecting metrics, and how to construct time series. Section 4 describes the detailed metrics used in detecting attacks and algorithms and experimental methodology for our detection model. In Section 5 we present our experimental results and a conclusion in Section 6.

2. Related Work

In this section, we describe popular different attack models and discuss different statistical detection metrics and then summarize the related work on shilling attacks.

An attack consists of attack profiles that are introduced into the system in order to alter recommendation lists of a set of target items. There are two types of main attacks according to the intent of attackers. A push attack is an attack that aims to promote an item and boost its ranking, whereas a nuke attack is an attack designed to demote an item and lower its rankings.

The word "shilling" was first termed in [1]. There have been some recent research efforts aimed at detecting and reducing the effects of profile injection attacks [5, 6]. These attacks consist of a set of attack profiles, each containing biased rating data associated with a fictitious user identity. Since "shilling" profiles look similar to authentic profiles, it is difficult to identify them. Many attack profiles are based on random and average attack models which were introduced originally in Chirita et al. [4]. In this section we concentrate on research on attack detection in a CF recommender system. There are three categories of attack detection algorithms: supervised, unsupervised, and semisupervised.

In the first category, attack detection techniques are modelled as a classification problem. Most early detection algorithms [7] exploited signatures of attack profiles. These techniques were considered less accurate, since they looked at individual users and ignored the combined effect of such malicious users. Moreover, these algorithms do not perform well when the attack profiles are obscured. Some of these techniques use nearest neighbours classifiers, decision tree methods, rule based classifiers, Bayes classifiers, neural networks classifiers, or SVM based classifiers [8, 9].

In the second category, unsupervised detection approaches address these issues by training on an unlabeled dataset. These methods involve much less computational effort as compared to supervised approaches. The benefit of this is that these techniques facilitate online learning and improve detection accuracy. There has been significant research interest focused on detecting attack profiles using the unsupervised approach. Some of the techniques use clustering [10], association rules methods [11], and other statistical approaches [4, 12]. Zhang et al. [13] used a Singular Value Decomposition (*SVD*) method to learn a low-dimensional linear model. They examined the effectiveness of two popular attacks (random attack and average attack) on a model-based algorithm. The results show that the *SVD-based* algorithm is more resistant to attacks than memory-based algorithms. They examined approaches for detecting suspicious rating trends based on statistical anomaly detection. Zhou et al. [5, 14] proposed a novel technique for identifying group attack profiles which uses an improved metric based on Degree of Similarity with Top Neighbors (*DegSim*) and Rating Deviation from Mean Agreement (*RDMA*) using statistical strategy. They also extend a detailed analysis of target item rating patterns. The proposed methods improve the accuracy of detection using *TIA*.

In the third category, semisupervised detection approaches make use of both unlabelled and labelled user profiles for multiclass modelling. Wu et al. [15] proposed a system called *HySAD* for hybrid attack detection. In general, *HySAD* is a semisupervised learning system that makes use of both unlabeled and labeled user profiles for multiclass modeling.

3. Preliminary Knowledge

CF algorithms are widely used in recommender systems. But recommender systems using CF algorithms are vulnerable to attacks. Malicious users can utilize different methods to create shilling attack profiles and employ different techniques to make the detection of attack more difficult. Section 3.1 reviews two different types of CF algorithms, Section 3.2 introduces the different attack models, and Section 3.3 discusses different statistical detection metrics.

3.1. Collaborative Filtering. There are two kinds of CF algorithms: user-based and item-based collaborative filtering. In order to generate recommendations, both user-based and item-based CF recommender systems follow a regular process. Firstly, establish similarity or dissimilarity between users or items, and then weight the similarities to emphasize users (or items) that are most influential in establishing

Table 1: Features of the attack models.

Attack model	I_S (selected items)	I_F (filler items)	I_T (target items)
Random attack	∅	Random ratings	$r_{\max}/r_{\min}$
Average attack	∅	Mean of each item	$r_{\max}/r_{\min}$
Bandwagon attack	$r_{\max}$	Random ratings	$r_{\max}/r_{\min}$
Segment attack	$r_{\max}$	Random ratings	$r_{\max}/r_{\min}$

similarity or dissimilarity. Predictions are finally computed by taking users' (or items') ratings as well as their similarities into account.

Rating data is represented as a user × item matrix R, with $R_{u,i}$ representing the rating given by user u for item i, if there exists a rating on item i, or otherwise there will be a null value. Similarity between users is then computed using the Pearson correlation [16]:

$$W_{uv} = \frac{\sum_{i \in I} \left(R_{ui} - \overline{R}_u\right)\left(R_{vi} - \overline{R}_v\right)}{\sqrt{\sum_{i \in I} \left(R_{ui} - \overline{R}_u\right)^2 \left(R_{vi} - \overline{R}_v\right)^2}}, \quad (1)$$

where I is the set of items that users u and v both rated, R_{ui} is the rating user u gave to item i, and $\overline{R}_u$ is the average rating of user u.

3.2. Attack Models. An attack consists of attack profiles that are introduced into the system in order to alter recommendation lists of a set of target items. Target items are usually unpopular items (low average rating) that are not rated by many users. Shilling attacks can be divided into push and nuke attacks to make a target item more or less likely to be recommended, respectively. A push attack is an attack that aims to promote an item and boost its ranking, whereas a nuke attack is an attack designed to demote an item and lower its rankings. Based on different assumptions about the attacker's knowledge and purpose, a number of attack models have been identified, as described in [1, 4]. There are four popular attack models in recommender systems: random attack, average attack, bandwagon attack, and segment attack models. Ratings in an attack profile can be divided into three sets of items: a target item I_T, a selected item I_S, and a set of filler items usually randomly chosen I_F. Features of the attack models are shown in Table 1.

Filler items in a malicious profile are a set of items that make the profile look normal and makes a malicious profile harder to detect. The quality of the filler items depends on the existing knowledge gathered from the recommender system. As more knowledge is obtained, an attack generated is more sophisticated. The major difference of attack models is how the ratings of filler items and the selected items are determined. The differences among attack models are the variance rating distribution in filler items and the selected items.

3.2.1. Random Attack Model. Random attack model is a naive attack in which the injected profile rates the set of randomly chosen fillers using a normal distribution and the standard deviation around the average rating of the system, as described in [1]. Attackers then rate the set of target items with the maximum or minimum allowable rating based on the purpose of the attack. For example, if the rating scores for a recommender system are between 1 and 5, where 1 represents an unfavourable rating and 5 represents a favourable rating, an attacker would rate the target item at 5 for a push attack and rate the target item at 1 for a nuke attack.

3.2.2. Average Attack Model. Average attack model is a more sophisticated attack model than random attack model and requires knowledge of the average rating of each item in the recommender system. Attackers rate items in the filler set randomly using a normal distribution with average set to the average rating of the filler items being rated and the standard deviation, as described in [4]. By introducing the average attack model, attackers disguise themselves and are harder to differentiate when compared to genuine users and thus have a larger effect on recommendations. As with the random attack model, the ratings of target items are set to either the maximum or minimum allowable rating based on the purpose of the attack.

3.2.3. Segment and Bandwagon Attack Model. In addition to random and average attack models, several more sophisticated models have been studied. In this work we have evaluated two other models, the bandwagon and segment attacks. We call attacks that have selected items with maximum ratings group attacks. Segment attack and bandwagon attack with different selected sets can be seen as group attacks. The principle behind the group attack is that the best way to increase the cost/benefit of an attack is to target one's effort to those already predisposed towards one's product. In other words, it is likely that an attacker wishing to promote a particular item will be interested not in how often it is recommended to all users but how often it is recommended to likely buyers. The segment attack model is designed to push an item to a targeted group of users with known or easily predicted preferences. In the bandwagon attack model, the attacker using Zipf's law will build attack profiles containing those items that have high visibility. Such profiles will have a good probability of being similar to a large number of users, since the high visibility items are those that many users have rated.

3.3. Detecting Metrics. Attack profiles differ from that of genuine profiles in a statistical way. There are two main differences: (1) the rating given to the target item (items); (2) the rating distribution among the filler items. Due to this there are different metrics that have been proposed to

measure the difference between rating profiles. In this section we will look at two metrics [4], *RDMA* and *DegSim*. *RDMA* value of attack profiles is higher than that of genuine profiles, while *DegSim* value of attack profiles is lower than that of genuine profiles.

3.3.1. Rating Deviation from Mean Agreement. *RDMA* measures the deviation of agreement from other users on a set of target items, combined with the inverse rating frequency for these items. *RDMA* can be calculated in the following way:

$$RDMA_u = \frac{\sum_{i=0}^{N_u} \left(\left|r_{u,i} - \overline{r_i}\right| / NR_i\right)}{N_u}, \tag{2}$$

where N_u is the number of items user u rated, $r_{u,i}$ is the rating given by user u to item i, and NR_i is the overall number of ratings in the system given to item i.

3.3.2. Degree of Similarity with Top Neighbours. The *DegSim* attribute is based on the average Pearson correlation of the profile's k nearest neighbours and is calculated as follows:

$$DegSim = \frac{\sum_{u=1}^{k} W_{uv}}{k}, \tag{3}$$

where W_{uv} is the Pearson correlation between user u and user v and k is the number of neighbours.

3.4. Construct a Time Series. We determine the suspicious profiles by constructing a time series proposed in [13]. There are millions of profiles and items in real-time online systems; it is time consuming to carry out detecting on the whole dataset. We find that group attributes and time clustering characteristics exist in shilling attacks. Most of profiles and ratings are low suspected of being attackers. We intend to divide the whole dataset into subsections and find suspected subsections, and techniques we proposed in our previous studies are used focusing on the suspected subsections. We get a subset of suspicious profiles. Scope of attack profiles is reduced and will make the algorithm more efficient.

To construct the time series of the above measure for an item, we first sort all the ratings for the item by their time stamps into a data stream and then group every disjoint w consecutive ratings into a window. Here, w is referred to as the window size. For each window, we compute its sample average. Then we obtain a time series for the selected item. As sample average is asymptotically normal, we can decide whether a window is an anomaly by testing whether the absolute value of its z-score (the difference from the mean divided by the standard deviation) for sample average (sample entropy) is larger than a threshold. To extract useful information from the rating distribution of an item, the measure we use to capture the degree of dispersal or concentration of a rating distribution is the *sample entropy* $M(X)$, which is defined as

$$M(X) = \frac{\left(\sum_{i=1}^{r_{\max}} n_i \times i\right)}{S}. \tag{4}$$

In (4), i can be {1, 2, 3, 4, 5} in MovieLens Dataset, which is the integer score values of the rating matrix. n_i is the total number of ratings in a window histogram.

While

$$S = \sum_{i=1}^{r_{\max}} n_i, \qquad X = \left\{n_i, i = 1, \ldots, r_{\max}\right\}, \tag{5}$$

where X is an empirical histogram.

The standard score Z of a raw score X is

$$Z = \frac{X - \mu}{\sigma} \tag{6}$$

while

$$\mu = E(X), \tag{7}$$

where μ is the mean of the population; ρ is the standard deviation of the population

$$\sigma = \mathrm{Var}(X). \tag{8}$$

4. Detecting Profile Injection Attacks

In [7, 14], We proposed a novel technique for identifying group attack profiles which uses an improved metric based on Degree of Similarity with Top Neighbors (*DegSim*) and Rating Deviation from Mean Agreement (*RDMA*). We also extended our work with a detailed analysis of target item rating patterns. Experiments show that the combined methods can improve detection rates in user-based recommender systems. The efficiency of *TIA* methods became lower when the datasets increase.

Cost-benefit analysis is a widely used technique for deciding whether to undertake a course of action; it simply adds up the value of the benefits accruing from the action and subtracts the associated costs. Attackers usually inject profiles in a short period of time into the system in order to get high cost-benefit ratio and high prediction shift [17]. Sample average flows normal distribution when there are no attacks in the raw rating matrix. The value will be abnormal when attack profiles are injected in a short period of time. Figure 1 shows that z-score stands out clearly through the lens of sample average. This feature of attack profiles can be used to locate suspicious attack profiles segments (windows).

Algorithm 1 (*TS-TIA* Phase 1, Find suspicious rating segments (windows)).

Input. Rating matrix; Window size w; confidence level z;

Output. Suspicious rating segments *SUSSEG*;

(1) $SUSSEG = \emptyset$;

(2) $i \in I$, Sort ratings according to the time stamps;

(3) Divide ratings by window size w;

(4) For each window, sample average is calculated by (4) and (6);

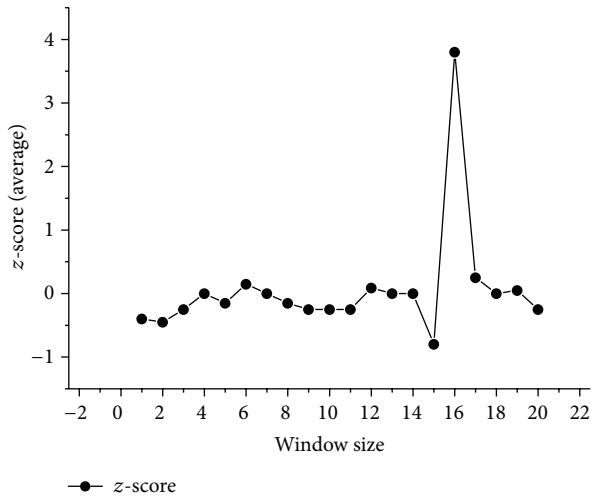

Figure 1: A push attack event stands out clearly through the lens of sample average. The window size is set to 50, and the sequence number of window that contains shilling attacks are from 14 to 19 (the confidence coefficient is set to 90%).

(5) Find abnormal rating segments by confidence level z;

(6) Find rating matrix of abnormal rating segments and put it in *SUSSEG*.

(7) **return** *SUSSEG*.

Our approach is divided into two phases. We first examine each data stream and find suspected rating segments in the data stream. In this phase, we determine suspected rating segments by constructing a time series. In the second phase, we use techniques in our previous study to detect shilling attacks in these anomaly rating segments using statistical metrics and *TIA* method. In the fine-tuning phase, target items in the potential attack profiles set are analyzed.

Algorithm 1 shows how suspicious rating segments (windows) are located using time series and abnormal z-score of sample average. In this phase, scope of attack profiles is greatly narrowed, which saves time and reduces the computing complexity. But genuine profiles and attack profiles are mixed together. We use the result of Algorithm 1 and the techniques we promoted in previous research to filter out genuine profiles.

Algorithm 2 (*TS-TIA* Phase 2, Apply *TIA* method on abnormal rating segments).

Input. The set of suspected profiles *SUSSEG*; item set I;

Output. Final detect result set *DetectedResult*;

(1) $SUSSEG = SUSSEG - RDMA_u \geq \epsilon_{RDMA} \cap DegSim_u \leq \epsilon_{DegSim}$

(2) $DetectedResult = \emptyset$;

(3) $\forall i \in I$, $count_i \leftarrow$ number of ratings in $item_i$ equal to r;

(4) **while** $\max(count) > \theta$ **do**

(5) $item_t \leftarrow \{item_i \mid count_i = \max(count)\}$;

(6) $\forall p \in SUS_{RD}$, $P \leftarrow p$ rate $item_t$ with r;

(7) $DetectedResult \leftarrow P \cup DetectedResult$;

(8) $SUS_{RD} \leftarrow SUS_{RD} - P$;

(9) **end while**

(10) **return** *DetectedResult*.

In the first phase, we find the suspected ratings on an item and then find suspected profiles that rated on the item during the specific time. In the second phase, we will apply our previous techniques to detect attack profiles using *DegSim* and *RDMA* metrics.

Overall attackers should have a high influence on the system in order to promote the target items effectively. However, there are three different features in attack profiles, which enable us to differentiate between genuine and attack profiles. Firstly, filler items are randomly chosen; thus the similarity based on these filler items between attack and genuine profiles should be lower. Secondly, since shilling attacks usually try to push items with low ratings or vice versa in nuke attacks, the users mounting such an attack will assign a rating that deviates from the average rating value assigned by the genuine profiles. Last but not least, all target items are assigned a highest or lowest value, and the count number of this value should be bigger than other values among items. Based on these three reasons, we choose the *RDMA* and *DegSim* metrics, which reveal these distinctive features in the rating patterns. Attackers should therefore have relatively high values for *RDMA*, as well as very low values in *DegSim*.

In this phase, an *RDMA* value for each profile is calculated. If the *RDMA* value for a profile u is above a maximum ϵ_{RDMA} threshold then we consider this profile as a suspicious profile:

$$RDMA_u = \frac{\sum_{i=0}^{N_u} \left(\left|r_{u,i} - \overline{r_i}\right| / NR_i\right)}{N_u} \geq \epsilon_{RDMA}. \quad (9)$$

From this process, we get a pool of suspicious profiles, SP_{RDMA}, which had *RDMA* values above the assigned threshold. We also calculate the *DegSim* value for each of the profiles. If the *DegSim* value for a profile u is below a minimum ϵ_{DegSim} threshold then we consider this profile as a suspicious profile:

$$DegSim = \frac{\sum_{u=1}^{k} W_{uv}}{k} \leq \epsilon_{DegSim}. \quad (10)$$

From this process, we get a pool of suspicious profiles, SP_{DegSim}, which had *DegSim* values below the assigned threshold. Lastly we consider the intersection between the pool of SP_{RDMA} and SP_{DegSim}, as our *SuspectedAttackers*:

$$SuspectedAttackers = SP_{DegSim} \cap SP_{RDMA}. \quad (11)$$

We set generous thresholds ϵ_{RDMA} and ϵ_{DegSim}, allowing more profiles to be considered as suspicious. We then filter out the misclassified profiles in the next phase.

Table 2: An example of rating matrix and attack profiles.

	Item_1	Item_2	Item_3	Item_4	Item_5	$\cdots$	Item_n
User_1	5	2	3	0	0	$\cdots$	5
User_2	2	0	4	1	2	$\cdots$	3
User_3	4	2	3	0	5	$\cdots$	0
User_4	0	3	0	3	4	$\cdots$	3
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
User_m	2	0	4	1	2	$\cdots$	3
Attacker_1	2	1	0	0	**5**	$\cdots$	4
Attacker_2	2	2	0	0	**5**	$\cdots$	3
Attacker_3	1	2	0	0	**5**	$\cdots$	2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Attacker_p	2	0	0	0	**5**	$\cdots$	4
Count (5)	2	2	2	2	9	$\cdots$	3

Since we have found *SuspectedAttackers Set*, in the next, we apply *TIA* method in Phase Two to refine the detecting result. For example, if 80% of the profiles in the suspicious pool rated an item with the highest (or lowest) possible rating, we consider that item as a target item. We then move all the profiles that rate the suspected target item with the highest (or lowest) rating in to the *Attackers* Set. These profiles are considered to be real attackers. The intuition behind this is that we believe that the attackers will have specific target items that they target when they commit an attack. They would rate target items with the highest or lowest possible rating depending on the type of attack. To detect the proportion, we use an absolute count threshold θ. For example, if θ is set to 20 and the count (highest rating) for an item $Item_i$ is greater than 20, then $Item_i$ is a target item, and the profiles that rated $Item_i$ with the highest rating are considered as attackers and moved to the *Attackers* pool.

Let us take push attack as an example. Table 2 is an example of a rating matrix and attack profiles. The matrix is an $m \times n$ matrix. Each row in the matrix is the rating for the m items by a user. Table 2 shows genuine user profiles from $User_1$ to $User_m$ and attackers profiles from $Attacker_1$ to $Attacker_p$. The last row is the count number of rating 5; in this example, $Item_5$ is the target item.

5. Experiments

The datasets used in the experiments are the widely used *MovieLens* Datasets, including *MovieLens* 100 K Dataset, 1 M and 10 M Dataset by the *GroupLens* Research Project at the University of Minnesota and a subset of *Netflix* dataset (http://www.netflixprize.com/). There are 100,000 ratings (1–5) from 943 users on 1682 movies in MovileLens 100 k Dataset. There are 1,000,000 ratings from 6,000 users on 4,000 movies in MovieLens 1 M Dataset and there are 10 million ratings in 10 M dataset. Each user has rated at least 20 movies. Each user can rate a movie from 1 to 5, where 1 is the lowest and 5 is the highest. The platform we implement all the experiments is as flows: hardware: CPU Intel Core i7 processors; software: Windows 7 with 16G RAM and all of our tests are implemented on Matlab 2012b platform.

5.1. Parameters Used in the Detection Method. In this section parameters used in the detection process will be discussed, including the value k in *kNN*, threshold values for ϵ_{RDMA} and ϵ_{DegSim}, and the absolute count threshold value θ in the second phase of *TS-TIA*. We have introduced the process of detecting the attacks in Section 4. Since Phase 2 is based on the result of Phase 1, the result of Phase 1 has a vital impact on the final result. Tests showed that if there exist false positives in the *SuspectedAttackers Set* from Phase 1, the false positives have a chance to be removed in Phase 2; but if there exist false negatives in Phase 1, it would be impossible for us to detect the false negatives in Phase 2 and there would be false negatives in the final result. If we remove the false positive profiles which were genuine profiles from a large set of profiles, the impact on the final recommender system is negligible. There may be a cumulative effect later on. So we would rather there exist false positives than false negatives in the detecting result [17]. Considering this fact, on choosing the threshold values, a heuristic approach applied was that we are comfortable with a lower false negative value even though the false positive value is higher.

In choosing the threshold values of *RDMA* and *DegSim*, we would like to get values that have high separability, because it is easier to distinguish between genuine and attack profiles when there is high separability. We adjusted parameters so that it produces lower false negatives and considerable false positives. In these experiments we noticed that the intervals between *DegSim* values for the profiles were smaller when compared to the intervals between *RDMA* values for the profiles.

We set the thresholds for ϵ_{RDMA} and ϵ_{DegSim} as

$$\epsilon_{DegSim} = \lambda \sum_{u=1}^{n} \frac{DegSim_u}{n},$$
$$\epsilon_{RDMA} = \gamma \sum_{u=1}^{n} \frac{RDMA_u}{n}. \tag{12}$$

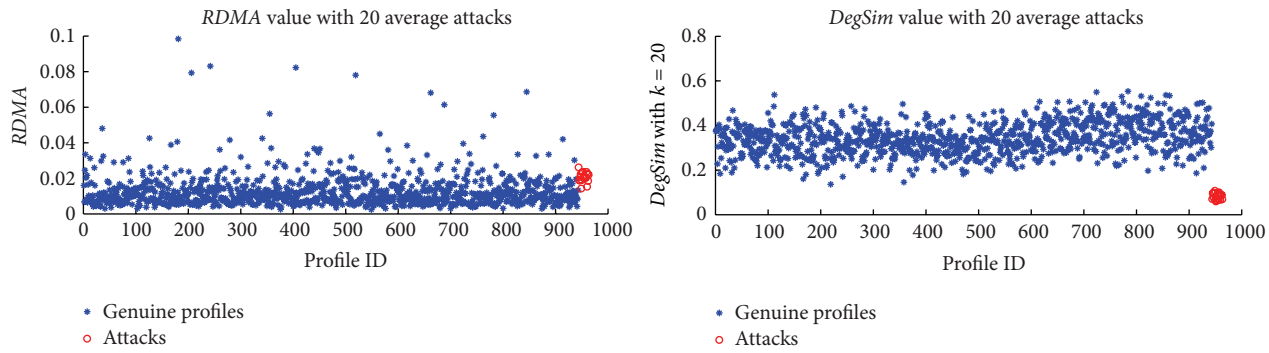

Figure 2: *RDMA* and *DegSim* value distribution with average attacks.

We choose a different weight for λ and γ. In the experiments we have done, we were more comfortable with the result when $\lambda = 1$ and $\gamma = 0.6$. Setting these weights we notice that the false negatives rate is lower and there are hardly any false positives. As we pointed out previously the threshold values of *RDMA* and *DegSim* are generous, thus allowing false negative profiles into the set of *SuspectedAttackers*.

In the second phase of our method, we need to set the threshold θ. We choose the threshold value based on the assumption that if attackers want to make a big prediction shift and *MAE* shift [1, 18] to the system and push a target item up, a certain number of injected attack profiles are required. From this assumption we can calculate the upper bound threshold that is necessary for a shift to occur.

In this experiment, an average size attack requires the number of attack profiles injected to be greater than 20 for a prediction shift and *MAE* shift [1]. To be conservative we chose a lower θ value of 6 in case of small scale attacks. Figure 2 shows the *RDMA* and *DegSim* value distribution in the random attack model, with the attack size of 20, filler size of 5%, and $k = 20$ in *DegSim*.

5.2. Experiments. In order to simulate real attacks in recommender systems, we injected attack profiles generated by certain attack models. In the experiments we varied two different variables: the attack size and the filler size. Because the median filler size of all profiles is 3%, we did not consider situations where the filler size is greater than 10%. We only consider attack size between 1% and 10% because it is realistic in a real world scenario. On the other side, the effects of attacks are hard to distinguish when the attack size is small [1]. In order to get certain prediction shift, we set the minimum number of attack profiles to 20. We varied the attack size from 20 to 200. We also varied the filler size from 1% to 10%.

This section is divided into three subsections. In the first subsection we introduce the experiment metrics. In the second we test the performance of our technique when filler size and attack size of attack profiles vary. In the third subsection we compared the performance of our technique with other approaches.

5.2.1. Experiment Metrics. To evaluate the performance of our technique we use two metrics: *detection rate* and *false positive rate*. These metrics are standard metrics used in similar experiments [18]. Detection rate is defined as the number of detected attacks divided by the number of attacks:

$$\text{Detection Rate} = \frac{\#\text{True Positives}}{\#\text{Attack}}. \tag{13}$$

False positive rate is the number of genuine profiles that are predicted as attacks divided by the number of genuine profiles:

$$\text{False Positive Rate} = \frac{\#\text{False Positives}}{\#\text{Genuine Profiles}}. \tag{14}$$

5.2.2. Experiments Result and Comparisons. In the first phase of our proposed method, sample average is used to locate suspicious attacks. In the experiment, we inject synthetic attack profiles with filler size 3%, which is the median of attack profiles with different attack size from 20 to 200. ∂ is defined as follows:

$$\partial = \frac{a}{A}, \tag{15}$$

where a means attackers that fall into the suspicious windows while A stands for number of attackers injected into the dataset. Figure 3 shows variation of ∂ when the confidence coefficient varies. In Figure 3, value of ∂ is relevant with confidence coefficient and attack size. value of ∂ increases when attack size and confidence coefficient increase. In general, ∂ value is greater when confidence coefficient increases under the same situation. On the other side, when choosing greater confidence coefficient, more genuine profiles are misjudged. When choosing lower confidence coefficient value, there would be more attackers that are misjudged as genuine profiles. This situation is worse because if these profiles are misjudged as genuine profiles, it is impossible to be detected in the final result using the method. As overall consideration, we choose 90% confidence coefficient in this paper.

Figure 4 shows the attack detection ratio when attack size varies under confidence coefficient 90%. The tendency of Figure 4 looks much like Figure 3. The detection rate increases with the increase of attack size. Detection rate of higher filler size is greater than that of lower filler size under the same attack size when attack size is below 100. The false

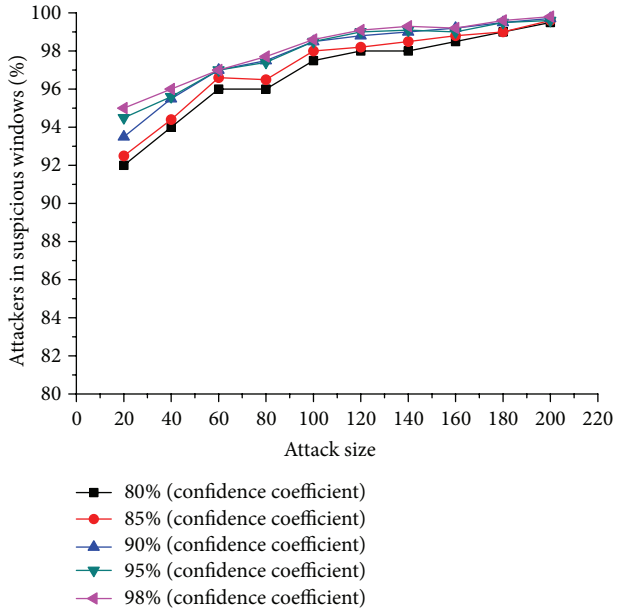

Figure 3: Attackers in suspicious rating segments ratio in Phase 1 when attack size and confidence coefficient vary.

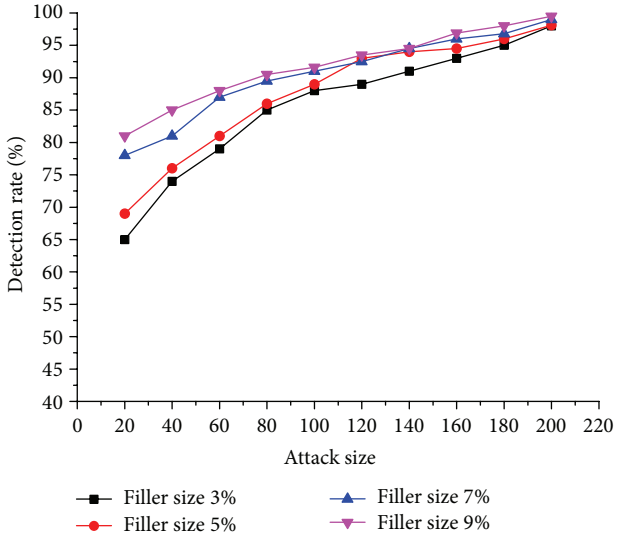

Figure 4: Attack detection ratio when attack size varies under confidence coefficient 90%.

positive rate becomes greater when attack size increased. But the false positive rate is low as a whole.

Figure 5 shows detection rate of different *TIA* algorithms when filler size is average and attack size varies. Detection rate of all algorithm becomes higher when attack size increases. $\beta\rho$-based method gets better detection rate than other *TIA*-based methods. All of the four methods get nearly 100% detection rate when attack size is over 120. False positive rate of *TS-TIA* is higher than that of *DeR-TIA* and *RD-TIA*. $\beta\rho$-based method gets highest false positive rate.

Figure 6 shows time consuming of different algorithms when attack size varies. All algorithms consume more time

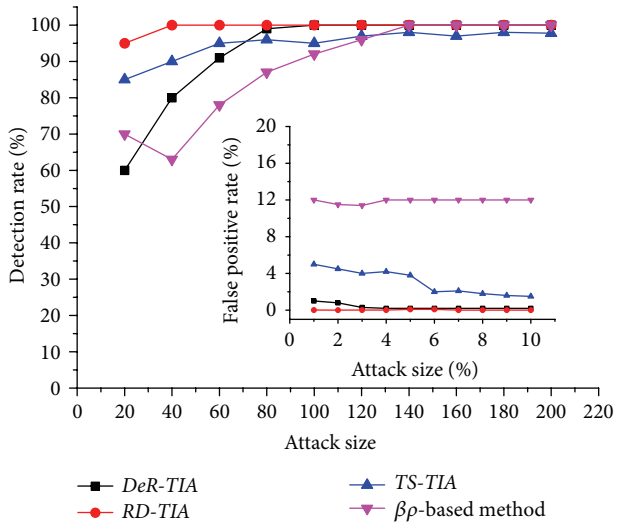

Figure 5: Detection rate of different target item analysis algorithms when filler size and average and attack size vary.

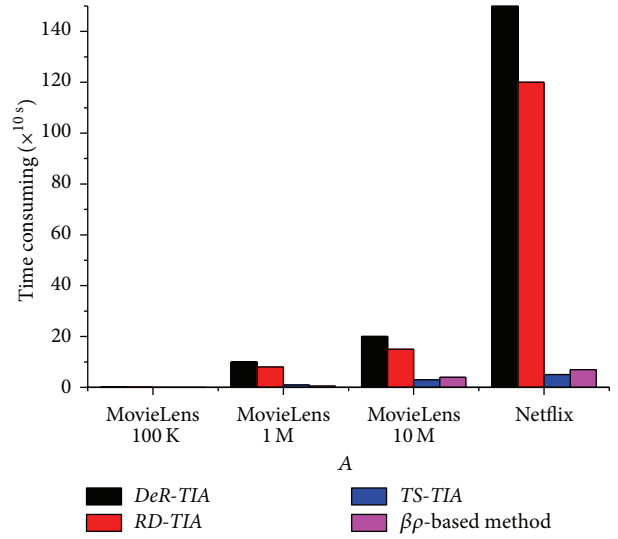

Figure 6: Time consuming of different algorithms when attack size varies.

when detecting bigger datasets. *DeR-TIA* consumes the most time in all detections and *RD-TIA* gets the second most time consuming, which is intolerable in big datasets. *TS-TIA* consumes the least time. $\beta\rho$-based method consumes more time than *TS-TIA* but less than the other two *TIA*-based methods.

6. Conclusion

Recommender systems suffer from attacks from malicious users because of their open nature. There exist group features between attack profiles, including rating a target item with the same score in a short period of time. Existing detecting methods do not use group features. We also construct a data stream by sorting ratings of an item, which saves time

and reduces the computing complexity. We proposed an algorithm called *TS-TIA* in the paper, using two statistical metrics and time series of attack ratings to detect anomaly profiles based on their rating patterns. We compared the proposed method *TS-TIA* with our previous research [8] and other unsupervised methods, and experiments showed that our results get better results in big datasets and occupy less computing capacity. In the future, we will focus on dynamic monitoring in real-time systems.

Disclosure

Parts of the paper have been published in IJCNN 2014 (International Joint Conference on Neural Networks).

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The research reported in this paper is supported by the NSFC under Grant no. 61379158, and the Ph.D. Programs Foundation of Ministry of Education of China under Grant no. 20120191110028, and Fundamental Research Funds for the Central Universities under Grant no. CDJZR13 09 551 and no. 106112014CDJZR095502, it is also supported by Medical Research Project of Chongqing Health and Family Planning Commission under Grant no. 20142124.

References

[1] S. K. Lam and J. Riedl, "Shilling recommender systems for fun and profit," in *Proceedings of the 13th International World Wide Web Conference Proceedings (WWW '04)*, pp. 393–402, May 2004.

[2] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 5–53, 2004.

[3] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web*, pp. 285–295, May 2001.

[4] P.-A. Chirita, W. Nejdl, and C. Zamfir, "Preventing shilling attacks in online recommender systems," in *Proceedings of the 7th ACM International Workshop on Web Information and Data Management (WIDM '05)*, pp. 67–74, November 2005.

[5] W. Zhou, Y. S. Koh, J. Wen, S. Alam, and G. Dobbie, "Attack detection in recommender systems based on target item analysis," in *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '14)*, pp. 955–958, Gold Coast, Australia, July 2014.

[6] M. P. O'Mahony, N. J. Hurley, and G. C. M. Silvestre, "Detecting noise in recommender system databases," in *Proceedings of the 11th International Conference on Intelligent User Interfaces (IUI '06)*, pp. 109–115, February 2005.

[7] M. P. O'Mahony, N. Hurley, and G. C. M. Silvestre, "Promoting recommendations: an attack on collaborative filtering," in *Proceedings of the 13th International Conference on Database and Expert Systems Applications (DEXA '02)*, pp. 494–503, 2002.

[8] M. Grčar, B. Fortuna, D. Mladenič, and M. Grobelnik, "*kNN* versus SVM in the collaborative filtering framework," in *Data Science and Classification*, Studies in Classification, Data Analysis, and Knowledge Organization, pp. 251–260, Springer, Berlin, Germany, 2006.

[9] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in Artificial Intelligence*, vol. 2009, Article ID 421425, 19 pages, 2009.

[10] L. Fu, D. H.-L. Goh, S. S.-B. Foo, and J.-C. Na, "Collaborative querying through a hybrid query clustering approach," in *Digital Libraries: Technology and Management of Indigenous Knowledge for Global Access*, vol. 2911 of *Lecture Notes in Computer Science*, pp. 111–122, Springer, Berlin, Germany, 2003.

[11] C.-H. Lee, Y.-H. Kim, and P.-K. Rhee, "Web personalization expert with combining collaborative filtering and association rule mining technique," *Expert Systems with Applications*, vol. 21, no. 3, pp. 131–137, 2001.

[12] N. Hurley, Z. Cheng, and M. Zhang, "Statistical attack detection," in *Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys '09)*, pp. 149–156, October 2009.

[13] S. Zhang, A. Chakrabarti, J. Ford, and F. Makedon, "Attack detection in time series for recommender systems," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '06)*, pp. 809–814, August 2006.

[14] W. Zhou, J. Wen, Y. S. Koh, S. Alam, and G. Dobbie, "Attack detection in recommender systems based on target item analysis," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '14)*, pp. 332–339, Beijing, China, July 2014.

[15] Z. Wu, J. Wu, J. Cao, and D. Tao, "HySAD: a semi-supervised hybrid shilling attack detector for trustworthy product recommendation," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*, pp. 985–993, August 2012.

[16] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: an open architecture for collaborative filtering of netnews," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pp. 175–186, October 1994.

[17] N. J. Hurley, M. P. O'Mahony, and G. C. M. Silvestre, "Attacking recommender systems: a cost-benefit analysis," *IEEE Intelligent Systems*, vol. 22, no. 3, pp. 64–68, 2007.

[18] S. Zhang, Y. Ouyang, J. Ford, and F. Makedon, "Analysis of a low-dimensional linear model under recommendation attacks," in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 517–524, August 2006.