Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 31 (2014) 165 – 174

Procedia Computer Science

Information Technology and Quantitative Management, ITQM 2014

A novel shilling attack detection method

Alper Bilge[a], Zeynep Ozdemir[a], Huseyin Polat[a,*]

[a]*Computer Engineering Department, Anadolu University, 26470 Eskisehir, Turkey*

Abstract

Recommender systems provide an impressive way to overcome information overload problem. However, they are vulnerable to profile injection or shilling attacks. Malicious users and/or parties might construct fake profiles and inject them into user-item databases to increase or decrease the popularity of some target products. Hence, they may have an effective impact on produced predictions. To eliminate such malicious impact, detecting shilling profiles becomes imperative. In this work, we propose a novel shilling attack detection method for particularly specific attacks based on bisecting *k*-means clustering approach, which provides that attack profiles are gathered in a leaf node of a binary decision tree. After evaluating our method, we perform experiments using a benchmark data set to analyze it with respect to success of attack detection. Our empirical outcomes show that the method is extremely successful on detecting specific attack profiles like bandwagon, segment, and average attack.

Selection and peer-review under responsibility of the Organizing Committee of ITQM 2014.

Keywords: Detection, shilling attacks, bisecting clustering, recommender systems, accuracy

1. Introduction

Customers want to buy the most liked items through online vendors. However, there are too many choices. Collaborative filtering (CF), one of the recommendation techniques, helps users select appropriate products. It is used to deal with information overload problem by producing highly accurate predictions. The major assumption of CF techniques is that users having similar experiences on past items are tend to agree on new items [?]. CF systems utilize a very sparse $n \times m$ user-item matrix, which includes n users' preferences about m products. The systems produce recommendations to their users by evaluating other like-minded users' preferences.

CF methods are successful at providing accurate referrals about products [?]. They are also able to overcome information overload problem by matching users with right items for them. However, they are vulnerable to profile injection or shilling attacks [?]. There might be malicious users and/or companies that aim to manipulate the recommender systems' outcomes on behalf of their advantages. Some malicious entities might want to increase the popularity of particular target items. Similarly, others might want to decrease the popularity of some other target products. To manipulate the popularity of target items, bogus profiles are created and injected into the system's database. If such malicious profiles are not detected by the system, they are then able to affect the accuracy of the predictions depending on robustness of the recommendation algorithm. The quality of predictions depends

*Corresponding author. Tel.: +902223213550; fax: +902223239501.
E-mail address: polath@anadolu.edu.tr.

Selection and peer-review under responsibility of the Organizing Committee of ITQM 2014.
doi:10.1016/j.procs.2014.05.257

on data quality. In other words, producing accurate predictions is possible with high quality data. Bogus profiles make data quality worse and that leads to inaccurate recommendations. Therefore, detecting shilling profiles is imperative for the success of CF systems.

The attackers construct fake profiles by using information about CF systems. Two parameters used to design the attacks in general are called *filler size* and *attack size*. Filler size is related to the number of filled cells with fake ratings. Attack size is about the number of bogus profiles injected into the system. The values of such parameters affect the success of shilling attacks. After designing the attack profiles, the attackers insert them into the system's database as though they are authentic users. Hence, they manage to manipulate the popularity of a target item in favor of themselves. Shilling attacks can be categorized as push or nuke attacks according to their intend. Push attacks are designed to increase the popularity of a target item while nuke attacks are tend to decrease the popularity of a target item. It is almost impossible to prevent shilling attacks at all. In other words, it is more likely to have bogus profiles because it is very difficult to verify the identity of each customer over the Internet. As a rule of thumb, if you do not prevent an attack at all, you should detect them. Thus, detecting attack profiles is one of the effective ways of defending against such attacks [?] .

There are a number of shilling attack detection methods like statistical techniques, classification-based methods, unsupervised clustering-based schemes, variable selection, and other techniques [?] . While statistical methods focus on anomalies and try to detect outliers caused by suspicious rating profiles, classification-based techniques aim to detect malicious users based on generic attributes derived from each profile, which reveals hidden deviations from general trend of collected data. Clustering is used as an incremental method to detect malicious profiles in which the database is clustered periodically to detect significant changes in cluster centers and avoid profiles causing such alterations [?] . Since shilling profiles also resemble high similarity to genuine users, unsupervised clustering techniques are applied with several classification attributes to improve detection skills [?] .

We propose a novel approach to detect bogus profiles. In our method, a binary decision tree (BDT) is constructed by recursively clustering the training data to locate the fake attack profiles via bisecting k-means clustering algorithm. Utilizing the fact that shilling attack profiles are generated according to a certain strategy, which yields very similar profiles, we propose to recursively cluster user-item matrix and distinguish attack profiles by huddling them at some level. While forming a BDT via bisecting k-means clustering algorithm, we divide the matrix into two clusters at each level and calculate an intra-cluster correlation coefficient for each internal node. We hypothesize that internal nodes holding attack profiles demonstrate high intra-cluster correlation due to their high similarity among themselves. We continuously repeat such process until there remains at most a predefined number of users in any leaf node. Then, we traverse BDT to detect anomalies with intra-cluster correlation coefficients to detect and label the node holding all or most of the attack profiles.

The rest of the paper is organized as follows. We briefly describe previous related research in Section 2. A brief information about shilling attacks is presented in Section 3. In the next section, we describe our approach in detail. The experiments conducted to analyze the success of the approach and the empirical results are discussed in Section 5. Finally, we provide our conclusions and future research directions in Section 6.

2. Related Work

For the overall success of recommender systems, detecting shilling profiles is imperative. Hence, there are various studies focus on detecting bogus profiles. Many researchers propose different techniques such as statistical methods, classification-based techniques, unsupervised clustering-based schemes, variable selection, and other techniques [?] . The authors in [?] propose statistical anomaly detection technique, which is based on item average values, to detect anomalies in user-item matrices. Those items that do not comply with the general behavior of the data are determined using statistical process control techniques. In another study [?] , Neyman-Pearson statistical detection theory-based shilling attack detection method is proposed. Another method proposed to detect bogus profiles is based on probabilistic Bayesian network models [?] . Statistical process control-based attack detection method is proposed by [?] . The method utilizes the user deviations from the average of rating numbers to create control chart, which is then used to detect fake profiles according to the warning rules of the chart.

Generic attributes-based classification techniques are also widely used to detect malicious users [?] . Deviation from mean agreement, degree of similarity, weighted deviation from mean agreement, weighted degree of agreement,

and length variance are examples of such generic attributes. Bogus profiles are detected based on generic attributes derived from each individual profile. In addition to generic attributes, attack model specific attributes are utilized to detect fake profiles [?]. To employ generic or model specific attributes as part of classification, different classification algorithms like kNN, C4.5, and SVM classifiers are utilized [??]. Cao et al. [?] propose an algorithm depends on semi-supervised learning to detect bogus profiles. They use naïve Bayes as initial detection method for labeled users and use EM-λ to strengthen effect of the method on unlabeled users.

Clustering can be used to filter out malicious users or fake profiles [?]. The main idea behind clustering is to cluster the user-item matrix periodically and check whether cluster centers are changing significantly or not. Significant changes are most probably caused by shilling profiles. PLSA-based clustering is employed to group users for determining those users whose data can be used for prediction generation [?]. The authors in [?] utilize a clustering algorithm based on statistical characteristics of data set to detect attack profiles by applying k-means clustering on produced profiles. Two clustering-based algorithms, CLUTR (clustering by using trust) and WCLUTR (clustering with weighed similarities derived from trust), are proposed to percolate suspect bogus profiles and to improve the robustness of CF algorithms [?]. Chakraborty and Karforma [?] propose three strategies to detect bogus profiles. Their approaches depend on outlier analysis. Their first strategy is based on PAM clustering, which is used to determine whether a new user is fake or not. The method detects bogus profiles with large number of filler items with high success. However, it is not very successful for finding outliers with small number of filler items. Additionally, they generate an approach to detect bogus profiles in large clusters. Also, they propose an angle-based outlier detection algorithm to detect bogus profiles in the database. Although clustering is effectively used to detect fake profiles, it might not be easy to differentiate bogus profiles from real profiles because malicious and genuine users might be very similar to each other. To overcome this challenge, PCA-based variable selection method is proposed to determine malicious users by computing covariance between users [?]. Examples of other methods proposed to detect attack profiles are time interval-based method [?] and data lineage method [?].

As presented above, there are different studies in the literature focusing on detecting attack profiles due to its importance. In addition to developing attack strategies and designing different attacks, it is also vital to form detection schemes. Besides other methods, clustering is widely used for determining malicious users or profiles. Since clustering can be effectively utilized as a detection scheme, we propose a novel clustering-based approach, which is very effective to detect attack profiles having a specific aim. To the best of our knowledge, this study is the first one that employs bisecting clustering method for attack profile detection. Also, our study is an explicit approach, which means that there is no need for preliminary information about attacks to detect them.

3. Shilling Attacks

Shilling attack profiles should be designed in such a way so that they serve their intended aims. They are constructed according to different attack models using the knowledge of the CF system, its rating database, its products, and/or its users [?]. Push attacks aim to increase the popularity of a target item so that the recommender system returns it as the recommended item to its users. Conversely, nuke attacks try to decrease the popularity of a target item to make it less likely suggested. A general shilling attack model is shown in Fig. 1 [?]. It consists of four sets of items. I_S, set of selected items, determines the characteristics of the attack. They are determined using a rating function δ. I_F is selected randomly with a rating function σ to obstruct detection of an attack. A unique item i_t is targeted with a rating function, γ, to form a bias on. Remaining items are left unrated indicated as I_ϕ.

Although there are various attack models, we focus on specific shilling attacks. They can be briefly described as follows [?]. **Segment attack** is designed for a group of users who tend to be interested in the target object. The attacker wants her products to be recommended to related users. If an attacker whose products are related with horror movies, she wants to recommend her products to a group of users who like horror movies. The attacker rates the products the segment users will like as *maximum* rating value. She rates the other items as *minimum* rating value. She rates the target item as *maximum* rating value. In **bandwagon attack**, popular items are chosen as selected items and rated as *maximum* rating value in the ratings' interval so that attack profiles like other users much more. Filler items are chosen randomly and rated around the system's *mean*. The target item's rating is determined as *maximum* rating value. The set of filler items is chosen randomly and rated around the each item's *mean* in **average attack**. The target item's rating is determined as *maximum* or *minimum* value of the interval of system ratings for push and nuke attacks, respectively.

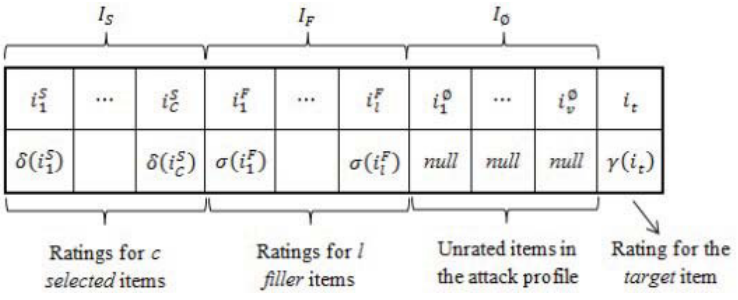

Fig. 1. General form an attack profile

4. Detecting Shilling Profiles via Bisecting *k*-means Clustering

Recommender systems aim to provide useful guidance for users about their online activities by providing personalized predictions. Such recommendations lead them to shop particular products, read tailored news, see specific movies, or listen to certain kinds of songs. Although these systems provide such referrals based on sensitive user preferences, they usually are not able to check authenticity of users in online platforms. Since they are open for public usage, it is straightforward for an attacker to leak into a recommender system database. Thus, they are subjected to manipulations of malicious users/parties that aim to mislead recommendations for authentic users in favor of particular target items. Such manipulations cause misguidance of users towards irrelevant products, which in turn results with time losses for unsatisfied customers and financial losses for online vendors. Hence, detecting and removing shilling attempts is vital for CF systems to provide a pleasing service to their customers. In this section, we first describe how to apply bisecting *k*-means clustering to form a BDT before detection and introduce utilization of intra-cluster correlation as a detection attribute. Then, we explain a novel detection scheme for specific attacks on traversing produced BDT and explain utility of the proposed algorithm in detection process.

4.1. Building a BDT with intra-cluster correlation attribute

Bisecting *k*-means clustering-based CF system is proposed by ? to improve accuracy and scalability. In such scheme, the central server produces a BDT off-line by applying bisecting *k*-means clustering algorithm on collected user-item matrix prior to prediction estimation. Given the user-item matrix and an optimal number of neighbors (N) for the recommendation process as a stopping criterion, *k*-means clustering is applied to group users into two distinct clusters at each level recursively. Cluster centers are indexed at all internal nodes of BDT to be used as a forwarding tool for newcomers and such process is repeated until at most N users remain in leaf nodes. Finally, a tree is obtained with indexed cluster centers at internal nodes and clustered users at leaf nodes. Such tree is utilized as a binary decision tool at each level in order to forward a new user to their neighbors according to the similarity levels to cluster centers. The tree grows by the same methodology as if population of any leaf node exceeds the stopping criterion, then corresponding node is bisected, which provides an easily scalable and maintainable structure for the scheme.

In addition to indexed cluster centers at internal nodes, we propose to include a detection attribute, intra-cluster correlation, to each internal node, which defines how close are the clustered users to the cluster center, as explained through a pseudo-code layout in Algorithm 1. Bisecting *k*-means clustering procedure divides a given user-item matrix, $U_{n\times m}$, into two sub-clusters for each internal node recursively. According to Algorithm 1, intra-cluster correlation for each sub-cluster, $C_{1\times m}$, is calculated as the average of Pearson's correlation coefficient values between each member of the sub-cluster and the cluster center. In other words, we measure average of similarities of each member of the internal cluster to the corresponding cluster center to quantify how tight an internal node is. Degree of intra-cluster correlation and changes of such attribute over levels is used as a detection parameter in the proposed detection scheme. Indeed, since shilling profiles are generated with a certain strategy, it is expected that such profiles do not scatter much around the cluster center and resemble high intra-cluster correlation.

In order to form a BDT to be utilized in detection process with intra-cluster correlation values for internal nodes, the data holder follows the procedure described in Algorithm 2 ? . Given $U_{n\times m}$ and an optimal number of

Algorithm 1 Intra-cluster correlation calculation for a given sub-cluster

Require: Cluster center ($C_{1\times m}$) & Cluster members ($M_{k\times m}$)

function ICC(C, M) ▷ intra-cluster correlation calculation
  Initialize: $similarities_{1\times k} \leftarrow 0$ ▷ correlation level for each member
  $\mu_C \leftarrow$ MEAN(C) ▷ cluster center's mean
  $\sigma_C \leftarrow$ STD(C) ▷ cluster center's standard deviation
  Calculate similarities between each member and cluster center:
  for all u_i in M ($i \leftarrow 1$ to k) **do**
    $similarities(i) =$ PCC($M(i), C, \mu_C, \sigma_C$) ▷ Pearson's correlation coefficient calculation
  end for
  Return average of similarities:
  return MEAN($similarities$)
end function

neighbors or N for a CF system, resultant BDT is estimated via bisecting k-means clustering by dividing $U_{n\times m}$ into two distinct clusters at each level. Cluster centers and intra-cluster correlation values are indexed to be used as a forwarding tool and detection parameter for each corresponding level, respectively. If the number of rating profiles in any sub-cluster exceeds the stopping criterion, N, then bisecting procedures is repeated to divide those clusters using the same approach. Such procedure continues repeatedly until obtaining a BDT having indexed cluster centers and intra-cluster correlation values as branch nodes and having at most N profiles at leaf nodes.

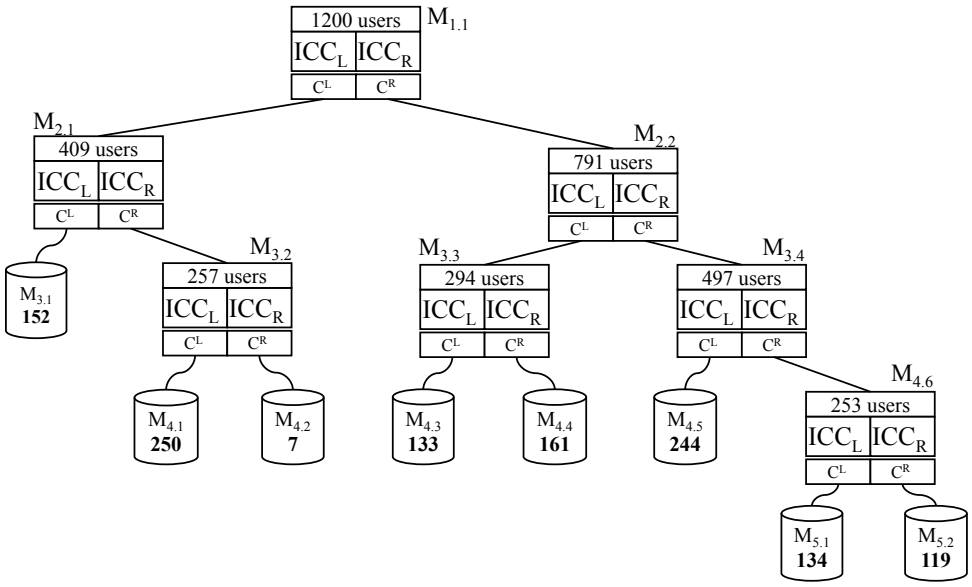

Fig. 2. An example binary decision tree

An example BDT produced via the proposed bisecting k-means clustering-based detection scheme can be seen from Fig. 2. The user-item matrix holds 1,200 users initially and stopping criterion N is determined to be 250 in this example for the sake of clarity. Members of root node are denoted with $M1.1$, where integer part indicates current height of BDT, which is one for the case of root, and fractional part represents index of sub-clusters at such level. Then, 1,200 users of root node are divided into two groups with 409 and 791 users at the first level indicated as $M_{2.1}$ and $M_{2.2}$, respectively. Corresponding intra-cluster correlation values are calculated and indexed for left and right sub-clusters, indicated as ICC_L and ICC_R, respectively. In addition, cluster centers are indexed at the root of the tree as C^L and C^R, where superscripts denote of which sub-tree the cluster center belongs to, either left or right. Similarly, in the left sub-tree of root, 409 users are clustered into two groups of 152 and 257 users,

denoted as $M_{3.1}$ as a leaf node and $M_{3.2}$ as a branch node, respectively. Corresponding intra-cluster correlation values and cluster centers are indexed at $M_{3.1}$ branch node. Going on so forth, each branch node is divided into two sub-clusters unless they contain N or fewer users. Finally, the BDT is completed with intra-cluster correlation values and two cluster centers at each branch nodes to facilitate detection of possible malicious attack attempts.

Algorithm 2 BDT formation via bisecting k-means clustering

Require: User-item matrix ($U_{n\times m}$) & Stopping criterion (N)

function BKM(U, N) ▷ bisecting k-means cluster
  Initialize: $IDX(n) \leftarrow 0$ & $BDT.centers_{2\times m} \leftarrow null$
  $BDT.left \leftarrow null$ ▷ pointer to left sub-tree
  $BDT.left.ICC \leftarrow null$ ▷ intra-cluster correlation for left sub-tree
  $BDT.right \leftarrow null$ ▷ pointer to right sub-tree
  $BDT.right.ICC \leftarrow null$ ▷ intra-cluster correlation for right sub-tree
  $BDT.LST$: a new BDT, $BDT.RST$: a new BDT ▷ left and right sub-trees
  Cluster: $[IDX, BDT.centers]$ = k-means(U, 2)
  for all u_i in F ($i \leftarrow 1$ to n) **do**
    if $IDX(u_i)$ = "left" **then**
      append u_i into $BDT.left$
    else
      append u_i into $BDT.right$
    end if
  end for
  $BDT.left.ICC \leftarrow$ ICC($BDT.centers(left)$)
  $BDT.right.ICC \leftarrow$ ICC($BDT.centers(right)$)
  if size($BDT.left$) > N **then**
    $BDT.LST$ = BKM($BDT.left, N$)
  end if
  if size($BDT.right$) > N **then**
    $BDT.RST$ = BKM($BDT.right, N$)
  end if
  return BDT
end function

4.2. Traversing BDT for detecting shilling profiles

Shilling attack profiles are usually created by a bot and follow a certain strategy according to the attack type in order to manipulate produced predictions, as described in Section 3. Since recommender systems operate based on quantified correlations among rating profiles over co-rated items, such strategy causes attack profiles to resemble more similarity to each other than genuine profiles. For example, while filler items of all average attack profiles are rated with such items' mean votes, bandwagon and segment attacks focus on specifically selected popular items according to the general community tastes or a set of particular users' favorites. Such parallel voting strategy increases similarity among attack profiles. In addition to the strategy of voting selected and/or filler items in attack profiles, such attacks aim to manipulate a particular product's prediction by rating it with the maximum possible vote. Therefore, rating for the target item is both the same and marginal in all attack profiles, which further intensifies correlation among malicious profiles.

Relying on the fact that shilling profiles demonstrate high similarity among themselves, we can infer that the correlation-based clustering algorithms are heuristically more prone to group shilling profiles together. Such discrimination can be observed sharper when number of clusters is reduced. Hence, we propose a bisecting clustering approach to detect malicious profiles. Although bisecting clustering approach is able to filter out shilling profiles from genuine ones over levels of clustering, a detection mechanism is needed to locate where such profiles reside in a BDT. For this purpose, we introduce intra-cluster correlation values of nodes in Section 4.1. In this

section, we explain how we utilize intra-cluster correlation attribute for detecting shilling profiles in recommender system databases.

Algorithm 3 Traverse BDT for detection

Require: Binary decision tree (BDT) & Upper and lower limit specifier (ρ)

1: **function** DETECT(BDT, ρ) ▷ determine predominant cluster holding *all* or *most* of shilling profiles
Start forwarding from root node:
2: **if** $BDT.leftElements.ICC > BDT.rightElements.ICC$ **then**
3: $parentTree = BDT.leftElements$
4: $BDT = BDT.left$
5: **else**
6: $parentTree = BDT.rightElements$
7: $BDT = BDT.right$
8: **end if**
Continue traversing according to changes in ICC:
9: **while** ($BDT.leftElements.ICC$ **AND** $BDT.rightElements.ICC$ is not in range $[parentTree.ICC \mp parentTree.ICC \times \rho\%]$) **do**
10: **if** $BDT.leftElements.ICC > BDT.rightElements.ICC$ **then**
11: $parentTreeElements = BDT.leftElements$
12: **if** BDT has a left child **then**
13: $BDT = BDT.left$
14: **else** *break*
15: **end if**
16: **else**
17: $parentTreeElements = BDT.rightElements$
18: **if** BDT has a right child **then**
19: $BDT = BDT.left$
20: **else** *break*
21: **end if**
22: **end if**
23: **end while**
24: **return** $parentTreeElements$
25: **end function**

After producing the BDT as described in Section 4.1, our proposed detection scheme follows a traversal procedure to determine a node (branch or leaf) that is most likely to hold *all* or *most* of inserted fake profiles. A detailed procedure of the proposed scheme is outlined as a pseudo code layout in Algorithm 3. The algorithm relies on two key points to continue. The first key point describes how to flow along the BDT to keep track of gathered attack profiles and the second one defines where to stop traversing and label current node as the target node holding shilling profiles. The algorithm starts traversing from root node and descends one level at a time until a stopping condition occurs; and finally returns the node where it breaks to be treated as holder of shilling profiles. Initialization procedure selects either left or right child of the root node according to intra-cluster correlation values, where the higher the correlation, the more likely for the node to hold attack profiles. Thus, the goal of the first key point while traversing the BDT is to be directed towards higher intra-cluster correlation values because huddled attack profiles heuristically increase correlation in clusters. Then the algorithm continues in a while loop, choosing one of the left or right children at each time and checking for the stopping condition to be occurred. The main anchor point of the stopping condition is that while the intra-cluster correlation values of shilled clusters are higher than the ones with genuine profiles, when the shilled clusters consisting totally or mainly of attack profiles are divided into two clusters, intra-cluster correlation of two children nodes cannot have diversely different intra-cluster correlation than their parent node. Such phenomenon is the signal to stop traversing and label parent node as the target node of attack profiles. For this purpose, Algorithm 3 checks if intra-cluster correlation of both children is lower or upper than a factor of parent node's intra-cluster correlation. If not, the

algorithm continues traversing by selecting the child node with higher intra-cluster correlation. However, if both children's intra-cluster correlation lie in the boundaries of change factor, then it can be concluded that the parent node holds attack profiles. When it is clustered, the attack profiles distribute to left and right children in such a way so that intra-cluster correlation of children do not change significantly and reside in change factor boundaries for both children. Note that the range of such change is determined by ρ parameter in Algorithm 3 and its optimal value can be determined experimentally.

The proposed detection scheme by traversal of a BDT with intra-cluster correlation values is expected to be more effective with attacks having strong and specific characteristics rather than random production of attack profiles. Therefore, we claim that the proposed scheme is more powerful at detecting shilling strategies like bandwagon, segment, and average attacks rather than random attack. Another interesting point about the detection procedure is that if the recommender system is shilling-free, or in other words, no shilling profiles are inserted into the system; it is very unlikely for the detection scheme to return false positive results, i.e., detecting a mass of genuine profiles as malicious. That is because if the system is shilling-free, then genuine profiles definitely demonstrate a diversity among themselves so that no two children of a parent node could have intra-cluster correlation values within a narrow range. Such intuitive phenomenon should be verified by experimental procedures.

5. Experiments

We conducted various experiments using a benchmark data set to probe how our method performs. After producing the BDT via bisecting k-means clustering, we evaluated the effectiveness of the proposed scheme. We used publicly available data set MovieLens Data (MLP). It contains 100,000 ratings in a 5-star rating scale for 1,682 movies. The ratings are collected from 943 users. Each user has rated at least 20 movies. We chose *precision* and *recall* as evaluation metrics. Let **A** be the number of attack profiles that classified correctly as fake, **B** be the number of authentic profiles that misclassified, and **C** be the number of attack profiles that is not correctly identified. Then, *Precision (P)* = **A** / (**A**+**B**) and *Recall (R)* = **A** / (**A**+**C**). We also utilized *F-Measure* as a combination of *precision* and *recall* as follows: $F1 = 2PR/(P + R)$.

We first performed experiments to show how varying values of ρ affect overall performance of the proposed scheme. We set filler size to 25 while we varied attack size from 5 to 25. We also changed the values of ρ from 1 to 10. For each attack size, we conducted our experiments 100 times while varying ρ values. We finally computed the overall averages of precision and recall values for all attack size values and displayed them in Table 1.

Table 1. Effects of varying ρ values on overall performance

	Precision					**Recall**				
ρ	1	2	4	7	10	1	2	4	7	10
Segment	0.955	0.933	0.875	0.850	0.863	0.950	0.955	0.965	0.952	0.967
Bandwagon	0.574	0.577	0.521	0.469	0.396	0.371	0.572	0.815	0.942	0.988
Average	0.746	0.743	0.749	0.751	0.701	0.622	0.619	0.623	0.628	0.638

In Table 1, we showed how our scheme performs with varying ρ values. In terms of precision, ρ values of 1, 2, and 7 provide the best outcomes for segment, bandwagon, and average attacks, respectively. Similarly, in terms of recall, ρ values of 10, 10, and 10 provide the best outcomes for segment, bandwagon, and average attacks, respectively. In order to find out the joint effects of such measures, we also computed *F1-measure* values for each attack. The values of 1, 4, and 7 output the best results with respect to *F1-measure* for segment, bandwagon, and average attacks, respectively. Thus, we selected them as optimum values and utilized them in the following trials.

After determining the optimum values of ρ for each attack, we performed experiments to show how varying filler size values affect the detection ability of our proposed method. In these trials, we set attack size to 25. We conducted our experiments while varying filler size from 3 to 25. After running our trials 100 times, we computed the overall averages of precision and recall. We finally displayed our outcomes in Table 2.

The outcomes in Table 2 show that our method is very successful for detecting specific attacks like segment, bandwagon, and average attacks. For average attack, detection ability of our method increases with augmenting filler size values. On the other hand, performance of our scheme slightly becomes worse with increasing filler

Table 2. Effects of varying $fillersize$ values on overall performance

	Precision					**Recall**				
Fillersize	3	5	10	15	25	3	5	10	15	25
Segment	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.999	0.984
Bandwagon	0.904	0.897	0.929	0.984	0.985	0.922	0.914	0.945	1.000	0.999
Average	0.521	0.916	0.992	0.990	0.988	0.498	0.800	0.826	0.877	0.949

size values for segment attack. For bandwagon attack, the best outcomes are observed when filler size is 15. The proposed method, in general, provides promising outcomes with respect to both precision and recall for almost all filler size values. To give an overall picture about the detection ability of our method, we calculated *F1*-measure values for each attack. Filler size values of 15, 15, and 25 provide the best outcomes for segment, bandwagon, and average attacks, respectively. Note that we fixed attack size at 25 and varied filler size values in these trials. Although our method performs well for larger filler and attack size values, it still provides promising results for smaller filler and attack size values especially for segment and bandwagon attacks.

We finally conducted experiments to demonstrate how our method performs with varying attack size values. We set filler size at 25 in these trials and varied attack size values from 3 to 25. Notice that we utilized the optimum ρ values. We again performed our experiments 100 times. After calculating overall averages, we displayed the final values of precision and recall in Table 3.

Table 3. Effects of varying *attacksize* values on overall performance

	Precision					**Recall**				
Fillersize	3	5	10	15	25	3	5	10	15	25
Segment	0.622	0.980	0.854	1.000	1.000	0.620	0.984	0.853	1.000	0.982
Bandwagon	0.053	0.085	0.070	0.947	0.985	0.970	0.973	0.352	0.987	0.999
Average	0.161	0.898	0.964	0.982	0.988	0.127	0.765	0.873	0.916	0.951

As seen from Table 3, overall performance of our scheme usually becomes better with increasing attack size values. In terms of precision, our method performs the best for detecting segment attack. It is also able to detect average and bandwagon attacks with very high success rates. The method is very successful for detecting all three attacks with respect to recall. Although our method seems for smaller attack size values especially for bandwagon attack, it is very successful for larger attack size values for all attacks. Also note that in order to affect the overall performance of any CF system, larger attack and filler size values should be utilized. In order to provide an overall picture, we again computed *F1*-measure values. The best results are observed for attack size values of 15, 25, and 25 for segment, bandwagon, and average attacks, respectively. In the literature, PLSA-based clustering is used as a detection scheme [?]. According to empirical results in [?], when filler size is 25, precision and recall values are about 0.80 for PLSA-based scheme for detecting average attack. Our method performs better than their scheme in terms of average attack. Our method is also very successful at detecting other attacks. When attack size is 15 and filler size is 25, *F1*-measure is 1.000 and 0.967 for segment and bandwagon attacks, respectively.

6. Conclusions and Future Work

Collaborative filtering schemes should provide accurate predictions efficiently. Since they might be subjected to shilling attacks, they also should utilize profile injection attack detection methods. Otherwise, they will not be able to provide accurate and dependable predictions. Clustering is a powerful tool used for different purposes. Due to its ability, there are different shilling attack detection proposals based on clustering. In this work, we also utilized the idea of clustering to filter out malicious profiles in a recommender system database. To the best of our knowledge, our work is the first one that uses the idea of bisecting k-means clustering as a shilling attack detection method. The clustering scheme recursively groups users into two sub-groups until the stopping criterion. Due to their high resemblance, it is most likely to have all or most of the attack profiles in one cluster. Our real data-based empirical outcomes show that our method is very successful at detecting bogus profiles generated from

specific attack models like segment, bandwagon, and even average attacks. There are three main factors (detection parameter, filler size, and attack size) that might affect the performance of our method. Therefore, we conducted various experiments to evaluate their effects on performance. With increasing filler and attack size, detection ability of our proposed method usually enhances. In general, our method provides promising results for almost all filler and attack size values. In addition to segment, bandwagon, and average attacks, there are other profile injection attacks like random, reverse bandwagon, love/hate attacks and so on. We will investigate whether our proposed method is able to successfully detect such attacks or not. Also, we want to further improve the overall success of our scheme by combining some existing shilling detection methods. And finally, we will utilize our method in some privacy-preserving environments.

Acknowledgements

This work is partially supported by the Grant 111E218 from TUBITAK.