From similarity perspective: a robust collaborative filtering approach for service recommendations

Min GAO (✉)[1,2], Bin LING[3], Linda YANG[3], Junhao WEN[1,2], Qingyu XIONG[1,2], Shun LI[1,2]

1 Key Laboratory of Dependable Service Computing in Cyber Physical Society (Chongqing University), Ministry of Education, Chongqing 400044, China

2 School of Software Engineering, Chongqing University, Chongqing 400044, China

3 School of Engineering, University of Portsmouth, Portsmouth PO1 3AH, UK

Abstract Collaborative filtering (CF) is a technique commonly used for personalized recommendation and Web service quality-of-service (QoS) prediction. However, CF is vulnerable to shilling attackers who inject fake user profiles into the system. In this paper, we first present the shilling attack problem on CF-based QoS recommender systems for Web services. Then, a robust CF recommendation approach is proposed from a user similarity perspective to enhance the resistance of the recommender systems to the shilling attack. In the approach, the generally used similarity measures are analyzed, and the *DegSim* (the degree of similarities with top *k* neighbors) with those measures is selected for grouping and weighting the users. Then, the weights are used to calculate the service similarities/differences and predictions. We analyzed and evaluated our algorithms using WS-DREAM and Movielens datasets. The experimental results demonstrate that shilling attacks influence the prediction of QoS values, and our proposed features and algorithms achieve a higher degree of robustness against shilling attacks than the typical CF algorithms.

Keywords collaborative filtering, service recommendation, system robustness, shilling attack

1 Introduction

Personalized recommender systems have been widely used to address information overload, including that in applications such as service selection, service discovery, and cloud deployment [1]. Collaborative filtering (CF) is a technique commonly used for recommendation [2–4]. CF recommender systems, whether user-based or item-based CF, are vulnerable to shilling attacks [5–8].

Shilling attacks are also a potential problem for QoS-based Web service recommender systems. Shilling attackers inject malicious performance values for Web services to recommender systems to manipulate Web service recommendations. In these systems, a higher QoS value of a Web service implies a higher recommendation probability. The most commonly used QoS features in the systems are response time and throughput capacity [3]. Similar to predicting ratings in movie and ecommerce recommender systems, the main task of the service recommender systems is to predict the QoS values such as response time and throughput values according to historical data of service invocations by users. Therefore, the CF-based Web service recommender systems are also susceptible to shilling attackers who inject fake users.

Shilling attack detection and robust attack-resistant CF have attracted significant attention in recent years. For detection, supervised learning-based detection systems using several user features [9–11], unsupervised detection based on clustering algorithms [12,13], and semi-supervised detection using both labeled and unlabeled data [14,15] are available. For robust attack-resistant CF, trust-aware CF based on constructing a user trust model [16,17] and item anomaly detection-based robust CF [18] are available. However, recent research on the shilling attacks in service recommender systems is inadequate.

The contributions of this paper are threefold: 1) The shilling attack problem in QoS-based Web service recommendation is presented; 2) a robust CF approach is proposed from the similarity perspective (None of the similarity irrelevant features are used in our approach); 3) random shilling attacks are injected to recommender systems to demonstrate how the predictions of typical item-based CF algorithms are influenced by attacks; then, these influences are compared with those on our proposed algorithms.

In this paper, we present an approach that utilizes the distributions of user similarities such as interest similarity and QoS similarity to determine the relative weights of the users, thereby distinguishing fake users from genuine ones. Our proposed approach follows three key steps: (1) The characteristics of the four most commonly used similarity measures are analyzed. (2) Features are extracted from the *DegSim* (Degree of similarities with top k neighbors) with those measures. (3) Users' weights calculated by a clustering algorithm with the features are combined to typical recommendation algorithms to predict the QoS values.

This paper is organized as follows. Section 2 presents shilling attack problems in CF-based Web service recommendations. In Section 3, we propose a robust CF approach for QoS-based service recommendations. Section 4 reports the experiments and results on WS-DREAM (Distributed Reliability Assessment Mechanism for Web Services) and Movielens datasets to compare between the robustness of typical item-based CF algorithms and our proposed algorithms. In Section 5, we analyze the experimental results, the general form of shilling attacks in service recommendation, and related work. Finally, we conclude this paper in Section 6.

2 Identifying shilling attack problem in CF-based Web service recommendation

In a typical CF for QoS-based service recommendation scenario [1,3], there is one or more $m \times n$ matrices, which includes a list of m users (IP addresses), a list of n services, and numerous QoS values, e.g., throughout. A QoS value $q_{u,s}$ implies a type of performance when the user u invokes the service s. The key step of CF for QoS-based service recommendation is to extrapolate the unknown QoS values for different users.

2.1 Shilling attacks on CF-based service recommendation

A typical attack on a recommender system is to arrange for a group of users to enter the system and vouch for certain items. These users become shills, also called fake users [6]. Therefore, the attacks are called shilling attacks (or profile injection attacks [19]).

User-based CF for service recommendation makes predictions by identifying peers with preference profiles; item-based CF for QoS-based service recommendation looks for services with similar profiles and makes predictions based on peer services' QoS values. It is feasible to identify which services have better QoS values in the invocations by the target segments of users; therefore, both user-based and item-based CF for service recommendation are affected by the attacks.

In this study, we focus on the random attack model to demonstrate the shilling attack problem because it is a low knowledge type of attack an attack whose execution does not require much knowledge [6] and straightforwardly constructed and applied to manipulate the QoS predictions in Web service recommender systems.

The random attack model for service recommendation is designed with the following characteristics. There are three sets of services in this attack model: a set of randomly selected filler services (S^F), a set of target services ($S^{T)}$, and the set of the other services (S).

S^F: All the services in S^F are assigned to random values that are in line with a certain distribution.

S^T: All the services in S^T are assigned to the most optimum value in the QoS matrix. It is convenient to obtain the value owing to the openness of the system. For example, the most optimum value of the response time is the smallest value in the response time matrix.

S: All the other services in the service set S are signed as Null, i.e., these services are not assigned to any values.

When an attacker injects fake users, the attack size and filler size will be used to regulate the number of fake users and the number of QoS values for the fake users. Attack size is the percentage of fake users. Filler size is the percentage of the filled ratings or QoS values for a fake user.

A schematic of a shilling attack in a QoS-based Web service recommender system is shown in Fig. 1. There are two fake users (users with fake locations) who will affect the recommendation to user A. If there is no fake user in the system, the system will recommend *Service* 1 to user *A* because the service has a shorter response time. However, after an attacker injects the fake users, the system will recommend *Service* 2 to user *A*. The location of the fake users appears in the vicinity of that of user *A*; however, they are actually at *Location* 2. Through spurious IPs, the fake users are treated as users similar to user *A*, and fake response time values are utilized to generate the recommendation for user *A*.

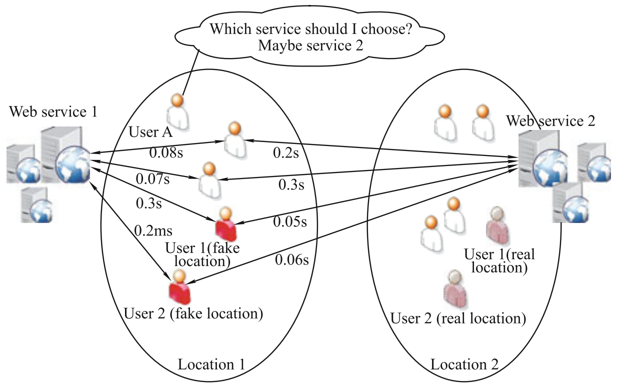


Fig. 1 Schematic of shilling attack in QoS-based Web service recommender system

2.2 An example: Shilling attack on CF-based service recommendation

The CF for QoS-based Web service recommender systems is susceptible to shilling attacks. For example, a platform collects the entire realty information available on the official websites of real estate organizations. It ranks and recommends services to users according to both the user demands and the Web services' QoS values. In the WS-DREAM Web Service QoS Dataset, suppose a user wishes to reside in Puget Sound and the corresponding recommendation list contains two Web services, WSID 5703 (see century21northhomes website) and WSID 5711 (see pjgoldhomes website) (Table 1).

Table 1 An example of attacks on a QoS-based Web service recommender system

Users	Response time/s	
	WS:5703	WS:5711
Average response time of all normal users (20 users)	**0.103**	**0.0939**
User 1: 128.119.41.210 (a fake user)	0.052	0.118
User 2: 128.111.52.64 (a fake user)	0.061	0.177
User 3: 128.112.139.80 (a fake user)	0.053	0.125
Average response time of all users (23 users)	**0.096**	**0.101**

Without any loss of generality, only the response time feature of the Web service is taken into consideration in the example. Suppose there were 20 invocations for each service, and the average response time value was calculated according to the response time values in those invocations.

In this case, Service 5711 should be recommended first because its average response time (0.0939s) is smaller than that of Service 5703 (0.103s).

As Table 1 depicts, after the attack (assume there are three fake users, whose IP addresses are 128.119.41.210, 128.111.52.64, and 128.112.139.80), the average response time of services 5703 and 5711 change from 0.103s and 0.939s to 0.096s and 0.100s, respectively.

Therefore, the shilling attack is effective because the response time of Web service 5703 becomes smaller than that of Web service 5711 after the attack.

The example can be generalized to all types of QoS attacks on CF Web service recommendation, such as random attack, average attack [6], and bandwagon attack [20].

Moreover, the IP address in an invocation could be manipulated. For example, as an important indicator of the QoS, the response time may be related to the distance from the user to the server of a Web service. Using a false IP, an attacker creates an illusion that an invocation at a distance from a service can obtain a short response time. Thus, the shilling attack would work effectively in a QoS-based Web service recommender system. In addition, the QoS values in a Web service recommender system are generally remarkably sparse. To solve the problem, a system can utilize the location-based Web service recommendation method [21], in which integrated QoS values of services in adjacent locations will be treated as the QoS value of the service for this region so that the illusion created by the attackers will severely affect the QoS of a service for the region and will result in biased recommendation.

3 A robust CF based on similarities for service recommendation

Mehta et al. [22] determined that fake users are highly correlated to each other, and he proposed an algorithm based on principal component analysis. Wang et al. [23] proposed an approach to eliminate maliciousness among the peers based on the neighbor similarity of peers in a group peer-to-peer (P2P) ecommerce network. These studies inspired us to propose an approach based on user similarities.

In order to provide robust service recommendation, four parts of work have been conducted (see Fig. 2).

Part 1: Analyze user similarity measures (Section 3.1).

Part 2: Extract features from the similarity perspective

(Section 3.2).

Part 3: The features are used to figure out fake users and reduce their weights; the weights are used with typical CF algorithms to calculate service similarities/deviations and predict QoS values (Section 3.3).

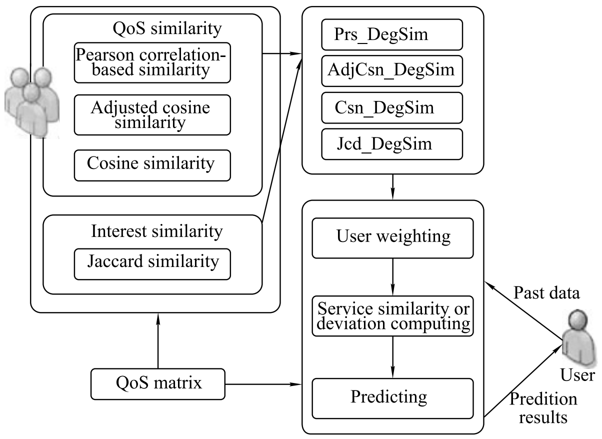


Fig. 2 Procedures of the proposed robust CF approach

The robustness of recommender systems implies the capability to make recommendations notwithstanding biased ratings [24].

3.1 User similarities

There are different types of similarities between users in a recommender system. In this section, we describe an approach for exploiting user similarities and features for detecting fake users from a similarity perspective. Two types of user similarities are investigated in particular: QoS similarity and interest similarity.

There have been certain commonly used similarity computing methods in recommender systems, such as Pearson correlation coefficient, adjusted cosine, cosine, and relevance-based methods [8,25–27]. The first three methods, measured by how two users' QoS values are similar or correlated to each other, can be considered as QoS similarities. They are more frequently used than the last one. The last one, also named interest similarity, is measured by the profile overlap [26], i.e., the number of services those have been invoked by two specified users, which represents the extent to which they share common interests in a specific set of services.

Pearson correlation coefficient, adjusted cosine similarity, and cosine-based similarity between the QoS values of users u and v are named $Psn\ (u, v)$, $AC\ (u, v)$, and $Csn\ (u, v)$, respectively.

$$Psn(u,v) = \frac{\sum_{i\in S(u)\cap S(v)}(q_{u,i}-\overline{q_u})(q_{v,i}-\overline{q_v})}{\sqrt{\sum_{i\in S(u)\cap S(v)}(q_{u,i}-\overline{q_u})^2}\sqrt{\sum_{i\in S(u)\cap S(v)}(q_{v,i}-\overline{q_v})^2}}, \tag{1}$$

$$AdjCsn(u,v) = \frac{\sum_{i\in S}(q_{u,i}-\overline{Q_u})(q_{v,i}-\overline{Q_v})}{\sqrt{\sum_{i\in S}(q_{u,i}-\overline{Q_u})^2}\sqrt{\sum_{i\in S}(q_{v,i}-\overline{Q_v})^2}}, \tag{2}$$

$$Csn(u,v) = \frac{\sum_{i\in S}(q_{u,i}\cdot q_{v,i})}{\sqrt{\sum_{i\in S}(q_{u,i})^2}\sqrt{\sum_{i\in S}(q_{v,i})^2}}, \tag{3}$$

where $\overline{q_u}$ is the average of the u's QoS values on the services in $S(u)\cap S(v)$, that is, $\overline{q_u} = \sum_{i\in S(u)\cap S(v)} q_{u,i}/|S(u)\cap S(v)|$; $\overline{q}_v = \sum_{i\in S(u)\cap S(v)} q_{v,j}/|S(u)\cap S(v)|$; $\overline{Q}_u = \sum_{i\in S(u)} r_{u,i}/|S(u)|$ is the average of all known QoS values of user u; $\overline{Q}_v = \sum_{i\in S(v)} q_{v,i}/|S(v)|$; $|S(u)|\cap|S(v)|$ is the set of services invoked by both user u and user v.

The relevance-based similarity can be calculated by Jaccard, measured by the fraction of shared services in the services jointly invoked by both the users (see Formula 4). This similarity between u and $\underline{v}$ is named $Jcd\ (u, v)$.

$$Jcd(u,v) = \begin{cases} \frac{|S(u)\cap S(v)|}{|S(u)\cup S(v)|}, & \text{if } (u \neq v); \\ 1, & \text{if } (u = v). \end{cases} \tag{4}$$

3.2 Analysis on *DegSim* based on different similarities

To analyze the features of the similarities, we calculated the mean of the *DegSim* [11] using Pearson correlation coefficient, adjusted cosine similarity, cosine, and relevance-based similarity. This is because *DegSim* has been identified to be effective for fake users' detection [10,28]. In a typical *DegSim*, the k most similar neighbors are used to calculate the mean Pearson correlation coefficient similarity for each user (Eq. (5)). In this paper, the *DegSim* with Pearson correlation coefficient similarity is named Prs_*DegSim*.

$$Prs\_DegSim_v = \frac{\sum_{u=1}^{k} Psn(u,v)}{k}. \tag{5}$$

Prs_DegSim_v is the mean of the similarities of the k most similar users of v. $Psn(u, v)$ is the Pearson-correlation-coefficient-based similarity between u and v.

However, *Prs_DegSim* itself is not adequate to detect fake users. For example, on Movielens 100k Dataset, under random attacks, the *Prs_DegSim* values of normal and fake users are shown in Fig. 3.

In these attacks, the attack sizes are 1%, 3%, 5%, and 10% and the filler sizes are 1%, 3%, 5%, and 10%. The blue points are normal users and the red points are fake ones.

Normal and fake users cannot be classified with *Prs_DegSim* values, as shown in Fig. 3. This could be the reason why researchers use *Prs_DegSim* as well as RDMA or other features [22,28] to detect fake users.

To further explore how user similarities help detect shilling attacks, we incorporate other similarities in Subsection 3.1 into the *DegSim* calculation, namely, *AC_DegSim* (Eq. (6)), *Csn_DegSim* (Eq. (7)), and *Jcd_DegSim* (Eq. (8)).

$$AC\_DegSim_v = \frac{\sum_{u=1}^{k} AC(u,v)}{k}. \tag{6}$$

$$Csn\_DegSim_v = \frac{\sum_{u=1}^{k} Csn(u,v)}{k}. \tag{7}$$

$$Jcd\_DegSim_v = \frac{\sum_{u=1}^{k} Jcd(u,v)}{k}. \tag{8}$$

The formulas are measured by the mean value of the adjusted cosine, cosine, and Jaccard similarities of the k most similar users of v.

To show the corresponding values of the *DegSim*s, similar random shilling attacks are generated into Movielens 100K Dataset with 1%, 3%, 5%, and 10% attack sizes and 1%, 3%, 5%, and 10% filler sizes.

We did not use WS-DREAM dataset for *DegSim* analysis because it has all the QoS values in the matrix; however, in an actual recommender system, the QoS matrix is generally very sparse [1]. Movielens Dataset (10% density) derived from an actual movie recommender system is suitable for analyzing the *DegSim* values.

The values of *AC_DegSim* are similar to those of *Prs_DegSim*; however, those of *Csn_DegSim* and *Jcd_DegSim* are significantly different, as shown in Fig. 4. The normal and fake users can be classified more conveniently with *Csn_DegSim* or *Jcd_DegSim* than with *Prs_DegSim* or *AC_DegSim*.

Furthermore, the normal and fake users can be conveniently classified using (*Csn_DegSim*, *Jcd_DegSim*) points. This is shown in Fig. 5, where the blue points represent normal users and the red points represent fake ones, for random attacks with 5% attack size and 5% filler size. The results with other attack and filler sizes are similar to those in Figs. 4 and 5.

3.3 CF approach based on the features from similarity perspective

There are two steps in this subsection: 1) identify fake users based on the features extracted in Section 3.2, and 2) deactivate the users in CF algorithms.

3.3.1 Identification of fake users

According to the results in Section 3.2, the points (*Jcd_DegSim*, *Csn_DegSim*) of fake users are at a distance from the expectation of the distribution. Moreover, the similarities between fake users are higher than those between normal users, which will be discussed in detail in Section 5.3. Thus, fake users can be clustered into a group using the (*Jcd_DegSim*, *Csn_DegSim*) values.

If the users are in the attack group, their weights should be exceedingly low; otherwise, the weights will be high. As

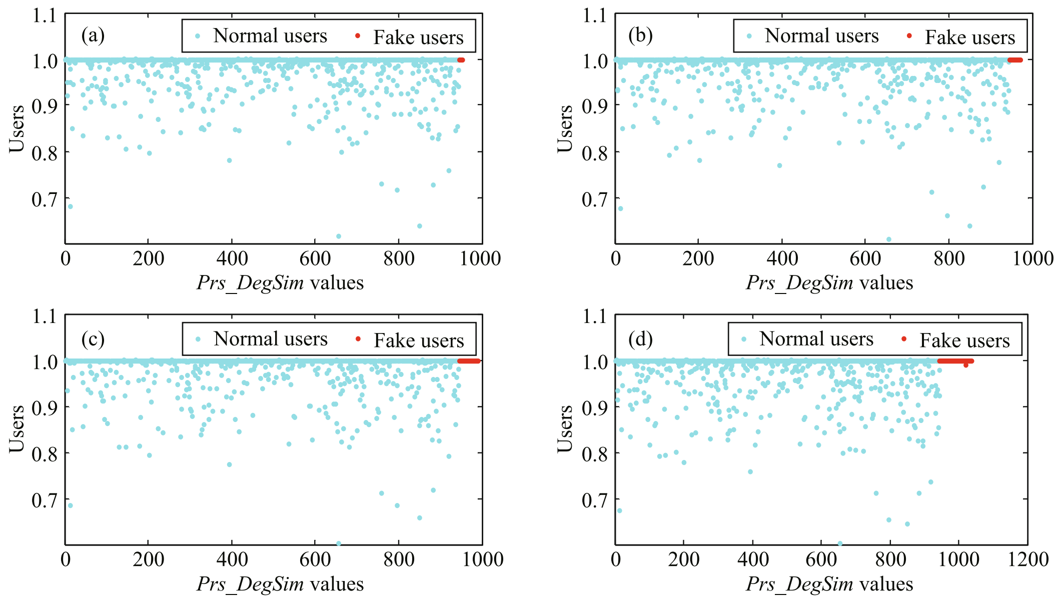


Fig. 3 *Prs_DegSim* values under attack. (a) Attack size=1% , filler size=1%; (b) attack size=3% , filler size=3%; (c) attack size=5% , filler size=5%; (d) attack size=10% , filler size=10%

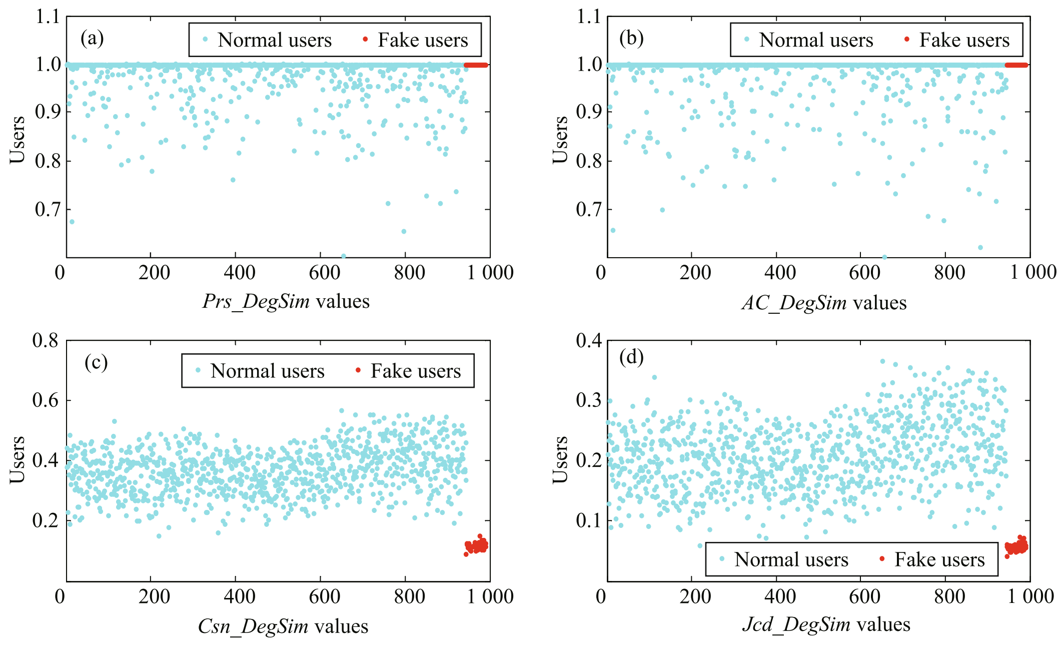


Fig. 4 *DegSim* values under attacks (attack size = 5%, filler size = 5%). (a) *Prs_DegSim*; (b) *AC_DegSim*; (c) *Csn_DegSim*; (d) *Jcd_DegSim*

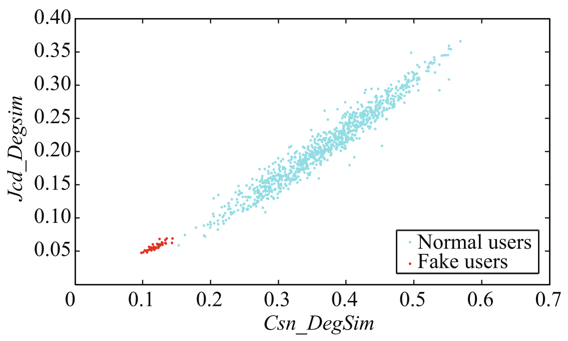


Fig. 5 *Csn_DegSim* and *Jcd_DegSim* under random attacks (attack size=5%, filler size=5%)

the weighting problem is also a clustering related problem, a particular type of algorithm can be used to group users, e.g., DBSCAN (Density-Based Spatial Clustering of Applications with Noise), OPTICS, and DENCLUE. In our approach, we use DBSCAN [29,30] to group dense users as an example. The pseudocode of the DBSCAN is shown as Algorithm 1.

The dataset D for the algorithm consists of users' (*Csn_DegSim*, *Jcd_DegSim*) values. A heuristic method [30] is used to determine the parameters *eps* and *minpts* dynamically. The Euclidean distance is used to calculate the *eps* neighborhood. Upon applying the DBSCAN algorithm, we determined the fake users group to be a cluster; however, it is challenging to cluster the normal ones in a group.

$$w_u = \begin{cases} 0, & \text{if } (u \in SU); \\ 1, & \text{if } (u \notin SU). \end{cases} \tag{9}$$

Then, we can identify the suspicious user group using the algorithms. The weight w_u of user u can be calculated by Eq. (9). SU is the set of suspicious users.

Algorithm 1 Density-based clustering algorithm

```
DBSCAN ($D$, $eps$, $minpts$)
{$Cls$ = Null
   for each unvisited point x in dataset D do
   { Mark $x$ as visited
      $\text{Neighbor}_{eps}x$ = all points within $x$'s $eps$ neighborhood
      if sizeof ($\text{Neighbor}_{eps}x$) $<$ $minpts$ then
            Make $x$ as Noise
      else
            {$Cls$ = next cluster
            EXPCluster ($x$, $\text{Neighbor}_{eps}x$, $Cls$, $eps$,$minpts$)
            }
   }
}
EXPCluster ($x$, $\text{Neighbor}_{eps}x$, $Cls$, $eps$,$minpts$)
{ add $x$ to cluster $Cls$
   for each y in $\text{Neighbor}_{eps}x$
   { if $y$ is not visited then
      {mark y as visited
      $\text{Neighbor}_{eps}y$ = all points within $y$'s $eps$ neighborhood
      if sizeof ($\text{Neighbor}_{eps}y$) >= $minpts$ then
         $\text{Neighbor}_{eps}x = \text{Neighbor}_{eps}x \cup \text{Neighbor}_{eps}y$
      if $y$ is not yet member of any cluster then
         add y to cluster $Cls$
      }
   }
}
```

3.3.2 Combine user weights in typical CF algorithms

Item-based CF [8] was proposed to compute the similarities between items and then to select the most similar items for prediction. It functions by comparing items based on the pattern of ratings across users. Adjusted cosine [8] and SlopeOne [31] are commonly used algorithms to calculate the similarities (or differentials) between items and to make predictions because they are reasonably accurate and conveniently analyzed [32].

Thus, in this study, the users' weights are incorporated with adjusted cosine-based CF (ACCF) and SlopeOne, forming weighted ACCF (wACCF) and weighted SlopeOne (wSlopeOne).

1) wCCF algorithm

• Step 1 Service similarity computing

In wACCF, the service similarities are calculated by Eq. (10).

$$Sim_{i,j} = \frac{\sum_{u\in U(i)\cap U(j)}(q_{u,i}-\overline{q_u})(q_{v,i}-\overline{q_v})\times w_u^2}{\sqrt{\sum_{u\in U(i)\cap U(j)}(q_{u,i}-\overline{q_u})^2}\sqrt{\sum_{u\in U(i)\cap U(j)}(q_{v,i}-\overline{q_v})^2}}. \tag{10}$$

Here, $U(i)$ is the set of users who have rated on service i, $\overline{q_u}$ is the average of user u's QoS values, and w_u is the weight of user u.

• Step 2 QoS prediction

To predict a QoS value $p_{u,i}$, the weighted sum is applied by Eq. (11), which is the crucial step in CF algorithms.

$$P_{u,i} = \frac{\sum_{j\in S(u)}(Sim_{i,j}\times q_{u,j})}{\sum_{j\in S(u)} Sim_{i,j}}. \tag{11}$$

2) wSlopeOne algorithm

• Step 1 Service differential computing

For wSlopeOne, the differential of services i and j $diff_{i,j}$ is calculated by Eq. (12), which is the average difference between the QoS values of i and j:

$$diff_{i,j} = \frac{\sum_{u\in S(i)\cap S(j)}(q_{u,i}-q_{u,j})\times w_u}{|S(i)\cap S(j)|}, \tag{12}$$

where $|S(i)\cap S(j)|$ is the cardinality of the set.

• Step 2 QoS prediction

The differentials of services are then used to predict the QoS values (Eq. (13)).

$$p_{u,i} = \frac{\sum_{j\in S(u)}(q_{u,i}-duff_{i,j})}{|q_u|}. \tag{13}$$

3.4 Computational complexity analysis

We suppose there are m users and n Web services in a recommender system.

3.4.1 Complexity of wACCF and wSlopeOne

For a service, we need to calculate the service–service similarities or deviations with all the n services in the training set. The computational complexity of each similarity or deviation computation is $O(l)$; l is the number of intersecting users between the two services. The parameter l is ordinarily a small number because the QoS matrix is generally a sparse matrix. There are n services; therefore, the time complexity of service–service similarity or deviation computation is $O(l\times n^2) = O(n^2)$. After offline similarity/deviation computation, the time complexity of the prediction computation for each active user on each service is $O(k)$ because k services will be used to predict the values. The parameter k is the number of similar services and is generally a small number. In the prediction procedure, we need to predict at most n services for each user; therefore, the time complexity of the prediction computation for each user is $O(kn)$. There are m users in total; therefore, the time complexity of prediction computation is $O(kmn)$. The computational complexities of both wACCF and wSlopeOne are $O(n^2+mn)$.

3.4.2 Complexity of prediction for an active user

For the computational complexity of users' interesting similarities and rating similarities, the complexity is $O(m^2)$. For the DBSCAN algorithm, with the use of an accelerating index structure, the computational complexity is $O(m\log m)$; otherwise, the computational complexity is $O(m^2)$. As discussed in 3.4.1, the computational complexity of service–service similarities or deviations is $O(n^2)$. All these computations can be conducted offline. They require at most $O(m^2+n^2)$ memory.

For an active user, the computational complexity of the prediction of each value is $O(k)$. The parameter k is the number of similar services. Therefore, the computational complexities of the wACCF and wSlopeOne for an active user are $O(kn)$.

In the research, we intend to present that the shilling attacks are a threat to QoS-based Web service recommendations and provide an approach from the user similarity perspective. The computational costs should be duly considered if there are numerous users and Web services. A solution is to calculate the users' weights, similarities, and differentials offline and to predict the QoS values for active users online.

4 Experimental evaluation

This is to demonstrate that CF-based Web service recommendation is influenced by shilling attacks and to evaluate the robustness of the proposed algorithms. In the experiments, we used two datasets: WS-DREAM [1,3,33–35] and Movielens [25,36]. The main goal of WS-DREAM is to offer Web service QoS data for Web service researchers. The dataset is commonly used in QoS-based Web service recommendations [33–35]. Movielens is also used in the evaluation of Web service recommendations [37,38].

There are two service features in WS-DREAM dataset: response time and throughput. It contains all the QoS values of Web service invocations on 5,825 Web services by 339 service users. The response time matrix is utilized in the experiments. There are 1,974,675 response time QoS values (0–20) from 339 users on 5,825 Web services.

Movielens 100K dataset includes 100,000 ratings (1–5) from 943 users on 1,682 movies. Each user has at least 20 ratings.

To evaluate the algorithms based on the prediction of QoS values, the WS-DREAM dataset is generally preprocessed to have different densities [1,35]. That is because the QoS matrix is generally very sparse in an actual recommender system [1,3,35]. The processed matrix in this paper has 10% response time values randomly selected from 339 users on 1,000 Web services, and each user has at least 20 invocations on different services. It has a density similar to that of Movielens 100k dataset.

4.1 Metrics for evaluation

In the experiments, we use the mean absolute error (*MAE* [24]) and *predictionshift* [5,6] metrics to examine the precision and shift of predictions as well as the *PoU* (proportions of users influenced by attacks) to reveal the number of users' hit ratios in their top *n* recommendation lists that are influenced.

1) *Predictionshift* is for the deviation between the predictions before and after the attacks (Eq. (14)).

$$Predshift = \sum_{i\in I}\sum_{u\in U}\frac{|p_{u,i} - p_{u,i}|}{|U|\times|I|}, \tag{14}$$

where the $p_{u,i}$ and $p'_{u,i}$ are the predictions before and after the attacks, respectively.

2) *PoU* is the proportion of users whose hit ratio values are influenced under the attacks. The hit ratio can be calculated by the ratio of the services in the *m* recommendation list that are actually in the users' *n* most favorite services (in test set). *PoU* can be calculated by using Eqs. (15) and (16).

$$Pou_u = \begin{cases} 0, & \text{if } (H'_u = H_u); \\ 1, & \text{if } (H'_u \neq H_u). \end{cases} \tag{15}$$

$$PoU = \sum_{u\in U} PoU_u. \tag{16}$$

Here, H_u and H'_u are the hit ratio values of the users in the test set U before and after the attacks, respectively.

3) *MAE* is a metric for the precision of the predictions, which is measured by the deviation of the predictions [24] from the true ratings or QoS values.

The lower these metrics are, the more effective the approach is. Because the critical criterion for robust recommender systems is to reduce the influence of fake user profiles rather than to improve the precision of predictions, we only used *MAE* to demonstrate the accuracy of the predicted ratings or QoS values and did not use other measurements to evaluate the accuracy. Meanwhile, we adopted two essential measurements for robust recommendations: the shift of prediction and the proportions of users who are influenced by attacks.

4.2 Experimental methodology

For the selection of *k* neighbors, a large *k* would generate excessive noise for users with high correlations, while a small *k* would result in ineffective predictions for those with low correlations [39]. In the experiments, we select the value of *k* from 10, 20, and 30 to execute the ACCK, wACCK, SlopeOne, and wSlopeOne algorithms.

To test the robustness of the proposed algorithms, the attack model, attack size, and filler size were set as below:

- Attack model: random attack because it requires little knowledge [6,20];
- Attack size: the percentage of attack profiles, valued at 5% and 10%;
- Filler size: the percentage of the filler ratings (I^F) in the attacks, valued at 5% and 10%.

Attack size and filler size are measured as percentages of the pre-attack user count and of the service count, respectively. The sizes are set to 5% and 10% because these values are typical in shilling attacks [5,6,40].

According to the random attack model's description in Subsection 2.1, the settings of the fake users' profiles for the response time QoS dataset are as follows:

- S^F: the randomly filling Web services are randomly valued by its mean $\mu = 2.05$ and variance $\sigma^2 = 2.03$;
- S^T: the target services are assigned to q_{best}; in the experiments, 10, 20, 30, and 50 services are randomly selected as the target services;
- S: all other services are assigned to null.

Here, the values of the mean and variance are calculated using the training set of WS-DREAM dataset.

According to the random attack model's description [6] and the values in Movielens, the settings of the attack profiles for Movielens are as follows:

- I^F: the randomly filling items are assigned to random values with its mean at $\mu = 3.6$ and variance at $\sigma^2 = 1.1$;
- I^T: the target items are assigned to r_{max}; in the experiments, 10, 20, 30, and 50 items are randomly selected as the target items;
- I: all other items are assigned to null.

Here, the values of the mean and variance are calculated using the training set of Movielens. The numbers of target items are set to 10, 20, 30, and 50, because of which 10 and 20 are generally used [7], and we wish to determine if the prediction shift and hit shift will be influenced by the number of target items.

The experimental procedure includes the following steps:

Step 1 To obtain *Csn_Sim* and *Jcd_Sim* matrices of users.

Step 2 To calculate their (*Csn_DegSim*, *Jcd_DegSim*) values.

Step 3 To compute the users' weights using their (*Csn_DegSim*, *Jcd_DegSim*) values and DBSCAN.

Step 4 To predict ratings or QoS values in $U_i test$ using ACCF and wACCF algorithms; comparing the predicted ratings or QoS values with the actual values in $U_i test$ to obtain *MAE*, prediction shift, and *PoU*.

Step 5 To predict ratings or QoS values in $U_i test$ applying SlopeOne and wSlopeOne algorithms; calculating *MAE*, prediction shift, and *PoU*.

Step 6 To fill fake users' profiles into the rating matrix and respond-time QoS matrix with different attack sizes and filler sizes; then, repeat all the steps 50 times.

In the preprocessed WS-DREAM and Movielens test sets, numerous services and items have only a few QoS values or ratings. Numerous users have rated only a few service and items as well. Thus, for the metric *PoU*, when calculating the hit with the top n (from 5 to 40, interval is 5), we select only the users who invoke more than n services or rated more than n items.

4.3 The experimental results

4.3.1 Comparisons of prediction shift

To demonstrate how random shilling attacks influence the predictions of QoS and ratings and the stability of the proposed algorithms, the comparison results of prediction shift are shown in Figs. 6–8.

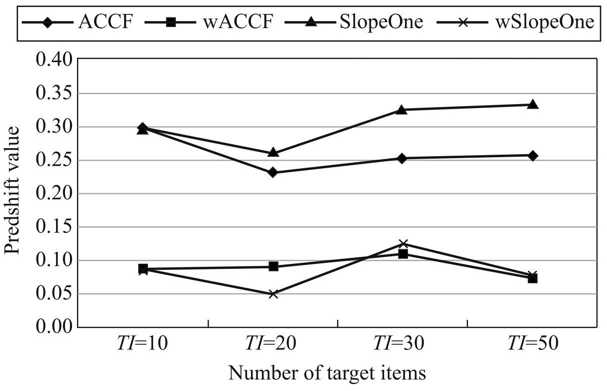


Fig. 6 Prediction shift comparison with different *TI* on Movielens

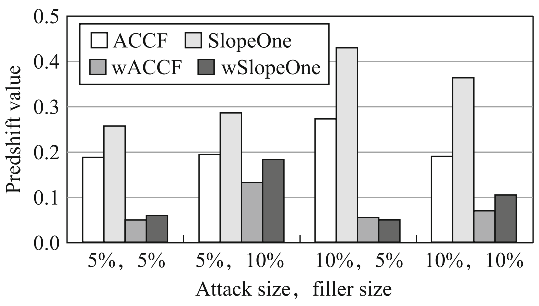


Fig. 7 Prediction shift comparison with different attack sizes and filler sizes on Movielens

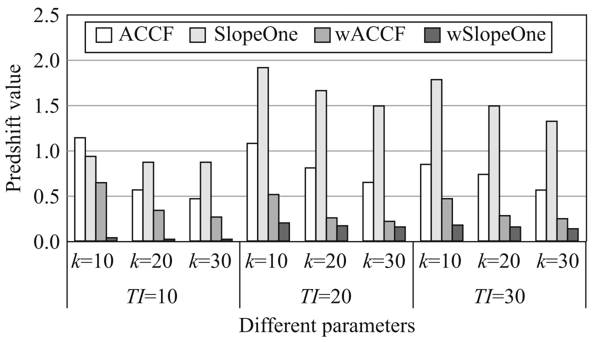


Fig. 8 Prediction shift comparison with different *TI* and number of neighbors (*k*) on WS-DREAM

The experimental results demonstrate that 1) compared with ACCF and SlopeOne, the predictions of wACCF and wSlopeOne vary negligibly under attacks with different numbers of target items; 2) the prediction shifts of SlopeOne are higher than those of ACCF, which indicates that SlopeOne is more vulnerable than ACCF on the prediction shift metric under random attacks, as shown in Fig. 6.

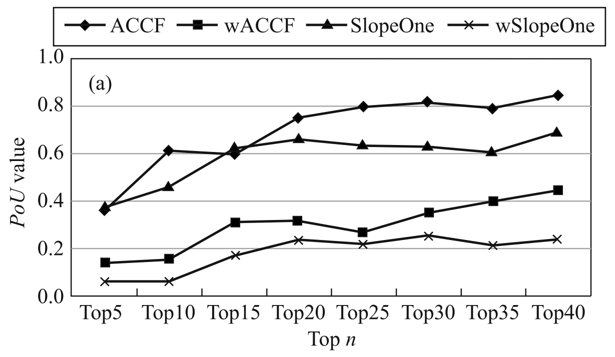
The prediction shifts when the system undergoes random attacks with different attack sizes and filler sizes are shown in Fig. 7. In these attacks, the number of target items is 20.

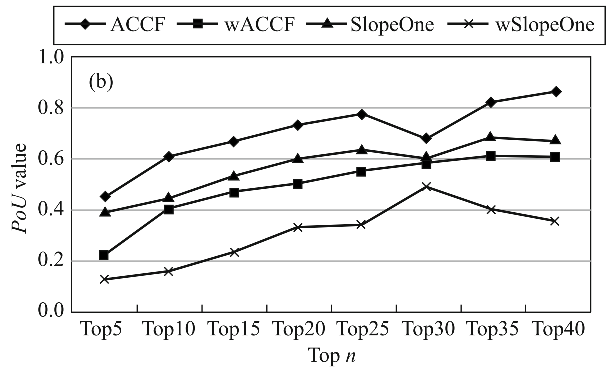
The experimental results demonstrate that 1) the prediction shift values of ACCF and SlopeOne are evidently higher than those of wACCF and wSlopeOne when the system suffers from attacks with different attack sizes and filler sizes; 2) In general, the prediction shift values of wACCF are less than those of wSlopeOne.

The experimental results with WS-DREAM are similar to those with Movielens, as shown in Fig. 8. The figure shows the prediction shift with different *TI* (*TI* = 10, 20, and 30) and different number of neighbors (*k* = 10, 20, and 30)

4.3.2 Comparisons of proportion of users influenced by attacks

The manner in which hit values of *PoU* are influenced by random attacks are shown in Figs. 9–11.





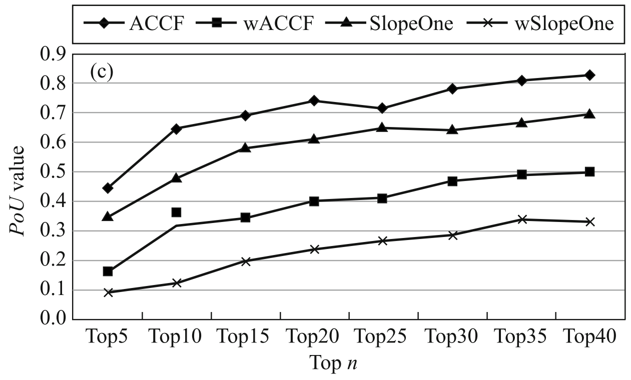


Fig. 9 *PoU*-value comparison with different *TI* and different Top *n*. (a) *PoU* values with *TI*=10; (b) *PoU* values with *TI*=30; (c) *PoU* values with *TI*=50

The *PoU* values under random attack with 10, 30, and 50 target items are presented in Fig. 10. The values of wACCF *and* wSlopeOne are apparently less than those of ACCF and SlopeOne, and the *PoU* values generally increase with *n* and TI.

The *PoU* values of ACCF are higher than those of SlopeOne, which indicates that ACCF is more vulnerable than SlopeOne on the *PoU* metric under random attacks.

The *PoU* values with different attack sizes and filler sizes on the Movielens 100k Dataset are presented in Fig. 10. The number of target items is 20 in the attacks. These *PoU* values are computed for the users' top 20 items. The figure shows that 1) the *PoU* values of wACCF and wSlopeOne are apparently less than those of ACCF and SlopeOne and 2) the *PoU* values of wSlopeOne are less than those of wACCF.

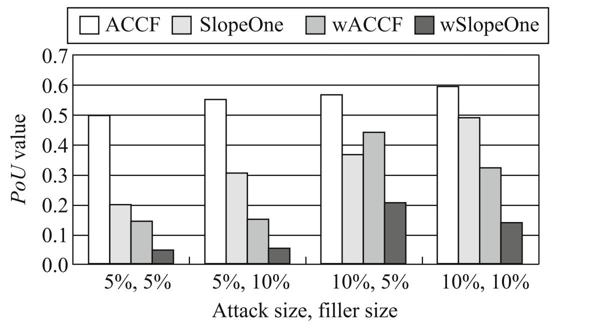


Fig. 10 *PoU* values influenced by the attacks with different attack sizes and filler sizes on Movielens

The *PoU* values on WS-DREAM Dataset are presented in Fig. 11. The number of target services is 30 in the attacks, and the attack size and filler size are both 10%. The figure shows that the *PoU* values of wACCF and wSlopeOne are apparently less than those of ACCF and SlopeOne; however, it is challenging to conclude which is more effective between wSlopeOne and wACCF, based on the results. The results with WS-DREAM are negligibly different from those with Movielens.

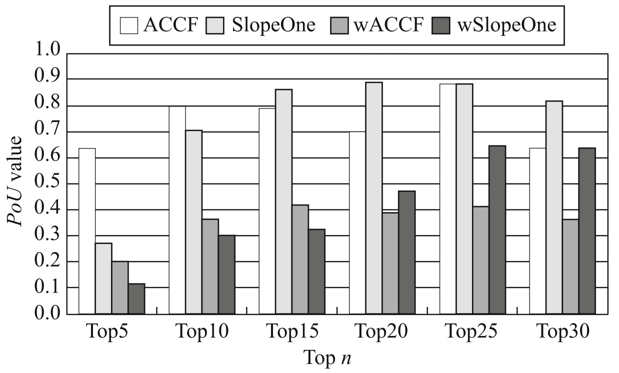


Fig. 11 *PoU* values with *TI* = 30 on WS-DREAM

4.3.3 Comparison of *MAE* values

The results of the *MAE* comparison between the WS-DREAM and Movielens dataset are presented in Table 2. The *MAE* values of ACCF and wACCF algorithms are approximately equal, while those of SlopeOne and wSlopeOne are marginally different. When the k in those kNN algorithms increases, those *MAE* values decrease. The lower the *MAE* value, the more effectively the algorithm predicts. The results are consistent with the research [39]: *MAE* decreases sharply as k varies from 10 to 30 with a step value of 10.

Table 2 *MAE* values with WS-DREAM (response time) and Movielens datasets

	K	ACCF	wACCF	SlopeOne	wSlopeOne
WS-DREAM	10	1.552	1.553	2.819	2.864
	20	1.387	1.388	2.209	2.249
	30	1.254	1.255	1.880	1.912
Movielens	10	0.739	0.737	1.061	1.049
	20	0.692	0.690	0.880	0.875
	30	0.672	0.671	0.788	0.782

As the intention of a robust recommender system is to reduce the influence of bogus ratings rather than to improve the precision of predictions, the shifts of the rating prediction and *PoU*, rather than *MAE*, are the essential measures. Our algorithms wACCF and wSlopeOne reduce the *predictionshift* and *PoU* values under random attacks with comparable *MAE* with ACCF and SlopeOne.

5 Analysis

5.1 Experimental analysis

As is apparent from the experimental results in Section 4, the robustness of the proposed algorithms is of a higher degree than that of typical algorithms with comparable *MAE* values.

Firstly, the proposed algorithms achieve stable QoS predictions for the system under random shilling attacks. Secondly, the proposed algorithms decrease the number/proportion of users influenced by the attacks. Thirdly, the accuracy of the proposed algorithms is comparable to those of typical CF algorithms. That is, the approach has the capability to make stable recommendations notwithstanding bias ratings injected with random attacks.

A likely reason for this is that the weights of the users are not taken into consideration in the baseline approaches. That is, the weights of the fake and normal users are similar.

For the prediction shift metric, 1) with the increase of the number of target items and services, the prediction shift values are generally increasing; 2) with the increase in the attack size and filler size, there is considerable variability in the prediction values of SlopeOne; however, those of ACCF are stable; 3) the prediction values of wSlopeOne are more or less higher than those of wACCF.

For the *PoU* values, 1) with the increase in the parameter n, the values of all the four algorithms more or less increase; 2) the values of ACCF are more or less higher than those of SlopeOne, and the values of wACCF are more or less higher than those of wSlopeOne; 3) with the increase in *TI*, the trend becomes increasingly apparent.

Therefore, for the prediction shift metric, wACCF and ACCF outperform wSlopeOne and SlopeOne, respectively; for the *PoU* values, wSlopeOne and SlopeOne outperform wACCF and ACCF, respectively.

A likely reason is that the similarities among the services are larger than the distances among the services in maintaining prediction stability, while the distances among the services are larger than the similarities among the services in maintaining the hit stability.

In this experiment, we discussed the algorithm only under random attack because it is inexpensive. For other types of shilling attacks, the proposed approach can also reduce their influence because the attacks can be considered as special forms of random attacks as they also give ratings to randomly selected services. The attacks are likely to enlarge the deviation of the prediction of the QoS values and increase the number of influenced users. However, they require higher knowledge of the items and recommendation approaches of the target systems. The cost of these types of attacks will be high.

5.2 General attack profile and attack models

In this study, the random attack problem has been analyzed, and the proposed approach can solve the problem. To specify the shilling attack problem in CF-based services recommendations, we present the general form of an attack profile based on the research [9], as presented in Table 3.

1) S^F is a set of randomly selected filler services $s_1^F \sim s_m^F$; there are m QoS values ($qs_1^F \sim qs_m^F$) on these services;

2) S is a set of uninvoked services ($s_1 \sim s_n$); there is no QoS value on these services;

3) S^S is a set of selected services ($s_1^S \sim s_l^S$), which exhibit certain relationships with the target services; there are l QoS values ($qs_1^S \sim qs_l^S$) on the services;

4) S^T is a set of target services ($s_1^T \sim s_p^T$); there are p QoS values ($qs_1^T \sim qs_p^T$) on the services.

Table 3 General form of an attack profile in CF-based service recommender system

	S^F	S	S^S	S^T
Attack profile	$s_1^F \sim s_m^F$	$s_1 \sim s_n$	$s_1^S \sim s_l^S$	$s_1^T \sim s_p^T$
Value	$qs_1^F \sim qs_m^F$	Null	$qs_1^S \sim qs_l^S$	$qs_1^T \sim qs_p^T$

Suppose that there are totally k services in a recommender system; an attack profile consists of a k-dimensional vector of QoS values, where $k = |S^F|+|S|+|S^S|+|S^T| = m+n+l+p$.

Different selection strategies for S^S and different values for the sets S^F and S^S form different attack models, such as random attack, average attack, bandwagon attack, and segment attack models [6,20].

In this study, we discussed the algorithm only under random attack because it is inexpensive. The proposed approach can reduce the influences of other types of shilling attacks also because the attacks can be considered as special forms of random attack as they too inject biased QoS values to random select services.

5.3 Feature analysis

In the study, the features from the similarity perspective (*Csn_DegSim* and *Jcd_DegSim*) are used to identify fake users injected by attackers with random attack and are further used in weighting the users. The experimental results in Section 4 demonstrate the robustness of the proposed algorithms based on the features that have been improved. To analyze why these features are beneficial to the approach, we observed their distributions. We observed that both *Csn_DegSim* and *Jcd_DegSim* are likely to belong to normal distributions, as is apparent from the histograms in Fig. 12.

To test if the samples of *Csn_DegSim* and *Jcd_DegSim* belong to normal distributions, the following hypotheses are to be tested by χ^2 test.

H_0: the data belong to a normal distribution.

H_1: the data do not belong to a normal distribution.

The following procedures were adopted to test *Jcd_DegSim*:

1) One hundred samples were randomly selected from genuine users;

2) Maximum Likelihood Estimation was adopted to estimate the mean μ and variance σ^2;

3) The values of *Jcd_DegSim* were partitioned into k disjoint subintervals;

4) Compared χ^2 statistic Q with the 1-α quantile of the χ^2 distribution with k-3 degrees-of-freedom to determine if H_0 should be accepted.

Here, we carried out the test at the level of significance $\alpha = 0.05$. The procedures adopted to test *Csn_DegSim* are similar.

The results of the χ^2 test demonstrate that the random samples of *Jcd_DegSim* and *Csn_DegSim* belong to normal distributions with means μ_1, μ_2 and variances σ_1^2, σ_2^2, respectively. The parameter for the *Jcd_DegSim* values was denoted as X

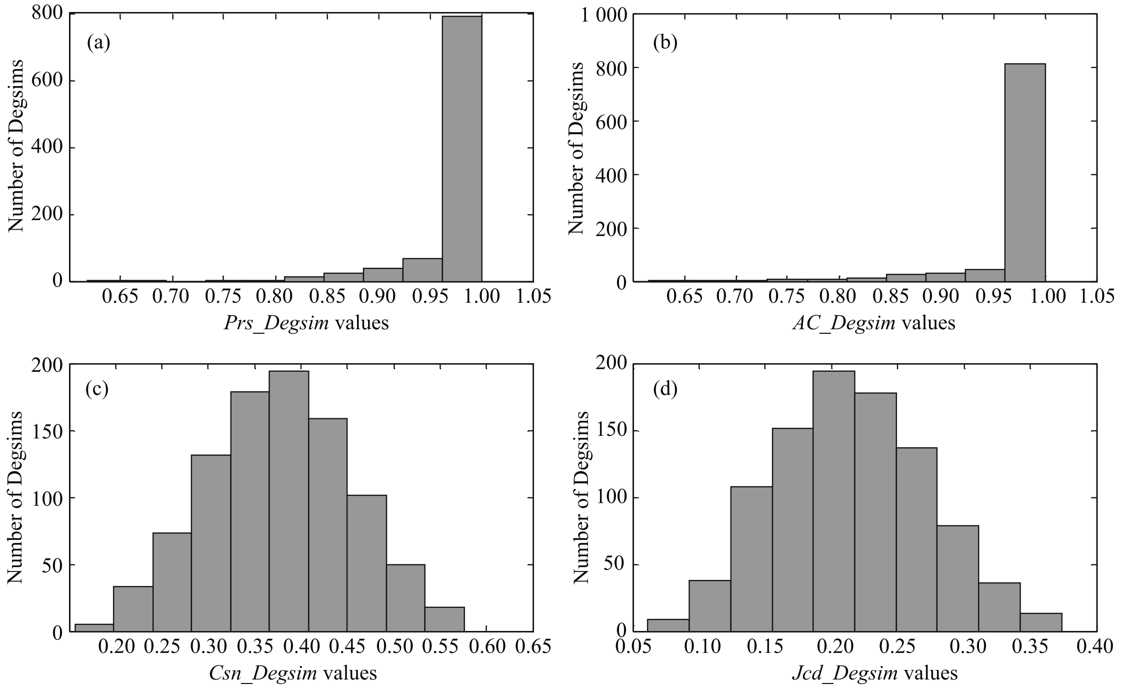


Fig. 12 Histogram of *DegSim* values. (a) *Prs_DegSim*; (b) *AC_DegSim*; (c) *Csn_DegSim*; (d) *Jcd_DegSim*

and that for the *Csn_DegSim* values was denoted as Y. By observing randomly selected samples of X and Y of fake users, we observed the features $|X-\mu_1| > 2\sigma_1$ and $|Y-\mu_2| > 2\sigma_2$. Because the probability P $\{|X-\mu| > 2\sigma\} < 0.0456$, the normal and fake users could be classified more conveniently with *Jcd_DegSim* or *Csn_DegSim*.

Furthermore, the point (*Jcd_DegSim*, *Csn_DegSim*) was noted as (X, Y). Set $Z = \sqrt{X^2 + Y^2}$. The result of the χ^2 test demonstrated that the distribution of Z could be accepted to be a normal distribution with mean μ_3 and variance σ_3^2. The Z values of fake users had the feature $|Z-\mu_3| > 3\sigma_3$. Because P $\{|Z-\mu_3| > 3\sigma_3\} >$ P$\{\mu_3-3\sigma_3 < Z <= \mu_3 + 3\sigma_3\} = 0.0026$, the points (*Jcd_DegSim*, *Csn_DegSim*) vary from the mean by three times the standard deviation and can be regarded as outliers.

Thus, from the similarity perspective, *Csn_DegSim* and *Jcd_DegSim* features are selected to detect fake users in this study.

5.4 Related work

A number of recent studies have focused on robust (or "trust-aware") CF. In the early stage, to improve the prediction accuracy of recommender systems, O'Donovan and Smyth [16] incorporate the trustworthiness of users into recommendation approaches. Moreover, a trust-aware CF approach based on "Web of trust" [41] is proposed to increase the coverage of recommendation while preserving the quality of predictions, particularly for new users. However, prediction accuracy and coverage are not essential metrics for robust recommender systems [32].

Subsequently, a few researchers [6,42,43] proposed trust-aware CF approaches based on specific user features or matrix factorization strategy. Despite the weak comparability to those related work, the experimental results with Movielens are provided in Section 4 for reference. The prediction shifts of the studies [20,44] are approximately in the range of 0.1–0.5; however, the shifts in this study are less than 0.1 in most cases.

New approaches have been proposed to detect shilling attacks in recent years, such as the fake user detection approaches via spectral clustering [13] and semi-supervised learning [15,16], and the item anomaly detection approaches using dynamic time interval segmentation technique [7,19]. Compared to these studies, our approach provides a perspective from similarity to solve the shilling attack problem and can be considered as a reference for constructing a supervised or semi-supervised classifier, or unsupervised clustering approaches in the detection of fake users for CF-based recommender systems.

Moreover, the shilling attack was built on the Sybil attack [45–49], which has been studied in the security literature. The researchers observed that the attacks demonstrate the capability to severely distort recommendation results [48,49]. To solve the problem, DSybil [48] and RobuRec [49] were proposed based on *sufficient information* and *overwhelming condition*. RobuRec is suitable for the recommenders with general scoring systems, whereas Dsybil is suitable only for binary feedback systems. In RobuRec, the item trust is calculated from a mean value of specified ratings on the item; then, the threshold values in both the upper and lower boundaries are set up to block those malicious ratings, which are outside the boundaries. DSybil and RobuRec are suitable for recommending products such as movies, songs, and books. For those products, the ratings depend on the users' subjective experience, and the mean value can be interpreted as the users' final agreement for each item. However, they are unsuitable for QoS-based Web service recommendation because the QoS values of Web services vary considerably depending on the network conditions rather than relying on the users' subjective experience. It is challenging to find the *users' agreement*, *sufficient information*, and *overwhelming condition* for each service.

Although the improvement of the proposed method is marginal compared to certain available anti-attack methods based on multi features of user profiles, the proposed method achieves high capability to resist only the shilling attacks in traditional service recommendation methods, based on user similarities. Furthermore, the statistical characteristics of new similarity measures demonstrate the capability to distinguish normal and fake users effectively. It provides new measures for other researchers to combine the measures to the most recent semi-supervised or active-learning-based anti-attack approaches.

6 Conclusion and future work

Generally, in collaborative recommender systems, biased profile data conveniently sway recommendations toward inaccurate results that serve the attacker's objectives. In this study, we have identified the shilling attack problem in QoS-based Web service recommendation and provided an example to demonstrate how fake users can effectively attack service recommender systems. *Csn_DegSim* and *Jcd_DegSim* features from the similarity perspective are used to detect fake user group and weight them. Next, two user-weight-based algorithms are proposed to calculate service similarities/deviations and predict ratings. The experimental results of this study demonstrate that 1) the predictions of QoS values are influenced under attacks, even under the most straightforward random attacks; 2) the most-used similarities between users are demonstrated to be effectively utilized to detect fake users; 3) against a random attack, the proposed algorithms achieve a higher degree of resistance than the typical item-based CF.

In this study, we conducted experiments only on the response-time QoS matrix. More experimental studies on other QoS properties (e.g., throughput) and more datasets will be conducted in a future work. We discussed only the algorithm under random attacks. The manner in which other types of attacks (e.g., average and bandwagon attacks) and obfuscated techniques distort QoS-based Web service recommendations will be studied. In the study on the proposed approach, we did not take time into consideration; however, time is an important factor because the distribution of users' similarities is time-varied, and the distribution of the QoS values of a service is also time-sensitive. We plan to conduct more studies to construct a time-aware approach to resist various types of shilling attacks.

Acknowledgements This research is supported by the Basic and Advanced Research Projects in Chongqing (cstc2015jcyjA40049), the National Natural Science Foundation of China (Grant No. 71102065), the Fundamental Research Funds for the Central Universities (106112014 CDJZR 095502), and the China Scholarship Council.

References

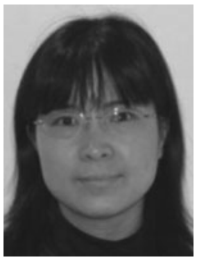
1. Zheng Z B, Ma H, Lyu M R, King I. Collaborative Web service qos prediction via neighborhood integrated matrix factorization. IEEE Transactions on Services Computing, 2013, 6(3): 289–299
2. Hernando A, Bobadilla J, Ortega F. A non negative matrix factorization for collaborative filtering recommender systems based on a Bayesian probabilistic model. Knowledge-Based Systems, 2016, 97: 188–202
3. Xu J L, Zheng Z B, Lyu M R. Web service personalized quality of service prediction via reputation-based matrix factorization. IEEE Transactions on Reliability, 2016, 65(1): 28–37
4. Jiang S H, Qian X M, Shen J L, Fu Y, Mei T. Author topic model-based collaborative filtering for personalized POI recommendations. IEEE Transactions on Multimedia, 2015, 17(6): 907–918
5. Mobasher B, Burke R, Sandvig J J. Model-based collaborative filtering as a defense against profile injection attacks. In: Proceedings of AAAI Conference on Artificial Intelligence. 2006, 1388–1393
6. Hurley N, Cheng Z P, Zhang M. Statistical attack detection. In: Proceedings of the 3rd ACM Conference on Recommender Systems. 2009, 149–156
7. Zhang S, Ouyang Y, Ford J, Makedon F. Analysis of a low-dimensional linear model under recommendation attacks. In: Proceedings of the 29th Annual ACM SIGIR Conference on Research and Development in Information Retrieval. 2006, 517–524
8. Sarwar B, Karypis G, Konstan J, Riedl J. Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on World Wide Web. 2001, 285–295
9. Mobasher B, Burke R, Bhaumik R, Williams C. Toward trustworthy recommender systems: an analysis of attack models and algorithm robustness. ACM Transactions on Internet Technology (TOIT), 2007, 7(4): 2301–2338
10. Zhou Q Q. Supervised approach for detecting average over popular items attack in collaborative recommender systems. IET Information Security, 2016, 10(3): 134–141
11. Chirita P A, Nejdl W, Zamfir C. Preventing shilling attacks in online recommender systems. In: Proceedings of the 7th Workshop on Web Information and Data Management. 2005, 67–74
12. Yang Z H, Cai Z, Guan X H. Estimating user behavior toward detecting anomalous ratings in rating systems. Knowledge-Based Systems, 2016, 111: 114–158
13. Zhang Z, Kulkarni S R. Detection of shilling attacks in recommender systems via spectral clustering. In: Proceedings of the 17th IEEE Conference on Information Fusion. 2014, 1–8
14. Cao J, Wu Z, Mao B, Zhang Y C. Shilling attack detection stilizing semi-supervised learning method for collaborative recommender system. World Wide Web, 2013, 16(5): 729–748
15. Wu Z, Wang Y Q, Wang Y Q, Wu J J, Cao J, Zhang L. Spammers detection from product reviews: a hybrid model. In: Proceedings of IEEE Conference on Data Mining. 2015, 1039–1044
16. O'Donovan J, Smyth B. Trust in recommender Systems. In: Proceedings of the 10th Conference on Intelligent User Interfaces. 2005, 39(4): 167–174
17. Moradi P, Ahmadian S. A reliability-based recommendation method to improve trust-aware recommender systems. Expert Systems with Applications, 2015, 42(21): 7386–7398
18. Xia H, Fang B, Gao M, Ma H, Tang Y Y, Wen J. A novel item anomaly detection approach against Shilling attacks in collaborative recommendation systems using the dynamic time interval segmentation technique. Information Sciences, 2015, 306: 150–165
19. Mobasher B, Burke R, Williams C, BhaumikR. Analysis and detection of segment-focused attacks against collaborative recommendation. In: Proceedings of Advances in Web Mining and Web Usage Analysis. 2006, 96–118
20. Mobasher B, Burke R, Bhaumik R, Williams C. Effective attack models for shilling item-based collaborative filtering systems. In: Proceedings of KDD Workshop on Web Mining and Web Usage Analysis. 2005, 21–28
21. Chen X, Zheng Z B, Yu Q, Lyu M. R. Web service recommendation via exploiting location and qos information. IEEE Transactions on Parallel and Distributed Systems, 2014, 25(7): 1913–1924
22. Mehta B, Hofmann T, Fankhauser P. Lies and propaganda: detecting spam users in collaborative filtering. In: Proceedings of the 12th ACM Conference on Intelligent User Interfaces. 2007, 14–21
23. Wang G J, Musau F, Guo S, Abdullahi M B. Neighbor similarity trust

against sybil attack in P2P e-commerce. IEEE Transactions on Parallel and Distributed Systems, 2015, 26(3): 824–833

24. O' Mahony M, Hurley N, Kushmerick N. Collaborative recommendation: a robustness analysis. ACM Transactions on Internet Technology, 2004, 4(4): 344–377
25. Shang M S, Lu L Y, Zeng W, Zhang Y C, Zhou T. Relevance is more significant than correlation: information filtering on sparse data. Europhysics Letters, 2009, 88(6): 68008
26. Ziegler C N, Golbeck J, Investigating interactions of trust and interest similarity. Decision Support Systems, 2007, 43(2): 460–475
27. Lee D H, Brusilovsky P. Social networks and interest similarity: the case of CiteULike. In: Proceedings of the 21st ACM Conference on Hypertext and Hypermedia. 2010, 151–156
28. Oh H K, Kim S W, Robust features for trustable aggregation of online ratings. In: Proceedings of the 10th ACM Conference on Ubiquitous Information Management and Communication. 2016, 13–19
29. Kriegel H P, Kroger P, Sander J, Zimek A. Density-based clustering. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2011, 1(3): 231–240
30. Ester M, Kriegel H P, Sander J, Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of KDD Workshop on Web Mining and Web Usage Analysis. 1996, 226–231
31. Lemire D, Maclachlan A. Slope one predictors for online rating-based collaborative filtering. In: Proceedings of SIAM Data Mining. 2005, 1–5
32. O' Mahony M P, Hurley N J, Silvestre G C. Promoting recommendations: an attack on collaborative filtering. Database and Expert Systems Applications. Springer Berlin Heidelberg, 2002, 2453: 494–503
33. Cao J, Wu Z A, Wang Y Q, Zhuang Y. Hybrid collaborative filtering algorithm for bidirectional Web service recommendation. Knowledge and Information Systems, 2013, 36(3): 607–627
34. Yao L, Sheng Q Z, Segev A, Yu J. Recommending Web services via combining collaborative filtering with content-based features. In: Proceedings of the 20th IEEE Conference on Web Services. 2013, 42–49
35. Zheng Z B, Ma H, Lyu M R, King l. QoS-aware Web service recommendation by collaborative filtering. IEEE Transaction on Services Computing, 2011, 4(2): 140–152
36. Jung J J, Attribute selection-based recommendation framework for short-head user group: an empirical study by movieLens and IMDB. Expert Systems with Applications, 2012, 39(4): 4049–4054
37. Rong W G, Liu K C, Liang L. Personalized Web service ranking via user group combining association rule. In: Proceedings of IEEE Conference on Services Computing. 2009, 445–452
38. Rong W G, Peng B L, Ouyang Y, Liu K C, Xiong Z. Collaborative personal profiling for Web service ranking and recommendation. Information Systems Frontiers, 2015, 17(6): 1265–1282
39. Chen X, Liu X D, Huang Z C, Sun H L. Region kNN: a scalable hybrid collaborative filtering algorithm for personalized Web service recommendation. In: Proceedings of IEEE Conference on Web Services. 2010, 9–16
40. Bryan K, O' Mahony M, Cunningham P. Unsupervised retrieval of attack profiles in collaborative recommender systems. In: Proceedings of ACM Conference on Recommender Systems. 2008, 155–162
41. Massa P, Avesani P. Trust-aware recommender systems. In: Proceedings of ACM Conference on Recommender Systems. 2007, 17–24
42. Zhang F G. Research on trust based collaborative filtering algorithm for user's multiple interests. Journal of Chinese Computer System, 2008, 29(8): 1415–1419
43. Mehta B, Nejdl W. Attack resistant collaborative filtering. In: Proceedings of the 31st ACM Conference on Research and Development in Information Retrieval. 2008, 75–82
44. Hurley N J, Robustness of recommender systems. In: Proceedings of the 5th ACM Conference on Recommender System. 2011, 9–10
45. Douceur J R. The sybil attack. In: Proceedings of International Workshop on Peer-to-peer Systems. 2002, 251–260
46. Karlof C, David W. Secure routing in wireless sensor networks: attacks and countermeasures. Ad Hoc Networks, 2003, 1(2): 293–315
47. Newsome J, Shi E, Song D, Perrig A. The sybil attack in sensor networks: analysis & defenses. In: Proceedings of the 3rd ACM International Symposium on Information Processing in Sensor Networks. 2004, 259–268
48. Yu H F, Shi C W, Kaminsky M, Gibbons P B, Xiao F. Dsybil: optimal sybil-resistance for recommendation systems. In: Proceedings of the 30th IEEE Symposium on Security and Privacy. 2009, 283–298
49. Noh G, Kang Y M, Oh H, Kim C K. Robust sybil attack defense with information level in online recommender systems. Expert Systems with Applications, 2014, 41(4): 1781–1791

Min Gao received her MS and PhD degrees in computer science from Chongqing University (CQU), China in 2005 and 2010, respectively. She is now an associate professor at the School of Software Engineering, CQU. She was a visiting researcher at the School of Business, University of Reading, UK. Her research interests include recommendation system, service computing, and data mining. She has published over 30 refereed journal and conference papers in these areas. She is a member of the IEEE and CCF.

Bin Ling received the BE, MS and MR. degrees from Staffordshire University, UK in 1995, 1998, and 1999, respectively. He was formerly a research fellow at School of Computer Science, University of St Andrews, UK from 2000 to 2006, and at School of Business, University of Reading, UK from 2006 to 2010. He is currently working on a project at School of Engineering, University of Portsmouth, UK. His research interests include information sharing, project management, and recommendation system.



Linda Yang received her PhD degree in computer science from Peking University, China in 1999, and her MS and BE degrees in computer software in 1996 and 1993, respectively. She has been a senior lecturer at University of Portsmouth, UK since 2004. Prior to that, she was a lecturer at the Robert Gordon University, a senior research fellow at University of St Andrews and Cardiff University, UK. Her research interests include information retrieval, recommendation system, data mining as well as their applications in social networks, health-informatics and business. She has worked on UK and European funded research projects since 1999 and published over 50 papers in refereed journals and conferences. She is a member of the IET, WES and HEA.

Junhao Wen received the PhD degree from the Chongqing University (CQU), China in 2008. He is a professor and the vice dean of the School of Software Engineering, CQU. His research interests include service computing, cloud computing, and software dependable engineering. He has published more than 80 refereed journal and conference papers in these areas. He has more than 30 research and industrial projects and developed many commercial systems and software tools.

Qingyu Xiong is the dean of the School of Software Engineering, Chongqing University (CQU), China. He received the BS and MS degrees from the School of Automation, CQU in 1986 and 1991, respectively, and the PhD degree from Kyushu University, Japan in 2002. His research interests include neural networks and their applications. He has published more than 100 journal and conference papers in these areas. Moreover, he has more than 20 research and applied grants.

Shun Li is a PhD student at the School of Software Engineering, Chongqing University (CQU), China. He received his BE degree from CQU in 2014. His research interests include service computing, recommendation system, and machine learning.