Preventing Shilling Attacks in Online Recommender Systems

Paul-Alexandru Chirita
L3S Research Center
University of Hannover
Hannover, Germany
chirita@l3s.de

Wolfgang Nejdl
L3S Research Center
University of Hannover
Hannover, Germany
nejdl@l3s.de

Cristian Zamfir
L3S Research Center
University of Hannover
Hannover, Germany
zamfir@l3s.de

ABSTRACT

Collaborative filtering techniques have been successfully employed in recommender systems in order to help users deal with information overload by making high quality personalized recommendations. However, such systems have been shown to be vulnerable to attacks in which malicious users with carefully chosen profiles are inserted into the system in order to push the predictions of some targeted items. In this paper we propose several metrics for analyzing rating patterns of malicious users and evaluate their potential for detecting such *shilling attacks*. Building upon these results, we propose and evaluate an algorithm for protecting recommender systems against shilling attacks. The algorithm can be employed for monitoring user ratings and removing shilling attacker profiles from the process of computing recommendations, thus maintaining the high quality of the recommendations.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.3.4 [**Information Storage and Retrieval**]: Systems and Software; H.3.5 [**Information Storage and Retrieval**]: Online Information Services

General Terms

Algorithms, Experimentation, Performance

Keywords

Web applications, Recommender systems, Collaborative filtering, Shilling attacks

1. INTRODUCTION

Recommender systems based on collaborative filtering play an increasing role in filtering information in an overloaded information system. They are not only helping users find relevant items, but are also beneficial to companies producing items by increasing both selling rate and cross-sales.

Currently, there are quite several commercial recommender systems used in e-commerce[1] [5], movie[2,3] and music recommendations[4] [14], etc., as well as some research oriented ones (see, for example, PocketLens [6], or ClickStream CF [3]), and even governmental ones (such as NASA's service for recommendation of related technical reports [7]). In such a collaborative filtering based recommender system, users build profiles by rating certain items, and obtain personalized recommendations for other, uknown items, based on the correlation between their ratings and those of other users.

The most popular types of algorithms for collaborative filtering (CF) are user-based and item-based:

1. User-based algorithms build for each user a neighborhood of users with similar opinions (i.e., ratings) in the system. Ratings from these users are then employed to generate recommendations for the target user.

2. Item-based algorithms compute a set of similar items for each item and use these similarities to compute recommendations.

Unfortunately, since good ratings promise a good selling rate, these systems are prone to manipulation from producers or malicious users. Examples of manipulation have been outlined in [4] and include attacks from popular systems like Amazon and eBay. Recent research has shown that most popular algorithms employed in current CF applications can be rather easily manipulated through biased profiles [4]. More specifically, this can be achieved by introducing fake user profiles that highly rate a set of target items, and then rate other items, in such a way that they become similar to many profiles of regular users. The desired result is known as a *shilling attack* and consists of either increasing (push attack) or lowering (nuke attack) the ratings of some target items.

Attacks on recommender systems can affect the quality of the prediction for many users, resulting in decreasing overall user satisfaction with the system. Such threats may cost users' time and money and pose a serious challenge to the recommender system administrators, who have to manually discover the shilling attackers. This vulnerability of recommender systems is even more severe if we think it actually extends to any personalized information system in which an

---

[1] http://www.amazon.com
[2] http://www.movielens.org
[3] http://www.tivo.com
[4] http://www.audioscrobbler.com

attacker can introduce fake profiles in order to increase the general interest for a set of target resources.

Based on the above observation that shilling attackers use synthetic profiles[5], in this paper we investigate the use of statistical metrics to reveal rating patterns of shilling attackers. We experimentally evaluate these metrics for existing zero-knowledge shilling attacks and propose an algorithm that makes use of them to detect and isolate shilling attackers. To the best of our knowledge, this is the first algorithm that effectively detects the most general attacks on recommender systems [4].

The rest of the paper is structured as follows: In Section 2 we introduce the most popular CF algorithms and outline existing work on developing and guarding against attacks on recommender systems. In Section 3 we define several statistical metrics, which could be utilized to identify rating patterns of shilling attackers, and then we empirically analyze each of them in Section 4. In Section 5 we propose an algorithm, which detects shilling attackers by exploiting these metrics. Finally, we show how it could be integrated into an web-based recommender systems in Section 6, and conclude with a summary and future work in Section 7.

2. BACKGROUND

2.1 Common CF Algorithms

User-based collaborative filtering. The most popular collaborative filtering algorithm is the kNN-based algorithm. Data is represented as a user × item matrix, with an entry *(u,i)* representing either the rating user u gave to item i, if she rated it, or *null* otherwise. Similarity between users is then computed using the Pearson correlation [11]:

$$W_{ij} = \frac{\sum_{k \in I}(R_{ik} - \bar{R}_i)(R_{jk} - \bar{R}_j)}{\sqrt{\sum_{k \in I}(R_{ik} - \bar{R}_i)^2 \sum_{k \in I}(R_{jk} - \bar{R}_j)^2}} \quad (1)$$

where I is the set of items users i and j both rated, R_{ik} is the rating user i gave to item k, and $\bar{R}_i$ is the average rating of user i. Finally, predictions for user i and item a are computed using the k-nearest neighbors formula below:

$$P_{ia} = \bar{R}_i + \frac{\sum_{j=1}^{k} W_{ij}(R_{ja} - \bar{R}_j)}{\sum_{j=1}^{k} W_{ij}} \quad (2)$$

Item-based collaborative filtering. Another popular CF algorithm is based on the item-item similarity [13]. Here, items are thought of as two vectors in the $|users|$ multidimensional space, and the similarity between items i and j can be computed using the cosine-based similarity:

$$Sim(i,j) = \frac{\vec{i} \cdot \vec{j}}{\| \vec{i} \| \cdot \| \vec{j} \|} \quad (3)$$

Then, the prediction for an item is computed using a weighted average of user's ratings R_{ua}, weighed by the similarity score:

$$P_{ui} = \frac{\sum_{all \ similar \ items,a} R_{ua} Sim(i,a)}{\sum_{all \ similar \ items,a} |sim(i,a)|} \quad (4)$$

---

[5]This is in fact reasonable, since no large-scale success could be achieved by manually inspecting items and rating them as a regular user would.

2.2 Identifying and Detecting Shilling Attacks

While there is a lot of work in the field of developing collaborative filtering algorithms, only recently some papers have concentrated on developing shilling attack models [9, 12] and on benchmarking the robustness of recommender systems against shilling attacks [4, 8].

Lam and Riedl [4] introduce the Random Bot and the Average Bot types of shilling attacks and evaluates their effectiveness in promoting the target items by computing the prediction shift and expected Top-N occupancy for these items in both user-based and item-based collaborative filtering environments.

A Random Bot attacker rates all the items in the system with the mean 3.6 out of 5 and a 1.1 deviation. The intuition behind this is that making random ratings within a certain average interval will allow the attacker to have a high influence in making predictions for other users. Depending on the objective of the attack, the items in the target set are rated with the minimum rating (for nuke attack) or maximum rating (for push attack). An Average Bot attacker is more effective but requires knowledge of the average rating for each item in the system. Each Average Bot attacker rates the items outside the target set randomly, following a normal distribution with a mean equal to the average rating for that item, thus becoming more similar to the real users than the Random Bot.

In [12] several other attack models are developed, under the assumption that the attacker has some knowledge about the ratings of the other users. We think that such knowledge is hard to obtain, if not impossible in a real world system. Another disadvantage of this approach is that the ratings introduced by an attacker are algorithm dependent. Finally, the detection of the attackers is not addressed in the paper. In fact, the only work that partially tackles this challenge is [15]. There, a spreading similarity algorithm is developed in order to detect groups of very similar shilling attackers. While this is indeed a first step, it only applies to a simplified attack scenario, whereas our algorithm applies to more general and powerful attack.

We think that zero-knowledge attacks such as the Random Bot are particularly interesting, since for the other attacks, recommender systems administrators could increase the privacy of user profiles using cryptographic means [6, 10, 2], thus falling back to the zero-knowledge ones. In general, the more insight a recommender system offers about its ratings, the more susceptible to attack it is, allowing powerful low cost (in terms of number of fake profiles) attacks to be mounted on the system.

3. METRICS FOR DETECTING RATING PATTERNS OF SHILLING ATTACKERS

3.1 Introduction

We have argued that shilling attacks could be very noxious to CF systems. Now, could we define an algorithm independent approach, which protects against these attacks by mining the rating patterns from the users database? We consider the answer to be positive. This section will lay the foundation towards such an approach, presenting first the statistical metrics that could be utilized to analyze user ratings, and then a naïve algorithm exploiting them. We will then complete our attack detection scheme in Section 5.

3.2 Metrics

In [1], a number of algorithm independent qualitative factors are used in analyzing the influence of a user on a recommender system. While the goal of [1] was not related to attacks at all, we think some of these factors could be useful in analyzing patterns for the fake profiles introduced by the different types of shilling attacks. More specifically, we found the following metrics suitable to address our problem of detecting shilling attacks:

1. *Number of Prediction-Differences (NPD)*
   NPD is defined for each user as the number of net prediction changes in the system after her removal from the system.

2. *Standard Deviation in User's Ratings*
   This metric represents the degree in which a rating given by a user to an item differs from her average ratings.

3. *Degree of Agreement with Other Users*
   The degree of agreement is in fact the average deviation in a user's ratings from the average rating of each item: $1/k \sum_{a=1}^{k} |R_{ia} - \bar{R}_a|$, where R_{ia} is the rating user i gave to item a and $\bar{R}_a$ is the average rating of item a.

4. *Degree of Similarity with Top Neighbors*
   As stated by its name, this metric describes the average similarity weight with the Top-K neighbors of a user. It uses the following formula: $\frac{\sum_{i=1}^{k} W_{ij}}{k}$

As we will see from Section 4 these metrics provide a useful insight into detecting shilling attackers, but are not sufficient for identifying attackers, as they output quite a few false positives. Therefore, we also defined a new measure, *Rating Deviation from Mean Agreement (RDMA)*. Intuitively, this can be seen as the measure of the deviation of agreement with other users on a set of target items, combined with the inverse rating frequency for these items. RDMA can be computed in the following way:

$$RDMA_j = \frac{\sum_{i=0}^{N_j} \frac{|r_{i,j} - Avg_i|}{NR_i}}{N_j} \qquad (5)$$

where N_j is the number of items user j rated, $r_{i,j}$ is the rating given by user j to item i, NR_i is the overall number of ratings in the system given to item i. Alternatively, one could also compute the number of ratings and the average rating using only a subset of users, thus giving a local view to our measure. We will discuss these variants in Section 5.

Since shilling attacks usually try to push items with low ratings, the users mounting such an attack will also have a high RDMA, because for the target items, the numerator of each term (the difference from the average rating) will be high, whereas the denominator (the number of predictions) will be low; thus the overall term will be high[6], and the attackers will be simply removed from the computation of recommendations.

---

[6]Users with very special tastes might also have a high RDMA value though. This is one of the reasons that determined us to seek a more complex algorithm for shilling attacks detection. The outcome will be presented in Section 5.

3.3 Basic Algorithm for Detecting Shilling Attackers

Considering the fact that attackers should have a high influence in the system in order to effectively promote the target items, we want the metrics in Section 3.2 to reveal distinctive features in the rating patterns. Attackers should therefore have very high values for *NPD*, *Average Similarity*, *Degree of agreement with other users*, and *RDMA*, as well as a very low value for *Standard Deviation in User Ratings*. The following algorithm detects shilling attackers based on these expectations:

Algorithm 1. Basic algorithm for detecting shilling attackers.

---

01. *Let* **MetricsLow**=*Standard Deviation in Ratings* and **MetricsHigh**= *NPD, Degree of agreement, Average Similarity, RDMA*
02. *for each* **m** in **MetricsHigh** and **MetricsLow**
03.   *for each* user u
04.   compute **m***(u)*
05. *for each* user u
06.   *if* u has high values in **MetricsHigh** and very low values in **MetricsLow** *then* u is a shilling attacker

---

In lines 2-4, the algorithm computes for each user the values for all statistical metrics, and then in lines 5-6 decides, based on her assessed probability of being an attacker, whether her profile will be discarded from the computation of recommendations or not.

4. EXPERIMENTS

4.1 Attack Scenario

An attack consists of a group of profiles that are introduced into the system in order to push the ratings of a set of target items. The target items are usually unpopular items (low average rating) that are not rated by many users[7]. Our experiments are conducted using the Random Bot shilling attack. In [4], the number of attacker profiles reaches almost 1% of the total number of users. The cost of an attack can be estimated as a function of the number of profiles that have to be introduced in the system. Usually, introducing more than 1% fake profiles can be considered infeasible, but we will go up to 3% attacker profiles. For implementing the experiments we used the open source MultiLens[8] collaborative filtering platform.

We have evaluated the patterns from section 3 for databases without any attackers and with 3% Random Bot attackers. The target items set is composed of three randomly selected items that have a low average rating, as well as a small number of ratings.

In our experiments we have used the MovieLens database containing 100,000 movie ratings for 1682 items from 943 users. For the attack scenario we introduced 30 additional users, which were Random Bot attackers with the same target items set. All users have rated at least 20 movies, with

---

[7]Popular items are already recommended by the system to some extent, and are therefore rated by a larger number of users. Thus, making them even more popular would usually imply a too costly effort from the attackers. We further discuss on this choice in Section 5.2.
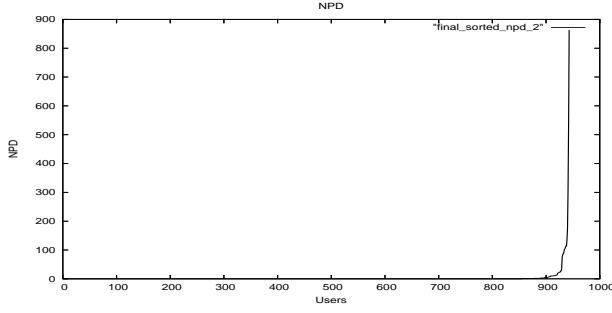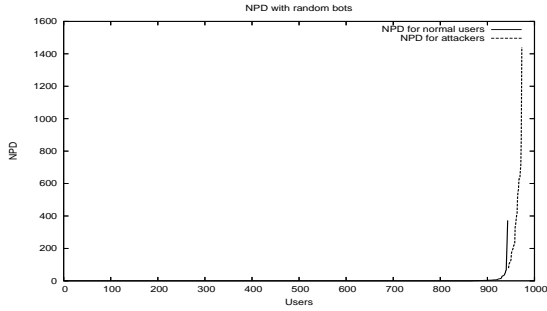[8]http://sourceforge.net/projects/multilens/

Figure 1: NPD without attackers

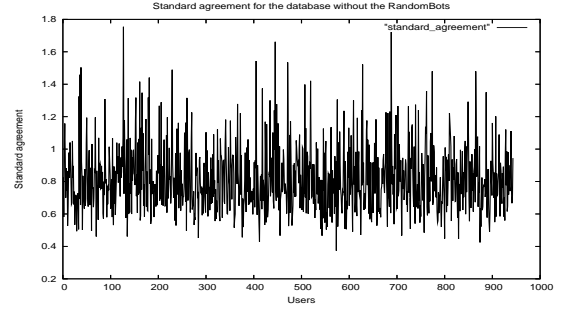Figure 2: NPD for a database with attackers

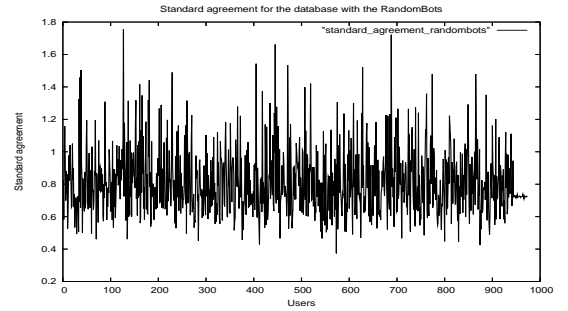Figure 3: Degree of Agreement for a database without attackers

Figure 4: Degree of agreement for a database with attackers. Notice that the variation of the *degree of agreement* for the attackers is very small (Attackers have user IDs between 944 and 973).

ratings between 1 and 5. The Random Bot attackers rated the items in the system as explained in Section 2.

Our experiments show that all the patterns studied are relevant in exposing behavior of the attackers. The values for the patterns studied are either high, or almost the same (with a small deviation) for most of attackers, but a few regular users can also be considered attackers if we use only one of these patterns to detect attackers. The rest of this section discusses each of the patterns in detail.

4.2 Using Rating Patterns to Detect Shilling Attacks

NPD. As discussed in [1], NPD shows a power-law distribution, most of the users having very low NPD, while only a few are having a very high NPD. This applies for both the database with and without attackers. Figure 1 presents the NPD values for the database without attackers. After having introduced the attackers (Figure 2), they will slightly decrease for the regular users, leaving the top NPD value to the malicious users. This is because the attackers are very similar to many users and removing them from the neighborhood of a user would result in prediction changes for all these users. However, we notice that they also overlap with the top 0.3% of the normal users.

Standard Deviation in User's Ratings. Random Bot attackers give random ratings within a relatively small interval, centered around 3.6. This distribution of ratings makes them outstanding from the database, because they are the only "users" having the Standard Deviation in Ratings close

to 0. Most users have a greater entropy in ratings and sometimes give extreme ratings like the minimum rating. However, some small percent of users have a small Standard Deviation in Ratings, so that attackers could disguise their behavior by increasing the entropy in their ratings and thus escaping detection using this pattern. Still, avoiding detection by increasing entropy will also decrease the power of attack, as it will decrease the attacker's similarity with other users in the system. Therefore, we conclude that analyzing this pattern for shilling attacks can be useful, as it will force malicious users to disguise their attacks, thus reducing the overall impact of the attack on the entire system.

Degree of Agreement With Similar Users. We have computed this metric using the top-25 similar users from the neighborhood formation phase of user-based collaborative filtering. In spite of the fact that attackers make random ratings, an interesting pattern we discovered was that they had almost the same value for this metric (Figures 3 and 4). Even though other regular users could also have very similar values, one could use the results from analyzing this pattern in order to reduce the false positives output by the attacker detection algorithm.

Average Similarity with Top Neighbors. The average similarity was also computed over each user's top-25 neighbors. We discovered that it resembles NPD: it has a
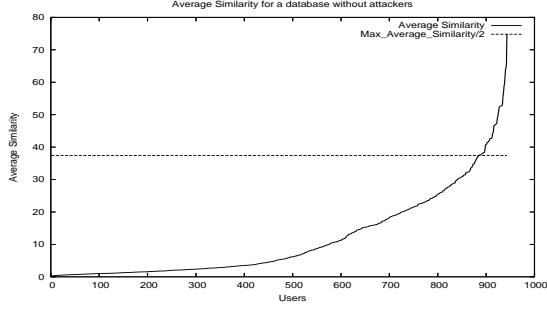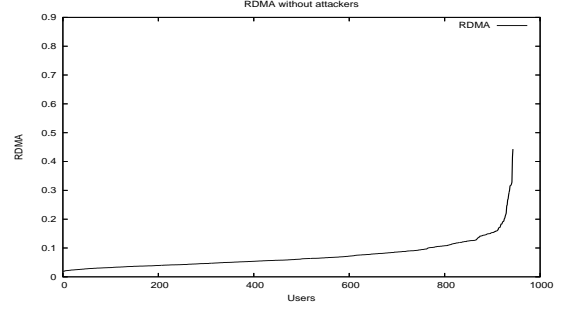
Figure 5: Average Similarity without attackers

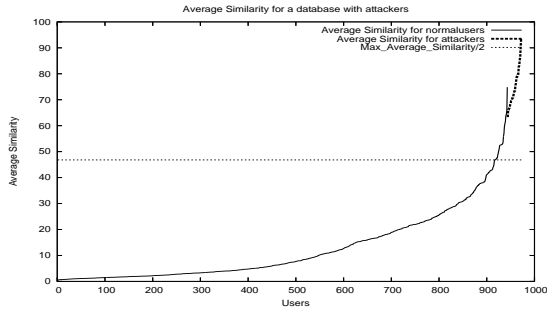Figure 7: RDMA without attackers

Figure 8: RDMA with attackers

Figure 6: Average Similarity with attackers
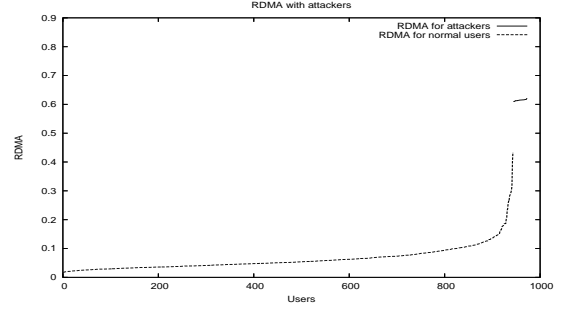
power law distribution and the values for the attackers overlap for some small set of users (Figures 5 and 6). While NPD exhibits scalability issues, computing the average similarity for each user is much faster, and thus preferable. More, selecting users that have a greater value than *1/2* of the maximum *average similarity* in the system, would include all the attackers in the output. Therefore, we chose to use this metric along with RDMA in our improved algorithm from Section 5.

RDMA. For a database without attackers, very few users had a high normalized RDMA, which was promisive, since we expected the attackers (once introduced) to have a very high RDMA. However, RDMA was high for the attackers only for a small attack size. Once we increased the attack size to 3%, several normal users had bigger RDMA than the attackers. This is partially because an attack of such a large size is enough to radically increase the average rating for the target items, so that regular users who rated these items with the minimum rating get in this case an increased RDMA. Generally, when the attackers give a rating centered around 3.6 to the items outside the target set, users that only expressed extreme like (the maximum rating) or utter dislike (the minimum rating) have an increased RDMA for a large-scale attack.

5. ENHANCED ALGORITHM FOR DETECTING SHILLING ATTACKERS

5.1 Description

When computing *RDMA*, we can use only a subset of the total users for computing the *Average Rating* and *Number of Predictions* for each of the items. For the database without any attackers, this results in very few users (2, in our experiments) having high RDMA so that removing them from the database will not affect the quality of the recommendations.

In the following, we describe an improved two step algorithm, which exploits the above mentioned idea: we first compute the average similarity with the top neighbors for all users and then select for computing *RDMA* only those users that have an average similarity smaller than *1/2* of the maximum average similarity in the system. We then associate with each value of *RDMA* a function that evaluates the probability that the respective user is a shilling attacker. The algorithm is depicted below.

Algorithm 2. Enhanced Algorithm for Detecting Shilling Attackers.

---

01. *for each* user
02.   Compute *Average_Similarity* using
    the Pearson Correlation
03. $max =$ the maximum Average_Similarity
    in the system
04. *for each item*
05.   Compute the average rating and the number
    of ratings using user's u profile if
    $Average\_Similarity(u) < max/2$

```
06. for each user u
07.    Compute normalized RDMA(u)
         using the average rating from 05.
08. avg = average RDMA over all users
09. for each user u
10.    if RDMA(u) < avg
11.        PS=0
12.    else
13.        PS=f(RDMA(u))
```

Where $f : [avg, 1] -> [0, 1]$

$$f(x) = \frac{1}{e^\alpha - 1}(e^{\alpha \frac{x - Avg\_RDMA}{1 - Avg\_RDMA}} - 1) \qquad (6)$$

For computing the average ratings and the number of pre-
dictions for each item, the algorithm selects (line 4) only
those users that have a high *Average Similarity*. In lines
5-7, using these values, we compute *RDMA* for each user.
Lines 8-12 assign a *0* shilling probability for users having
less than the average *RDMA* and use function f to compute
the shilling probability for the other users. We choose f such
that it evaluates to almost 1 for values of x very close to 1,
and very close to 0 for the rest of the values, thus trans-
lating the high values for *RDMA* into an almost 1 shilling
probability. We have tested different values of α and found
only a small variation in results (the experiments presented
here were conducted with $\alpha = 10$).

The algorithm exploits the fact that eliminating regular
users from computing predictions and average rating for
each item (as used in the equation for *RDMA*) does not
result in these users having a high *RDMA*. On the other
hand, eliminating the attackers and the regular users that
are detected as false positives in the first step results only
in the group of attackers having a high *RDMA* (see also the
discussion from the previous Section, as well as Figures 7
and 8).

The resulting plots for the overall shilling probability com-
puted by our algorithm are shown in Figure 9 for a database
without attackers, and in Figure 10 for a database with at-
tackers. The results show that our algorithm is very effective
against the Random Bot attack. In a separate stream of ex-
periments (not presented here due to space limitations), we
have obtained very similar results for an Average Bot type
of attack. We can thus conclude that our algorithm is ef-
ficient against both zero-knowledge and limited-knowledge
attacks.

5.2 Discussion

Target items. Our experiments have shown that many
items in the system are rated by only a few users, and their
ratings can be pushed very easily. Also, many items with
a low average rating have few ratings because users tend to
rate only the items that are recommended to them by the
system, which are usually the items that also have a high
average. Using these experimental observations we conclude
that items more likely to be target items for a push attack
are the ones that have both a low average rating and a small
number of ratings. Moreover, attacks are quite powerful
using this strategy. In Equation 2, the presence of only
three attackers in the neighborhood of one user is enough
to create a large prediction change and to push such target
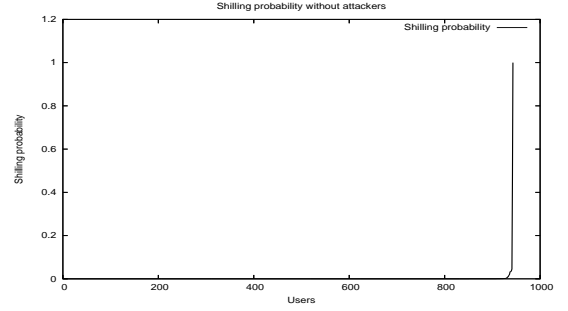items to the top-5 recommended items. The reason for this

**Figure 9: Shilling attack probability for a database
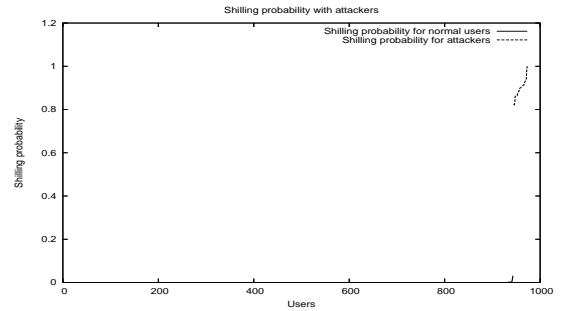without attackers**

**Figure 10: Shilling attack probability for a database
with attackers**

is that since the item is rarely rated, most of the terms in
the sum will be 0, and thus only the ratings given by the
attackers count. Because this item is in the target items
set, the attackers have given it the maximum rating, so that
the predicted rating for the item is also very close to the
maximum rating in the system.

Of course, once such an item is rated by the active user, it
will not be displayed any more in the recommendation list.
The more the actual rating and the predicted rating differ,
the less similar the active user and the attackers will become.
However, this is a slow process, and if the users are already
very similar, the active user has to rate many bad items
before the attacker is excluded from her neighborhood. In
other words, all collaborative filtering algorithms have the
power to adapt, but depending on the power of the attack
and other human factors like the frequency of use of the
system, this process can be very slow.

Employing an algorithm such as the one described in this
paper allows us to periodically monitor the system for the
rating patterns of shilling attackers and will provide a much
faster protection against such malicious users.

False positives. Experiments from Section 4 show that
Algorithm 1 leads to a high number of false positives in iden-
tifying attackers. On the other hand, the enhanced Algo-
rithm 2 does not output false positives even for large attack
sizes. Furthermore, it is scalable since it uses only two of the
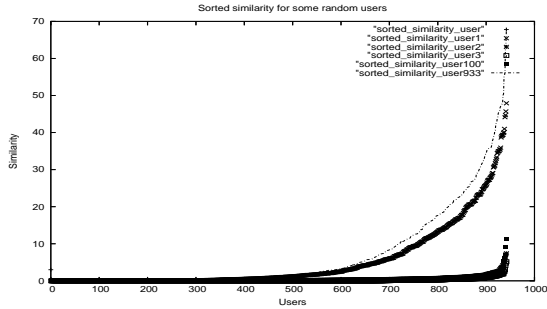metrics discussed in Section 3.2. Eliminating many falsely

Figure 11: Sorted similarity for some random users in the MovieLens database

identified shilling attackers from the recommendation process can result in loss of quality for the recommendations. This can be argued by looking at the power law distribution of similar neighbors in user-based collaborative filtering (Figure 11). For an active user u, there are a few very similar users and many users with much lower similarity values. If the algorithm outputs some false positives that are among the very similar users, then recommendations will be quite influenced by the less similar users and their quality will inherently decrease.

6. INTEGRATION WITH A WEB-BASED RECOMMENDER SYSTEM

The previous sections showed that rating patterns can be very useful in detecting attackers, with only a small number of false positives. Combining *RDMA* and the *Average Similarity Metric* results in a strong metric which is able to detect all attackers in the system in our experiments.

As ratings in a recommender system are given in a democratic way, a user may give an item the maximum rating even if this is considered a bad product (i.e., the average rating for this item in the whole system is very low). Thus, attackers in a collaborative filtering based recommender system should not necessarily be perceived as malicious by the other users, since the process of giving ratings to items is mostly a question of taste. It is thus arguable whether the users suspected of attack should be totally excluded from computing recommendations for the other users.

To address this issue, we present a modification to neighborhood formation in the user-based collaborative filtering algorithm that weighs each user's influence in generating recommendations according to her probability of being a shilling attacker:

$$W'_{i,j} = W_{i,j} * (1 - PS_j) \qquad (7)$$

where PS_j is the probability for shilling, and $W_{i,j}$ is the similarity between users i and j. Since the probability for regular users is almost 0, and for attackers is almost 1, the effect of using this modification in a user-based collaborative filtering algorithm is to practically filter out malicious users from making recommendations, while reducing the influence of users with special preferences.

In an on-line web based recommender system, we can provide the user with an additional button to be pressed for activating protection against shilling attackers. This protection will probably be activated by a user who gets predictions she strongly disagrees with. More, a user should be able to chose to activate or deactivate the protection against shilling attackers for each Top-N recommendation list she receives. However, the recommender system would continuously monitor for shilling attack patterns and compute the appropriate shilling attack probabilities. The only overhead for the recommender system would be to take the shilling probability into account when computing the recommendation list. Thus, there is no need to rebuild the model, making the algorithm scalable to large numbers of users and items.

7. CONCLUSIONS AND FUTURE WORK

This paper proposed and investigated the use of statistical metrics for detecting patterns of shilling attackers in a recommender system. We have evaluated these metrics for the MovieLens database and shown that attackers do indeed exhibit special, noticeable rating patterns. We also proposed the use of an additional metric, *RDMA*, to measure a user's disagreement with the other users in the database, weighted by the inverse rating frequency of her rated items.

Based on these investigations we have developed an algorithm that computes the probability of a user to be a shilling attacker by studying the rating patterns wihtin the system, namely exploiting the *RDMA* and *Average Similarity* metrics. We empirically proved this algorithm to be effective in identifying all attacks defined in [4], even for large attack sizes of up to 3% of the system.

We intend to further improve *RDMA*, as well as study other rating patterns for attackers. Moreover, we are interested in developing and analyzing other more complex low cost shilling attacks on recommender systems.

8. REFERENCES

[1] G. K. Al Mamunur Rashid and J. Riedl. Influence in ratings-based recommender systems: An algorithm-independent approach. In *Proceedings of SIAM International Conference on Data Mining*, 2005.

[2] J. Canny. Collaborative filtering with privacy. In *SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy*, page 45, Washington, DC, USA, 2002. IEEE Computer Society.

[3] D.-H. Kim, V. Atluri, M. Bieber, N. Adam, and Y. Yesha. A clickstream-based collaborative filtering personalization model: towards a better performance. In *WIDM '04: Proceedings of the 6th annual ACM international workshop on Web information and data management*, pages 88–95, New York, NY, USA, 2004. ACM Press.

[4] S. K. Lam and J. Riedl. Shilling recommender systems for fun and profit. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 393–402, New York, NY, USA, 2004. ACM Press.

[5] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[6] B. N. Miller, J. A. Konstan, and J. Riedl. Pocketlens: Toward a personal recommender system. *ACM Trans. Inf. Syst.*, 22(3):437–476, 2004.

[7] M. L. Nelson, J. Bollen, J. R. Calhoun, and C. E. Mackey. User evaluation of the nasa technical report server recommendation service. In *WIDM '04: Proceedings of the 6th annual ACM international workshop on Web information and data management*, pages 144–151, New York, NY, USA, 2004. ACM Press.

[8] M. O'Mahony, N. Hurley, N. Kushmerick, and G. Silvestre. Collaborative recommendation: A robustness analysis. *ACM Trans. Inter. Tech.*, 4(4):344–377, 2004.

[9] M. P. O'Mahony, N. Hurley, and G. C. M. Silvestre. Promoting recommendations: An attack on collaborative filtering. In *DEXA '02: Proceedings of the 13th International Conference on Database and Expert Systems Applications*, pages 494–503, London, UK, 2002. Springer-Verlag.

[10] H. Polat and W. Du. Privacy-preserving collaborative filtering using randomized perturbation techniques. In *ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining*, page 625, Washington, DC, USA, 2003. IEEE Computer Society.

[11] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, 1994. ACM.

[12] R. Z. Robin Burke, Bamshad Mobasher and R. Bhaumik. Identifying attack models for secure recommendation. In *Beyond Personalization 2005*, 2005.

[13] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *World Wide Web*, pages 285–295, 2001.

[14] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1, pages 210–217, 1995.

[15] H.-J. Z. Xue-Feng Su and Z. Chen. Finding group shilling in recommendation system. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, 2005.