Shilling Attack Prevention for Recommender Systems Using Social-based Clustering

Thesis by

Tak Man Desmond Lee, BSc.

Submitted in Partial Fulfillment of the Requirements for the degree of

Masters of Science

King Abdullah University of Science and Technology
Mathematical and Computer Sciences and Engineering Division
Applied Mathematics and Computational Science Department

Thuwal, Makkah Province, Kingdom of Saudi Arabia

June, 2011
The dissertation/thesis of Tak Man Desmond Lee is approved by the examination committee.

Committee Chairperson: Dr. David E. Keyes
Committee Member: Dr. Xiangliang Zhang
Committee Member: Dr. Aron Ahmadia
ABSTRACT

Shilling Attack Prevention for Recommender Systems Using Social-based Clustering

Tak Man Desmond Lee

A Recommender System (RS) is a system that utilizes user and item information to predict the feeling of users towards unfamiliar items. Recommender Systems have become popular tools for online stores due to their usefulness in confidently recommending items to users. A popular algorithm for recommender system is Collaborative Filtering (CF). CF uses other users’ profiles to predict whether a user is interested in a particular object. This system, however, is vulnerable to malicious users seeking to promote items by manipulating rating predictions with fake user profiles. Profiles with behaviors similar to “victim” users alter the prediction of a Recommender System. Manipulating rating predictions through injected profiles is referred to as a shilling attack. It is important to develop shilling attack prevention frameworks for to protect the trustworthiness of Recommender Systems.

In this thesis, we will demonstrate a new methodology that utilizes social information to prevent malicious users from manipulating the prediction system. The key element in our new methodology rests upon the concept of trust among real users, an element we claim absent among malicious profiles. In order to use trust information
for shilling attack prevention, we first develop a weighting system which makes the system rely more on trustworthy users when making predictions. We then use this trust information to cluster out untrustworthy users to improve rating robustness. The robustness of the new and classic systems is then evaluated with data from a public commercial consumer RS, Epinions.com. Several complexity reduction procedures are also introduced to make implementing the algorithms mentioned possible for a huge commercial database.
Acknowledgments

My most sincere gratitude goes to my co-supervisors Prof. Xiangliang Zhang and Prof. David E. Keyes for their continuous guidance and encouragement throughout the course of this work. Their enthusiasm and valuable insights made my study very enjoyable and exciting and ultimately fruitful with rich experience. I would also like to thank KAUST and people at KAUST for a superb research environment.

My further gratitude goes to my collaborators Dr. Georgios Pitsillis and Dr. Wei Wang who have collaborated with us in the research. Their valuable comments and ideas very much nurtured this work.

My eternal gratitude towards my parents who support me always from near and far.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Abbreviations</td>
<td>9</td>
</tr>
<tr>
<td>List of Illustrations</td>
<td>9</td>
</tr>
<tr>
<td>List of Tables</td>
<td>11</td>
</tr>
<tr>
<td><strong>I</strong> Introduction</td>
<td>12</td>
</tr>
<tr>
<td>I.1 Motivation of Recommender System (RS)</td>
<td>12</td>
</tr>
<tr>
<td>I.1.1 Example</td>
<td>12</td>
</tr>
<tr>
<td>I.2 RS Overview</td>
<td>13</td>
</tr>
<tr>
<td>I.2.1 User-based RS</td>
<td>14</td>
</tr>
<tr>
<td>I.2.2 Item-based RS</td>
<td>15</td>
</tr>
<tr>
<td>I.3 Shilling Attack Model</td>
<td>15</td>
</tr>
<tr>
<td>I.4 Shilling Attack Prevention</td>
<td>17</td>
</tr>
<tr>
<td><strong>II</strong> Related Work</td>
<td>18</td>
</tr>
<tr>
<td>II.1 Modeling of Shilling Attacks</td>
<td>18</td>
</tr>
<tr>
<td>II.2 Detection of Shilling Attack</td>
<td>19</td>
</tr>
<tr>
<td>II.2.1 Clustering</td>
<td>20</td>
</tr>
<tr>
<td>II.3 Trust</td>
<td>20</td>
</tr>
<tr>
<td><strong>III</strong> Shilling Attack Prevention by Clustering</td>
<td>22</td>
</tr>
</tbody>
</table>
## List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CluTr</td>
<td>Clustering with Trust-based filtering</td>
</tr>
<tr>
<td>WCluTr</td>
<td>Weighted Clustering with Trust-based filtering</td>
</tr>
<tr>
<td>CDF</td>
<td>Cumulative Distribution Function</td>
</tr>
<tr>
<td>CF</td>
<td>Collaborative Filtering</td>
</tr>
<tr>
<td>MAE</td>
<td>Mean Absolute Error</td>
</tr>
<tr>
<td>RS</td>
<td>Recommender System</td>
</tr>
</tbody>
</table>
List of Illustrations

III.1 Clustering with Trust-based filtering (CluTr) example flowchart 26

IV.1 CDF of Prediction Shift $\Delta_i$ in CluTr against weighted similarity and baseline CF 34

IV.2 CDF of Prediction Shift $\Delta_i$ in CluTr against different scenarios of trust and baseline CF 35

IV.3 CDF of Prediction Shift $\Delta_i$ in Weighted Clustering with Trust-based filtering (WCluTr) against different scenarios of trust and baseline CF 36
### List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.1</td>
<td>An example of Average attacks</td>
<td>16</td>
</tr>
<tr>
<td>IV.1</td>
<td>The MAE of weighted similarity, CluTr and WCluTr algorithms comparing to the baseline algorithm without clustering and trust</td>
<td>31</td>
</tr>
<tr>
<td>IV.2</td>
<td>The MAE of Average attacks in weighted similarity, CluTr and WCluTr algorithms comparing to the baseline algorithm without clustering and trust</td>
<td>32</td>
</tr>
<tr>
<td>IV.3</td>
<td>The mean of Prediction Shift of Average attacks in weighted similarity, CluTr and WCluTr algorithms comparing to the baseline algorithm without clustering and trust</td>
<td>32</td>
</tr>
</tbody>
</table>
Chapter I

Introduction

Recommender Systems (RS) are recognized tools in e-commerce for improving product promotions based on customers’ integrated preferences and purchasing behaviors. The system has demonstrated financial worth and is an active area of research in both academia and the industry.

I.1 Motivation of RS

In today’s highly concentrated marketplace, consumers and producers alike face information overflow. In making purchase and sales decisions, personalized product recommendations are in high demand. RS aids marketplace decision making by ranking products for a particular individual’s need based on his or her preferences. Preferences here mean the purchases of the customer in the past.

I.1.1 Example

A customer rates several science fiction books he has purchased with a very good rating (i.e., 5 stars out of 5) at an online bookstore. A naive, and obvious way to attract the customer to buy another book is to recommend another science fiction
novel. To recover more subtle relationship and make more useful recommendation, one may consider a recommender system which recommends popular science readings to readers of science fiction. This is an example of the user-based Collaborative Filtering (CF) approach mentioned in section [I.2.1]. This example or other findings of RS may not always be intuitive, which is exactly why Recommender Systems have a growing niche. CF is particularly suited for capturing subtle relations while eliminating unnecessary content analysis of books.

1.2 RS Overview

The goal of a Recommender System is to recommend items that are of interest to the user. Such systems are very versatile and diverse in employing all kinds of information, such as a user’s profile, a user’s browsing history, and an item profile. Some systems may simply recommend the top ten or most popular items. Other systems, instead, analyze the item content and try to fit it to customer’s preference. One of the best known techniques is Collaborative Filtering (CF). It predicts item ratings according to collaboration of a huge amount of data sources. The output of an RS for a particular user and items is usually a numerical value. Items associated with a higher value are likely to be of interest to the user. This value is the predicted rating. This value can be outside of the original rating range in some RS algorithms. Thus, a common practice of web stores is to display items with the highest value but never show the predicted ratings.

In this thesis, we focus on Collaborative Filtering Recommender System that predict the rating of a user for a particular item by using the rating records of a customer’s neighbors. Neighbors are defined as those who display similar preferences.

Two types of CF techniques are discussed: (1) user-based, and (2) item-based.
I.2.1 User-based RS

User-based RS assumes similar users will give similar ratings to the same item. In this work, similarity is taken to be of Pearson’s type. Pearson similarity is widely used as the similarity calculation in RS. The similarity $w_{u,v}$ of user $u$ and user $v$ is:

$$w_{u,v} = \frac{\text{cov}(U, V)}{\sigma_U \sigma_V}$$  \hspace{1cm} (I.1)

where $U, V$ are the product rating vectors of user $u$ and $v$, $\text{cov}(U, V)$ is the covariance function and $\sigma_U$ and $\sigma_V$ are the standard deviations of vectors $U$ and $V$ respectively. Product rating vectors contain the user’s ratings on all items in the system. Unrated items are marked 0 for convenience; however, they are ignored in the calculation of covariance. The covariance function here follows the common definition:

$$\text{cov}(U, V) = \frac{1}{N - 1} \sum_{i=1}^{N} (U_i - \mu_U) (V_i - \mu_V)$$  \hspace{1cm} (I.2)

where $\mu_U$ and $\mu_V$ are the means of vectors $U$ and $V$ respectively.

In a classic user-based approach, prediction of a user’s rating on an item is the weighted mean of ratings of the user’s neighbors. Classic user-based CF systems employ Resnick’s formula in predicting the rating $p_{u,i}$ of user $u$ on item $i$,

$$p_{u,i} = r_u + \frac{\sum_{v \in C_{u,i}} [w_{u,v} (r_{v,i} - r_v)]}{\sum_{v \in C_{u,i}} |w_{u,v}|}$$  \hspace{1cm} (I.3)

In equation $r_{v,i}$ is the user $v$ rating for item $i$, $r_u$ is user $u$’s average rating for his rated items, and $w_{u,v}$ is the Pearson similarity between user $u$ and $v$, one of $u$’s neighbors. $C_{u,i}$ is a subset of users considered neighbors of user $u$ that have rated item $i$. In the classic case $C_{u,i}$ will contain the top $m$ most similar users who have also rated item $i$. We also enforce a minimum similarity $\text{minsimm}$. A user will be included in $C_{u,i}$ only if he has the similarity greater than $\text{minsimm}$. This is used to ensure that
I.2.2 Item-based RS

Item-based RS is similar to user-based RS. It differs in that the prediction of the ratings is based on similar items instead of similar users. Item-based RS assumes items with similar ratings history from all users will receive similar ratings by the same user in the future. Inter-item similarity is also calculated by Pearson similarity.

We predict items’ ratings by the weighted mean of similar items as follows:

\[ p_{u,i} = \frac{\sum_{j \in similaritems_i} [w_{i,j} (r_{u,j})]}{\sum_{j \in similaritems_i} |w_{i,j}|} \]  

(I.4)

In equation I.4, \( r_{u,j} \) denotes the rating of user \( u \) for item \( j \) and \( w_{i,j} \) is the Pearson similarity between item \( i \) and \( j \). \( similaritems_i \) is a subset of items considered neighbors of item \( i \) and user \( u \) has rated it. In the classic case, it will be the top \( m \) most similar items rated by user \( u \).

I.3 Shilling Attack Model

Both user-based and item-based CF make prediction using neighbors. Unfortunately, they are prone to manipulations. For example, malicious users may intend to “push” (increase) the predicted rating to promote their own product or “nuke” (decrease) the predicted rating of a competitor’s product. This is an issue especially when malicious users’ injected profiles are recognized as neighbors of genuine users, which leads to poor recommendations. This could potentially prohibit it from achieving its goal and hurt the reputation of the system.

Under a push attack, the injected profile will give the target item the highest rating in the system. In the injected profiles, non-target items may also be rated as a filler or camouflage. A number of attack algorithms for manipulating predictions
have been proposed with this framework. The most well-known of these are the Randombot and the AverageBot [1].

Randombot attacks are relatively simple and require very little information from the system. The ratings of these filler items are random sampling from uniform distribution to imitate a normal user.

The Average attack is more sophisticated. It takes advantage of readily accessible average ratings of items, further concealing the working and existence of shillers. In an AverageBot attack, there are $N_{fake}$ injected profiles that rate the target item by the highest possible rating, while they rate the other randomly selected $N_{filler}$ items by samples of a normal distribution whose mean is the averaged ratings of these items given by a set of normal users. The normal users’ predicted ratings are then affected, because they probably have high similarity with the injected fake users on rating the filler items.

We show an example of AverageBot attacks in Table I.1 where two fake user profiles are injected that rate a target item 5 (here, the highest rating) then randomly rate a filler item $i$ by a random value sampled from $N(\mu_i, \sigma)$. $\mu_i$ is the average rating of item $i$. $\sigma$ is a constant set to 1.1 as suggested by [1], considering that this information is not available to attacker and they are forced to make an educated guess. The non-target and non-filler items are unchanged. The setting of parameters $N_{fake}$ and $N_{filler}$ in the attack model will be discussed in section IV.4.

<table>
<thead>
<tr>
<th>user1</th>
<th>Item1</th>
<th>Item2</th>
<th>Item3</th>
<th>Item target</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>user2</td>
<td></td>
<td>3</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>user fake</td>
<td>$N(\mu_1, \sigma)$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>user fake</td>
<td>$N(\mu_1, \sigma)$</td>
<td>$N(\mu_2, \sigma)$</td>
<td></td>
<td>5</td>
</tr>
</tbody>
</table>

These attacks are effective since neighbors in CF are usually decided by the similarity among users. So filler items also are an agent to increase the similarity between
real and fake users. Measures to shield the RS from these attacks are devised and discussed herein.

I.4 Shilling Attack Prevention

The literature has suggested clustering as a means to separate genuine users from shillers and improve RS performance [2, 3, 4]. If we partition users into different clusters, and only consider neighbors in the same cluster, we can separate real and fake users and improve the performance of the RS. If the fake users are separated, they have no effect on the prediction of the ratings to the real users.

One metric for clustering is the social relations among users for determining one’s authenticity. In this thesis, a social relationship is a directional trust given by a user to another. For example, when a user finds another user’s review on certain item helpful, he can flag that he trusts that user. We will consider it as a uni-directional trust. On the other hand, we can use social network friendship connection status as a proxy for bi-directional trust information.

This thesis uses two different methods to combine the trust information in Recommender Systems. The first one is based on weighted similarity. If we could find a way to increase genuine user’s weight and decrease shiller’s weight in prediction, we would obtain a more robust way against attacks. Pitsilis, et al. have shown that using a weight derived from the trust will improve accuracy [4]. Thus we can improve both the accuracy and robustness in the system with this method.

Another idea is to perform trust filtering in clustering via extra trust information to separate genuine users from shillers. We assume that shillers will receive little trust from genuine users, and be excluded from any user cluster.
Chapter II

Related Work

Shilling attacks have implication in both theory and business practice, and are receiv-
ing growing attention. We will review the literature on modeling the attack, detecting
the attack and the use of trusts in Recommender Systems.

II.1 Modeling of Shilling Attacks

In a seminal paper in the field of shilling attacks, Lam and Riedl explored several
open questions regarding the effectiveness of shilling attacks \[1\]. They discussed
two attack algorithms: RandomBot and AverageBot, and measured their effect on
items with the rating predicted and how often a recommendation is made to a user.
User-based CF algorithms were more vulnerable to shilling attacks than item-based
CF algorithms and shilling attacks were not detectable using traditional measures of
algorithm performance, such as recommendation and prediction metrics. The paper
illustrated a research direction for improving the robustness of RS systems.

Mobasher, et al. studied six different attack models and measured their effective-
ness in both user-based and item-based CF \[5\]. The prediction shift and hit ratio
were used to measure how attacks can affect the RS. Their results show that the dif-
ferent types of attacks can effectively and practically harm the standard user-based
and item-based CF algorithms.

Attack detection algorithms based on \( k \)-means \[6\] and \( k \)-nn \[7\] rely on the assumption of high similarities among injected profiles. Upon the advances in attack detection algorithms, to challenge them, Cheng and Hurley \[8\] created effective Random attacks and Average attacks with very low pair-wise similarities, dropping the assumption of high similarities among malicious attack profiles in \[9\]. The authors showed that \( k \)-means and \( k \)-nn cannot detect such low diversity attacks since they rely on the high similarity assumption.

II.2 Detection of Shilling Attack

Su, \textit{et al.} focused on a type of coalition shilling attack, the group shilling \[10\]. They constructed a bipartite graph for the users and items and used the similarity spreading algorithm to find user clusters. Then they labeled one cluster as “abnormal group of shilling users”, if the size (number of members) and average similarity of this cluster was smaller than a pre-defined threshold.

Chirita, \textit{et al.} proposed four different metrics for analyzing rating patterns of malicious users, including the number of prediction-differences, the standard deviation in user’s ratings, the degree of agreement with other users, and the degree of similarity with top neighbors \[11\]. The shilling attacks are detected by monitoring and evaluating the rating patterns.

Zhang, \textit{et al.} detected attacks by treating the ratings of an item as a time series according to their given time \[12\]. The sample average and sample entropy within a disjoint window of \( k \) consecutive ratings are calculated to capture the change in an item’s likability and the distributional change in an item’s ratings.
II.2.1 Clustering

Clustering is a powerful tool in shilling attack prevention. Technically, clustering attempts to mark similar instances with uniform labels. The research can focus on the clustering method itself or on what instance’s properties should be used as input. Most of the time the users’ ratings are used as input.

Using the assumption that shillers work together and are highly correlated, Mehta used PLSA (Probabilistic Latent Semantics Analysis, a soft clustering method) and PCA (Principal Component Analysis) to eliminate clusters of shilling users [9] and improve detection accuracy. Mehta, et al. have also tried to use SVD (Singular Value Decomposition) to cluster out shillers which results in a very stable algorithm with a high computation complexity [13].

Other studies cluster out shillers by investigating their special properties. Previous works consider unique behavior of the shillers and attempt to cluster them out [14,15]. Rathee used statistical method to catch outlier and considered them shillers. Rathee did not use the ratings but the statistics (e.g., the mean and SD of ratings) as input to the clustering. Different clustering methods, including $k$-means are used in clustering phase to cluster those outliers.

Ungar, et al. studied clustering for improved accuracy of CF [16]. They used the ratings as input, and tested several clustering methods and their variations. Their results show that clustering may improve accuracy in certain cases. This is not related to the shilling attack, but shows that clustering does not generate a negative side effect on accuracy.

II.3 Trust

Trust information is another method to filter out the shillers, which uses the assumption that shillers would receive minimum trust from real users. Trust in this thesis
refers to the explicit indication of a user trusting another user. In a RS system, a user $u$ can flag another user $v$ trusted and this will be treated as the indication $u$ trust $v$.

Dubois, et al. [2] studied an RS using trust information in clustering users. They assumed that trusted users are identified by the similarity of their preferences and then predicted ratings by the mean of the members in the cluster. They then compared this prediction from trust cluster method to the traditional Pearson correlation CF. In both cases a higher weight is given to the neighbor user in the same cluster. They assessed its accuracy and observed any measurable improvements.

Yet another approach utilizing trust information in improving the accuracy of RS is suggested by Pitsilis, et al. [4]. They replaced the weight $w_{u,v}$ in the traditional Resnick’s predation formula (see equation 1.3). Classically, the weight equals the Pearson similarity. They redefined it to include additional trust information. They also obtained a small improvement.

Although clustering and trust have been combined to improve recommendations [2, 4], to the best of our knowledge, there is no work which uses both to protect an RS. Using trust information to strengthen or weaken the user similarities is a promising path as suggested by [4].
Chapter III

Shilling Attack Prevention by Clustering

III.1 Data Manipulation with Matrices

In order to speed up the calculation of intersections of each pair of users in the set in our model, we manage them in matrix form. In contrast to a trivial way which is a nested for-loop implementation, we can use all the tweaks and optimization designed for matrix manipulation.

III.1.1 Matrix form Intersection

Here is a matrix version of how to perform a intersection calculation in an $M$-user $N$-item environment for all $M$ users, i.e., find out the number of items mutually rated for all pairs of users. We present our data in the following form: an $M \times N$ matrix $A$ where the element $a_{i,j}$ is the rating of user $i$ on item $j$. It could be any integer from 0 to 5, where 0 means it is unrated. Then the intersection mentioned is just

\[ C = AA^T \]  

(III.1)
where $C$ is the matrix where element $c_{i,j}$ is the number of mutually rated item for the user pair $i, j$.

### III.2 Weighted Similarity

We define a new function for measuring the social similarity of users derived from their social connectivity with others. Our idea is inspired by [4], which shows that the prediction accuracy is improved through weighting the similarities by trust. Their proposed weighted similarity $w_{u,v}^{\text{weighted}}$ is defined as:

$$w_{u,v}^{\text{weighted}} = w_{u,v} (1 + \alpha S_{\text{average}}) \quad (III.2)$$

with $\alpha = 1/2$ in [4], where the parameter is not justified. The parameter $\alpha$ is an adjustment to the contribution of trust similarity and should be a non-negative number. Different values of $\alpha$ reflect different weights toward trust.

We call $S_{\text{average}}$ the Social Similarity between two users and it is defined as in equation (III.3) with $T_i$ and $T_j$ denoted as the sets of trusted users by user $i$ and user $j$ respectively.

$$S_{\text{average}}(i,j) = \frac{1}{2} \left( \frac{|T_i \cap T_j|}{|T_i|} + \frac{|T_i \cap T_j|}{|T_j|} \right) \quad (III.3)$$

$|T_i \cap T_j|$ is the number of elements in the set of intersection of $T_i$ and $T_j$. One can refer to equation (III.1) for an effective way to calculate the intersection.

### III.3 Clustering

Clustering assigns $N$ instances (points) into different clusters (disjoint subsets of the original input) so that instances in the same cluster are similar in some sense. The explicit goal is different from case to case. One way is to define a function in terms of the grouping and minimize that function. In our case, the clustering
aims to maximize the sum of inner-cluster averaged Pearson similarities. We use the clustering algorithm $k$-means++ which is derived from a widely adopted clustering algorithm, $k$-means.

**III.3.1 $k$-means**

$k$-means is a popular [17] clustering method proposed in [18]. $k$-means partition $N$ points to one of the $K$ clusters(subsets) $(C_1,C_2,...,C_K)$, where $K$ is a predefined parameter. A cluster center $m_k$ is the average of all data points that belong to cluster $k$. Every data point $x_i$ is associated with its nearest cluster center, one of the \{m_k, k = 1...K\}. $k$-means clustering minimizes the sum of squared distance between each point and its cluster center, given by,

$$\sum_{k=1}^{K} \sum_{x_i \in C_k} d(x_i - m_k)^2 \quad \text{where} \quad m_k = \frac{1}{|C_k|} \sum_{x_i \in C_k} x_i$$  (III.4)

We use Lloyd’s algorithm [19], a standard heuristic in $k$-means. It begins by choosing a number of centers and recursively adjusting the centers’ locations. This is supposed to minimize the sum mentioned in (III.4) However, it is not guarantee to reach the global minimum. Lloyd’s algorithm may converge close to a local minimum due to randomly selected initial cluster centers. For more reliable results, $k$-means++ is used and described below.

**III.3.2 $k$-means++**

$k$-means++ is proposed by Arthur, et al. [6]. By incorporating carefully selected initial cluster centers, it increases the chances of reaching close to the global minimum. It is shown that on top of improving the accuracy, computation time can also be improved in some case. The algorithm follows.

1. Randomly choose the first center $x_{c1}$ among the data points $(x_1, x_2, ..., x_n)$.
2. Calculate the distance \((d_1, d_2, ..., d_n)\) of each data point to its closest center point.

3. A new center will be chosen from the remaining non-center data points. The probability \(p_i\) of a particular data point \(x_i\) is chosen is proportional to the square \(d_i\) of its distance calculated in step 2. Using basic principle in probability we can calculate \(p_i\).

4. Repeat 2,3 until \(k\) centers have been chosen. Then proceed with the standard \(k\)-means.

### III.3.3 CluTr-Clustering with Trust-based filtering

Making use of trust information we can obtain a more reliable clustering result. In CluTr we proposed, users are first clustered by \(k\)-means++ based on their profile similarity. To exclude the potential attackers from the recommendation process, we apply filtering on the clusters. Following a conjecture in real social networks, suspicious and unreliable users are considered as those whose incoming trust from any other members of the same cluster is null. Moreover, to make the computation of predictions meaningful we exclude clusters whose population are smaller than a threshold \(t\). The reason is the recommendation is only based on the data within the cluster and a small cluster doesn’t contain enough information to make an effective prediction.

Fig. [III.1] is an illustration of how the algorithm works.

The description of each process in the figure follows.

1. Obtain user cluster\((C_1, C_2, ..., C_K)\) by \(k\)-means++

2. Dismiss small clusters, label all users inside as if they belong to no clusters.

3. The trust filtering does the following:

   for each remaining cluster \(C_k\) do
   
   for user \(u (x_u)\) in the cluster \(C_k\) do
   
   if user \(u (x_u)\) has no incoming trust within the cluster then
Figure III.1: CluTr example flowchart
label it as belonging to no cluster

end if

end for

d for

4. In the prediction step, we calculate $p_{u,i}$, using equation [I.3]. We change the notation of $C$ to $C'$ in the equation to avoid confusion to the $C$ above.

$$p_{u,i} = \bar{r}_u + \frac{\sum_{v \in C'_{u,i}} [w_{u,v} (r_{v,i} - \bar{r}_v)]}{\sum_{v \in C'_{u,i}} |w_{u,v}|}$$ \hspace{1cm} (III.5)

If $u$ belongs to no cluster, we construct $C'_{u,i}$ from every user disregarding what cluster they belong to, \textit{i.e.}, the original way in Chapter I. Otherwise $C'_{u,i}$ will be constructed only with users in the cluster that user $u$ belongs to.

Users in a user cluster obtained after trust filtering may not be trusted by some others in the cluster. However, as an approximation that reduces computation complexity, we find the results satisfactory as shown in chapter [IV].

### III.3.4 WCluTr-Weighted Clustering with Trust-based filtering

Improve upon CluTr, which rigidly filtering out suspicious users, we now incorporate trust information into the computation of the weighted similarity mentioned above between users and develop the method WCluTr. It takes advantages of both CluTr and weighted similarity and should produce better results.

The procedure of WCluTr follows:

1. Preparation and $C_{u,i}$ are same as CluTr.

2. To calculate $p_{u,i}$, use the same formula used in CluTr but replace the $w_{u,v}$ specified in equation [III.2].
Chapter IV

Experimental Evaluation

IV.1 Data and Attack

The data for evaluation of the algorithm is taken from the commercial RS Epinions.com, collected by Paolo Massa in 2003 with a Python crawler [20]. This data provides an explicit directional trust information together with user ratings on a huge number of items. To limit the computational complexity and to simulate a comparable data set size to other references [1], we use a subset of the data that contains the first 5,000 users randomly crawled and returned, their 206,000 trust expressions and 1.1 million ratings for 103,000 products.

We implement the AverageBot model proposed in [1], as it is the state of the art now, according to the test by Mobasher et al. [21]. See section I.3 for more details.

IV.2 Trust Attack Model

We strengthen RS by using social trust information. Unfortunately, attackers can inject shilling social trust too. Inject trust here means the injected profiles can label themselves trusting other users. We extend the Profile Injection model to reflect the social behavior of a potential attacker. We assume an injected profile trusts normal
users and his accomplices, following the intuitive scenario of attempting to establish social connections with victims. The shilling trust model, along with the Average attack model, assumes generation of random connectivity in the social network, originated from the fake users. To examine whether attackers can take advantage of the knowledge in the trust system, we have modeled four different scenarios.

1. **none**: no trust is injected;
2. **trust1**: attackers randomly trust 10% of all users;
3. **trust2**: attackers trust all users;
4. **trust3**: attackers trust all fake users plus 10% of normal users.

### IV.3 Metrics

The algorithms are designed to defend against attacks. However it makes no sense when it does not return meaningful (accurate) results. Accuracy is measured here by Mean Absolute Error (MAE) to the ratings masked out. Suppose in the ratings matrix provided, $r_{i,j}$ is a rating rated and $p_{i,j}$ is the predicted rating of user $i$ on item $j$. Since $p_{i,j}$ doesn’t depend on $r_{i,j}$ in all algorithms that we are going to test. We define the MAE as:

$$
\text{MAE} = \text{mean}(|r_{i,j} - p_{i,j}|) \forall i, j \text{ s.t. } r_{i,j} \text{ exists}
$$

The mean function is arithmetic mean over all $i, j$ pairs that $r_{i,j}$ exists. A smaller MAE means results have less error. It is an intuitive measurement. However, after an attack, it will measure both the error from the system itself and the attack. Error from the system itself is not the focus of this thesis as our concern is the robustness. Robustness here is related to how much the prediction is changed after the attack. The less the change, the higher the robustness. To measure only the error introduced by the attack we use *Prediction Shift*. 

Prediction Shift is another metric we use to measure the effectiveness of attacks in RS. As proposed and used in [1, 5], Prediction Shift is defined as the difference between the predicted rating before and after attacks. A positive value means that attacks have raised the rating. In our experiment, the effectiveness of a particular attack with a particular algorithm towards a particular item $i$ is measured by

$$\Delta_i = \frac{1}{|U|} \sum_{u \in U} (p'_{u,i} - p_{u,i})$$ (IV.2)

which is the average of difference between predicted ratings after attack ($p'_{u,i}$) and predicted rating before attack ($p_{u,i}$) on all users $U$. An advantage of Prediction Shift is that it does not require a true rating by the user. Therefore we can test it on any user and item pairs.

To obtain a reliable result we launched a number of independent attacks on different target items $i$, and report the mean and probability distribution of Prediction Shift $\Delta_i$ on a set of samples $i$.

### IV.4 Parameters

In equation IV.3,

$$p_{u,i} = \bar{r}_u + \frac{\sum_{v \in C_{u,i}} [w_{u,v} (r_{v,i} - r_v)]}{\sum_{v \in C_{u,i}} |w_{u,v}|}$$ (IV.3)

$m=20$ and $\minsim=0.1$ are used in constructing $C_{u,i}$ as suggested in [1].

The weighted coefficient $\alpha$ in III.2 is set to 3.5; this is chosen as the best result of repeating trial of different value ranging from 0.5 to 5 with a interval of 0.25 on the same data.

The parameter $K$ in kmeans++ is set to 10. Small clusters are dismissed if their size is smaller than $t=30$.

In the AverageBot attack model, the number of injected fake users $N_{fake}$ is set to
25, 50 and 100 for evaluating the robustness of our designed methods w.r.t. various intensity of attacks. The number of filler items $N_{\text{filler}}$ is set to $N_{\text{all items}} \times 1\% = 1000$.

**IV.5 Experimental Results**

First, we measure the performance of algorithms without shilling attacks. We employ the MAE measure mentioned in section IV.3. Smaller MAE means better results.

Table IV.1: The MAE of weighted similarity, CluTr and WCluTr algorithms comparing to the baseline algorithm without clustering and trust

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0.8035</td>
</tr>
<tr>
<td>Weighted Similarity</td>
<td>0.8019</td>
</tr>
<tr>
<td>CluTr</td>
<td>0.7413</td>
</tr>
<tr>
<td>WCluTr</td>
<td>0.7380</td>
</tr>
</tbody>
</table>

The Table IV.1 demonstrates comparable performance of our algorithms with the baseline approach. CluTr and WCluTr have better performance on predicting users rating than the baseline collaborative filtering algorithm without the purpose on shilling attack prevention. With the confidence of CluTr and WCluTr from these positive evaluation results, we next evaluate how good CluTr and WCluTr will protect RS systems from shilling attacks.

In Table IV.2 we show the accuracy measure MAE of various algorithms after attack. In Table IV.3 we show the Prediction Shift of Average attacks $\bar{\Delta}=1/|I| \sum_{i \in I} \Delta_i$, by taking the average of $\Delta_i$ on a set of samples $i$. The baseline for comparing is the conventional user-based CF algorithm without clustering and trust information. Our proposed algorithms CluTr and WCluTr are evaluated over four different scenarios of injecting fake trust information, as discussed in section IV.2 with the number of injected fake users $N_{\text{fake}}$ set to 25, 50 and 100. For some rare situations when no user in a cluster rated an item probably for the reason that the rating users are filtered out due to trust issues, we make up the prediction of rating by conventional user-based
Table IV.2: The MAE of Average attacks in weighted similarity, CluTr and WCluTr algorithms comparing to the baseline algorithm without clustering and trust

<table>
<thead>
<tr>
<th>Number of fake users: $N_{fake}$</th>
<th>Trust Scenarios</th>
<th>MAE</th>
<th></th>
<th>CluTr</th>
<th>WCluTr</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>none</td>
<td>No Clustering</td>
<td>Weighted Similarity</td>
<td>0.8769</td>
<td>0.8753</td>
</tr>
<tr>
<td></td>
<td>trust1</td>
<td>1.0196</td>
<td>0.9547</td>
<td>0.8804</td>
<td>0.8713</td>
</tr>
<tr>
<td></td>
<td>trust2</td>
<td></td>
<td></td>
<td>0.8390</td>
<td>0.8286</td>
</tr>
<tr>
<td></td>
<td>trust3</td>
<td></td>
<td></td>
<td>0.8638</td>
<td>0.8520</td>
</tr>
<tr>
<td>25</td>
<td>none</td>
<td>1.1891</td>
<td>1.0960</td>
<td>0.8884</td>
<td>0.8862</td>
</tr>
<tr>
<td></td>
<td>trust1</td>
<td></td>
<td></td>
<td>0.9226</td>
<td>0.9162</td>
</tr>
<tr>
<td></td>
<td>trust2</td>
<td></td>
<td></td>
<td>0.8184</td>
<td>0.8174</td>
</tr>
<tr>
<td></td>
<td>trust3</td>
<td></td>
<td></td>
<td>0.8117</td>
<td>0.8082</td>
</tr>
<tr>
<td>50</td>
<td>none</td>
<td>1.3235</td>
<td>1.2117</td>
<td>0.8305</td>
<td>0.8225</td>
</tr>
<tr>
<td></td>
<td>trust1</td>
<td></td>
<td></td>
<td>0.8615</td>
<td>0.8484</td>
</tr>
<tr>
<td></td>
<td>trust2</td>
<td></td>
<td></td>
<td>0.8367</td>
<td>0.8304</td>
</tr>
<tr>
<td></td>
<td>trust3</td>
<td></td>
<td></td>
<td>0.9015</td>
<td>0.8920</td>
</tr>
<tr>
<td>100</td>
<td>none</td>
<td>0.8506</td>
<td>0.8821</td>
<td>0.4075</td>
<td>0.4039</td>
</tr>
<tr>
<td></td>
<td>trust1</td>
<td></td>
<td></td>
<td>0.3995</td>
<td>0.3976</td>
</tr>
<tr>
<td></td>
<td>trust2</td>
<td></td>
<td></td>
<td>0.3801</td>
<td>0.3841</td>
</tr>
<tr>
<td></td>
<td>trust3</td>
<td></td>
<td></td>
<td>0.3491</td>
<td>0.3467</td>
</tr>
<tr>
<td></td>
<td>none</td>
<td>1.2290</td>
<td>1.0518</td>
<td>0.6665</td>
<td>0.6596</td>
</tr>
<tr>
<td></td>
<td>trust1</td>
<td></td>
<td></td>
<td>0.6224</td>
<td>0.6002</td>
</tr>
<tr>
<td></td>
<td>trust2</td>
<td></td>
<td></td>
<td>0.5916</td>
<td>0.5808</td>
</tr>
<tr>
<td></td>
<td>trust3</td>
<td></td>
<td></td>
<td>0.5179</td>
<td>0.5074</td>
</tr>
<tr>
<td></td>
<td>none</td>
<td>1.5791</td>
<td>1.5286</td>
<td>0.8033</td>
<td>0.7914</td>
</tr>
<tr>
<td></td>
<td>trust1</td>
<td></td>
<td></td>
<td>0.8311</td>
<td>0.8100</td>
</tr>
<tr>
<td></td>
<td>trust2</td>
<td></td>
<td></td>
<td>0.8194</td>
<td>0.8044</td>
</tr>
<tr>
<td></td>
<td>trust3</td>
<td></td>
<td></td>
<td>0.7342</td>
<td>0.7308</td>
</tr>
</tbody>
</table>

Table IV.3: The mean of Prediction Shift of Average attacks in weighted similarity, CluTr and WCluTr algorithms comparing to the baseline algorithm without clustering and trust

<table>
<thead>
<tr>
<th>Number of fake users: $N_{fake}$</th>
<th>Trust Scenarios</th>
<th>Prediction Shift</th>
<th></th>
<th>CluTr</th>
<th>WCluTr</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>none</td>
<td>No Clustering</td>
<td>Weighted Similarity</td>
<td>0.4075</td>
<td>0.4039</td>
</tr>
<tr>
<td></td>
<td>trust1</td>
<td></td>
<td></td>
<td>0.3995</td>
<td>0.3976</td>
</tr>
<tr>
<td></td>
<td>trust2</td>
<td></td>
<td></td>
<td>0.3801</td>
<td>0.3841</td>
</tr>
<tr>
<td></td>
<td>trust3</td>
<td></td>
<td></td>
<td>0.3491</td>
<td>0.3467</td>
</tr>
<tr>
<td>25</td>
<td>none</td>
<td>0.8506</td>
<td>0.8821</td>
<td>0.6665</td>
<td>0.6596</td>
</tr>
<tr>
<td></td>
<td>trust1</td>
<td></td>
<td></td>
<td>0.6224</td>
<td>0.6002</td>
</tr>
<tr>
<td></td>
<td>trust2</td>
<td></td>
<td></td>
<td>0.5916</td>
<td>0.5808</td>
</tr>
<tr>
<td></td>
<td>trust3</td>
<td></td>
<td></td>
<td>0.5179</td>
<td>0.5074</td>
</tr>
<tr>
<td>50</td>
<td>none</td>
<td>1.2290</td>
<td>1.0518</td>
<td>0.8033</td>
<td>0.7914</td>
</tr>
<tr>
<td></td>
<td>trust1</td>
<td></td>
<td></td>
<td>0.8311</td>
<td>0.8100</td>
</tr>
<tr>
<td></td>
<td>trust2</td>
<td></td>
<td></td>
<td>0.8194</td>
<td>0.8044</td>
</tr>
<tr>
<td></td>
<td>trust3</td>
<td></td>
<td></td>
<td>0.7342</td>
<td>0.7308</td>
</tr>
<tr>
<td>100</td>
<td>none</td>
<td>1.5791</td>
<td>1.5286</td>
<td>0.8033</td>
<td>0.7914</td>
</tr>
<tr>
<td></td>
<td>trust1</td>
<td></td>
<td></td>
<td>0.8311</td>
<td>0.8100</td>
</tr>
<tr>
<td></td>
<td>trust2</td>
<td></td>
<td></td>
<td>0.8194</td>
<td>0.8044</td>
</tr>
<tr>
<td></td>
<td>trust3</td>
<td></td>
<td></td>
<td>0.7342</td>
<td>0.7308</td>
</tr>
</tbody>
</table>
CF. All we want to test here is the robustness; in our experiment, a smaller amount of Prediction Shift means more robust.

From Table IV.2, we can see that CluTr and WCluTr outperform others in every case. When \( N_{fake} \) is higher, MAE remains roughly the same for CluTr and WCluTr but baseline method and weighted similarity perform badly.

We would like to focus the Prediction Shift to study the robustness part independently just as mentioned in section IV.3. As seen in Table IV.3, the conventional user-based CF algorithm without clustering and trust is adversely affected by Average attacks with mean Prediction Shift valued from 0.85 to 1.5. Especially when more fake users are injected, the predicted ratings will be pushed to a higher value. Such high rating shift is enough to change the victim’s opinion a target item.

The weighted similarity performance is not very attractive. It doesn’t contribute much to the robustness (less than 15\% decrease of Prediction Shift in both \( N_{fake} = 50 \) and 100 case) or even increase the Prediction Shift in \( N_{fake} = 25 \) case.

The proposed method CluTr and WCluTr have much lower Prediction Shift than the baseline CF algorithm. In a number of systems, ratings 1-5 are treated as “least preferred”, “less preferred”, “neutral”, “preferred” and “most preferred” accordingly. When attackers inject fake users, the pushed ratings shifts are less than 0.85, which will not change a user’s opinion on an item from “less preferred” to “preferred”. Moreover, CluTr and WCluTr have good performance no matter how attackers inject shilling trust information in the 4 different scenarios.

As arithmetic mean could be a biased measure on a set of data, Cumulative Distribution Function (CDF) is chosen to represent the whole results in a graphical form. We show the CDF of Prediction Shift \( \Delta_i \) in Fig. IV.1, IV.2 and IV.3. CDF describes the probability that the Prediction Shift \( \Delta_i \) will be found at a value less than or equal to \( x \), i.e., \( Pr(\Delta_i \leq x) \). A robust RS should have \( \Delta_i \) as small as possible, i.e., the CDF curve should go up quickly to 1 with \( \Delta_i \) increasing from 0 to a small
value. For simplicity, we merge all data for different values of $N_{fake}$.

In Fig. IV.1, we compare the weighted similarity CF against baseline CF. Both are without clustering and trust. We can see that the weighted similarity perform slightly better than the baseline. Thus, it has a slightly improvement on robustness.

In Fig. IV.2 we show the CDF of the Prediction Shift in the baseline CF without clustering and trust, and our method CluTr evaluated on 4 different scenarios of injecting fake trust information mentioned in section IV.2. Given the scenario that Prediction Shift should not exceed 1 (or 0.5), we find that $Pr(\Delta_i \leq 1) = 0.604$ (or $Pr(\Delta_i \leq 0.5) = 0.34$) in the baseline CF, and $Pr(\Delta_i \leq 1) \approx 0.981$ (or $Pr(\Delta_i \leq 0.5) = 0.95$) in CluTr. Prediction Shift $\Delta_i$ in the baseline CF has 40% (66%) chance to exceed 1 (0.5), while it only has 2% (5%) chance to exceed 1 (0.5) in CluTr.

We can interpret the promising results of CluTr method as follows: when AverageBot attacks happened, only 2% (5%) of the target items will be affected by raising the predicted rating by 1 (0.5). CluTr is 20 times more robust than the conventional
Figure IV.2: CDF of Prediction Shift $\Delta_i$ in CluTr against different scenarios of trust and baseline CF user-based CF algorithm. No matter how the attackers inject fake trust information, CluTr has good performance in all cases.

Fig. IV.3 shows that WCluTr is at least as good as CluTr in terms of robustness. Observing the CDF curves, we may conclude that WCluTr performs slightly better than CluTr.
Figure IV.3: CDF of Prediction Shift $\Delta_i$ in WCLuTr against different scenarios of trust and baseline CF
Chapter V

Conclusion and Future work

V.1 Contribution and Discussion

In this thesis, two effective attack-resistant algorithms for RS are proposed. CluTr and WCluTr, incorporate trust information in clustering to build robust RS against shillers. CluTr uses trust to filter out the suspicious fake users in the formed clusters. WCluTr additionally uses trust information to strengthen the similarities of among genuine users and to weaken the similarities between fake users and others. To present the shilling attack scenario, we studied one popular shilling attack, Averagebot, which strongly affects the standard user-based CF algorithm. We also inject the potential fake trust information to obfuscate the mutual trust among genuine users.

We evaluate the proposed method on data under various attack scenarios. The experimental results show that our proposed methods CluTr and WCluTr are much more robust than conventional user-based CF algorithm against Averagebot attacks. Interestingly our simulation showed that only 2% (5%) of the target items will be affected by raising the predicted rating by 1 (0.5), which is just enough to change user opinion for one discrete level (considering users are giving discrete ratings from 1 to 5).
V.2 Future Work

The work presented in this thesis can be extended in the following directions. In the real world, we may have difficulties obtaining the trust information. One of the directions is to use the social connection from a social network such as Facebook. The popularity of social network brings us the chance to derive more social link information. However, there could be challenges associated with this extension. The social information will be a mutual form instead of a directional form we used in this thesis, this might be mitigated with emerging social networks which orients upon directional social link, such as Google+. However, the implicit trust information might not be as useful as those collected explicitly. Privacy issues might also plague this potential future extension. However, we are optimistic given the latest trends in the development of more intelligent social networks. These hindering factors may well be solved with technical advancements, enabling this latest strand of research.
APPENDICES
Appendix A

Papers Submitted and Under Preparation

Bibliography/References


