The Impact of Attack Profile Classification on the Robustness of Collaborative Recommendation *

Chad Williams, Runa Bhaumik, Robin Burke, Bamshad Mobasher
Center for Web Intelligence, DePaul University
School of Computer Science, Telecommunication, and Information Systems
Chicago, Illinois, USA
{cwilli43, rbhaumik, rburke, mobasher}@cs.depaul.edu

ABSTRACT

Collaborative recommender systems have been shown to be vulnerable to profile injection attacks. By injecting a large number of biased profiles into a system, attackers can manipulate the predictions of targeted items. To decrease this risk, researchers have begun to study mechanisms for detecting and preventing profile injection attacks. In prior work, we proposed several attributes for attack detection and have shown that a classifier built with them can be highly successful at identifying attack profiles. In this paper, we extend our work through a more detailed analysis of the information gain associated with these attributes across the dimensions of attack type and profile size. We then evaluate their combined effectiveness at improving the robustness of user based recommender systems.

1. INTRODUCTION

Due to the open nature of collaborative systems, there is little to prevent attackers from inserting fake profiles in an attempt to bias the system in their favor. These attacks where a malicious user enters biased profiles in order to influence the system's behavior have been termed "shilling" or "profile injection" attacks. In theory, an attacker could swamp an unprotected system with enough profiles to control its recommendations completely. As a result, the vulnerabilities and robustness of collaborative recommender systems has been the subject of recent research [5, 1, 9, 12]. While there are techniques to increase the effort required to create profiles like requiring users to respond to a captcha[1] before an account is created, such measures also discourage participation thus decreasing the collaborative user base. As long as profiles are accepted without extraordinary measures to ensure their authenticity, it is possible for an attacker to launch such an attack. However this does not mean defending against such attacks is impossible.

Recent efforts have focused on reducing the impact of profile injection attacks through detecting and discounting suspected attack profiles. Chirita et al. [6] proposed several metrics for analyzing rating patterns of malicious users and evaluated their potential for detecting such attacks. Su, et al. [17] developed a spreading similarity algorithm for detecting groups of very similar attackers, which was shown to be effective for a simplified attack scenario. O'Mahony et al. [13] developed several techniques to defend against the attacks described in [9] and [12], including new strategies for neighborhood selection and similarity weight transformations. In our prior work we introduced an attack model-specific approach to profile classification and explored its effectiveness at detecting random, average, and segment push attacks [4, 11]. In [3] we showed our attack model-specific approach to be more effective at detecting smaller, more difficult to detect attacks than the technique described in [6].

Our approach to detecting attacks via pattern classification uses known attack models to build a training set of authentic and attack profiles, where standard data mining techniques are then applied to build a classifier. With such an approach, the closer an attacker imitates a known attack model, the greater the chance of detection. Of course, the attacker may build profiles that deviate from these models and thereby evade detection. However, our most effective attack models (described below) were derived by reverse engineering the recommendation algorithms to maximize their impact. We hypothesize, therefore, that they are optimal in the sense of providing maximum impact on the recommender system with the least amount of effort from the attack.

Our detection model is based on constructing a set of attributes that are calculated for each profile in the database. Supervised learning methods are then used to build classifiers based on these attributes, which are trained to discriminate between genuine profiles and those that are part of an attack. In this work we apply nearest-neighbor classification using kNN. Once attack profiles have been detected, they are eliminated to reduce the bias introduced by the attack. Ideally all attack profiles would be ignored and the system would function as if no bias had been injected. For a system to be considered robust, it should be able to withstand a direct attack on an item with minimal prediction shift.

In [3] we studied the classification performance of a single classifier in detecting a variety of push and nuke attacks and

*This research was supported in part by the National Science Foundation Cyber Trust program under Grant IIS-0430303.

[1] www.captcha.net/

WEBKDD'06 August 20, 2006, Philadelphia, Pennsylvania, USA

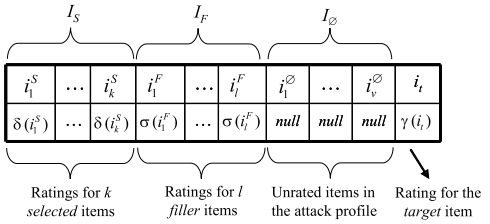

Figure 1: The general form of an attack profile.

compared our model-specific approach with that described in [6]. We extend the work in [3] by providing a deeper evaluation of the information gain associated with each of the detection attributes across the dimensions of attack type and profile size. In addition, we analyze the impact of these detection schemes on the robustness of a kNN recommender system. In particular, we measure the robustness via change in the predicted rating of a target item before and after attack. We show that such a detection scheme can improve the stability of a recommender system, keeping its predicted ratings steady under most attack scenarios.

2. ATTACK MODELS

For our purposes, a profile injection attack against a recommender system consists of a set of *attack profiles* inserted into the system with the aim of altering the system's recommendation behavior with respect to a single target item i_t. An attack that aims to promote i_t, making it recommended more often, is called a *push attack*, and one designed to make i_t recommended less often is a *nuke attack* [12].

An attack model is an approach to constructing the attack profiles, based on knowledge about the recommender system, its rating database, its products, and/or its users. The general form of an attack profile is depicted in Figure 1. The attack profile consists of an m-dimensional vector of ratings, were m is the total number of items in the system. The profile is partitioned in four parts. The null partition, $I_\emptyset$, are those items that have not been rated in the profile. The single target item i_t will be given a rating as determined by the function γ, generally this will be either the maximum or minimum possible rating, depending on the attack type. As described below, some attacks require identifying a group of items for special treatment during the attack. This special set I_S receives ratings as specified by the function δ. Finally, there is a set of filler items I_F whose ratings are added to complete the profile. Their ratings are provided by the function σ. It is the strategy for selecting items in I_S and I_F and the functions γ, σ, and δ that define an attack model and give it its character.

Two basic attack models, introduced originally in [9] are the *random* and *average* attacks. Both of these models involve the generation of attack profiles using randomly assigned ratings given to some filler items in the profile. In the random attack, the assigned ratings are based on the overall distribution of user ratings in the database. In our formalism, I_S is empty, the contents of I_F are selected randomly, and the function σ generates random ratings centered on the overall average rating in the database. The average attack is very similar, but the rating for each filler item is computed based on more specific knowledge of the individual mean for each item.

Of these attacks, the average attack is by far the more effective, but it may be impractical to mount, given the degree of system-specific knowledge of the ratings distribution that it requires. Further, as we show in [2], it is ineffectual and hence unlikely to be employed against an item-based formulation of collaborative recommendation. Our own experiments yielded three additional attack models: the bandwagon, segment and love/hate attacks described below. See [5, 1, 2] for additional details.

The *bandwagon attack* is similar to the random attack, but it uses a small amount of additional knowledge, namely the identification of a few of the most popular items in a particular domain: blockbuster movies, top-selling recordings, etc. This information is easy to obtain and not dependent on any specifics of the system under attack. The set I_S contains these popular items and they are given high ratings in the attack profiles. In our studies, the bandwagon attack works almost as well as the much more knowledge-intensive average attack.

The *segment attack* is designed specifically as an attack against the item-based algorithm. Item-based collaborative recommendation generates neighborhoods of similar items, rather than neighborhoods of similar users. The goal of the attack therefore is to maximize the similarity between the target item and the *segment items* in I_S. The segment items are those well-liked by the market segment to which the target item i_t is aimed. The items in I_S are given high ratings to increase the similarity between them and the target item; the filler items are given low ratings, to decrease the similarity between these items and the target item. This attack proved to be highly effective against the item-based algorithm as expected, but it also works well against user-based collaborative recommendation.

Our experiments also showed that the segment attack worked poorly as a nuke attack, as it was difficult to construct variants of it that were effective. The dislikes of a market segment are much more dispersed than its preferences, thus it is more difficult to construct a set I_S of items disliked by the target audience than to construct one that is generally liked. Our final attack model, the *love/hate attack* is a simple one that, nonetheless, is quite effective against both item-based and user-based algorithms as a nuke attack. It associates a low rating with the target item and high ratings with the filler items I_F (I_S is not used).

3. ATTACK PROFILE CLASSIFICATION

Our aim is to learn to label each profile as either being part of an attack or as coming from a genuine user. As described above, our approach is classification learning based on attributes derived from each individual profile. These attributes come in two varieties: generic and model-specific. The generic attributes are basic descriptive statistics that attempt to capture some of the characteristics that will tend to make an attacker's profile look different from a genuine user. The model-specific attributes are implemented to detect characteristics of profiles generated by known attack models.

3.1 Generic Attributes

We hypothesize the overall statistical signature of attack profiles will differ significantly from that of authentic profiles. This difference comes from two sources: the ratings given to the target item and I_S segment, and the distribu-

tion of ratings among the filler items. As many researchers in the area have theorized [9, 6, 12, 10], it is unlikely if not unrealistic for an attacker to have complete knowledge of the ratings in a real system. As a result, generated profiles will deviate from rating patterns seen for authentic users. This variance may be manifested in many ways, including an abnormal deviation from the system average rating, or an unusual number of ratings in a profile. As a result, an attribute that captures these anomalies is likely to be informative in identifying attack profiles.

Prior work in attack profile classification has focused on detecting the general anomalies in attack profiles. Chirita et al. [6] introduced several attributes for detecting these differences often associated with attack profiles. One of these attributes, *Rating Deviation from Mean Agreement* (RDMA), was intended to identify attackers through examining the profile's average deviation per item, weighted by the inverse of the number of ratings for that item. We propose two variants of the RDMA attribute which we have found to be valuable as well when used in a supervised learning context.

First, we propose a new attribute *Weighted Deviation from Mean Agreement* (WDMA) that is strongly based on RDMA, but we have found to provide higher information gain. Let U be the universe of all users u in the database. Let P_u be a profile for user u, consisting of a set of ratings $r_{u,i}$ for some items i in the universe of items to be rated. Let n_u be the size of this profile in terms of the numbers of ratings. Let l_i be the number of ratings provided for item i by all users, and $\overline{r_i}$ be the average of these ratings. The WDMA attribute can be computed in the following way:

$$WDMA_u = \frac{\sum_{i=0}^{n_u} \frac{|r_{u,i} - \overline{r_i}|}{l_i^2}}{n_u}$$

This attribute places higher weight on rating deviations for sparse items than the original RDMA attribute.

The second variation of the RDMA measure which we call *Weighted Degree of Agreement* (WDA) uses only the numerator of the RDMA equation and can be computed as follows:

$$WDA_u = \sum_{i=0}^{n_u} \frac{|r_{u,i} - \overline{r_i}|}{l_i}$$

This captures the sum of the differences of the profile's ratings from the item's average rating divided by the item's rating frequency.

In addition to rating deviations, some researchers have hypothesized that attack profiles are likely to have a higher similarity with their closest neighbors than real users would, because they are all being generated using the same process whereas genuine users have preferences that are more dispersed [6, 15]. This hypothesis was confirmed in our earlier experiments, which found that the most effective attacks are those in which a large number of profiles with very similar characteristics are introduced. This intuition is captured in the *Degree of Similarity with Top Neighbors* (DegSim) feature, also introduced in [6].

The DegSim attribute is based on the average similarity of the profile's k nearest neighbors and is calculated as follows:

$$DegSim_u = \frac{\sum_{v \in neighbors(u)} W_{u,v}}{k}$$

where $W_{u,v}$ is the similarity between users u and v calculated via Pearson's correlation, and k is the number of neighbors.

One well-known characteristic of correlation-based measures is their instability when the number of data points is small. Since it is the number of items co-rated by two users that determines their similarity, this factor can be taken into account and similarity decreased when two users have few items that they have co-rated. This feature is computed as follows. Let $I_{u,v}$ be the set of items i such that ratings exist for i in both profiles u and v, that is $r_{u,i}$ and $r_{v,i}$ are defined. $|I_{u,v}|$ is the size of this set. The similarity of profiles u and v is adjusted as follows:

$$W'_{u,v} = W_{u,v} \frac{|I_{u,v}|}{d}, \text{ if } |I_{u,v}| < d$$

The co-rate factor can be taken into account when calculating *DegSim*, producing a slightly different attribute *DegSim′*.

A third generic attribute that we have introduced is based on the number of total ratings in a given profile. Some attacks require profiles that rate many if not all of the items in the system. If there is are large number of possible items, it is unlikely that such profiles could come from a real user, who would have to enter them all manually, as opposed to a soft-bot implementing a profile injection attack. We capture this idea with the measure *Length Variance* (LengthVar) which measures of how much the length of a given profile varies from the average length in the database.

$$LengthVar_u = \frac{|n_u - \overline{n_u}|}{\sum_{u \in U} (n_u - \overline{n_u})^2}$$

where $\overline{n_u}$ is the average length of a profile in the system.

3.2 Model-Specific Attributes

In our experiments, we found that the generic attributes are insufficient for distinguishing a true attack profiles from eccentric but authentic profiles. This is especially true when the profiles are small, containing fewer filler items. Such attacks can still be successful in influencing recommendation results, so we seek to augment the generic attributes with some that are designed specifically to match the characteristics of known attack models.

As shown in Section 2, attacks models can be defined based on the characteristics of the attack profile partitions i_t (the target item), I_S (selected items), and I_F (filler items). Model-specific attributes are those that aim to recognize the distinctive signature of a particular attack model. These attributes are based on partitioning each profile in such a way as to maximize the profile's similarity to one generated by a known attack model. Statistical features of the ratings that make up the partition can then be used as detection attributes. One useful property of partition-based features is that their derivation can be sensitive to additional information (such as time-series or critical mass data) that suggests likely attack targets.

Our detection model discovers partitions of each profile that maximizes its similarity to a known attack model. To model this partitioning, each profile is split into two sets. The set $P_{u,T}$ contains all items in the profile with the profile's maximum rating (or minimum in the case of a nuke attack); the set $P_{u,F}$ consists of all other ratings in the profile. Thus the intention is for $P_{u,T}$ to approximate $\{i_t\} \cup I_S$ and $P_{u,F}$ to approximate I_F. (We do not attempt to differentiate i_t from I_S.) It is these partitions, or more precisely, their statistical features that we use as detection attributes.

Average Attack Detection Model. The average attack model divides the profile into three partitions: the target item given an extreme rating, the filler items given other ratings (determined based on the attack model), and unrated items. The model essentially just needs to select an item to be the target and all other rated items become fillers. By the definition of the average attack, the filler ratings will be populated such that they closely match the rating average for each filler item. We would expect that a profile generated by an average attack would exhibit a high degree of similarity (low variance) between its ratings and the average ratings for each item except for the single item chosen as the target.

The formalization of this intuition is to iterate through all the highly-rated items, selecting each in turn as the possible target, and then computing the mean variance between the non-target (filler) items and the overall average. Where this metric is minimized, the target item is the one most compatible with the hypothesis of the profile as being generated by an average attack, and the magnitude of the variance is an indicator of how confident we might be with this hypothesis. More formally, we compute $MeanVar$ for a profile P_u twice; once for push attacks, and once for nuke attacks. This metric can be computed as follows, where we define the set of ratings that are potential targets $P_{u,T} = \{i \in P_u, \text{ such that } r_{u,i} = r_{max}\}$ (or r_{min} for nuke attacks.). $P_{u,F}$ is the rest of the profile: $P_u - P_{u,T}$.

$$MeanVar_u = \frac{\sum_{j \in P_{u,F}} (r_{u,j} - \overline{r_u})^2}{|P_{u,F}|}$$

Whichever of the calculations yields the lowest value, we consider this the optimal partitioning for $P_{u,T}$ and $P_{u,F}$, and the value so computed we use as the *Filler Mean Variance* feature for classification purposes. We also compute *Filler Mean Difference*, which is the average of the absolute value of the difference between the user's rating and the mean rating (rather than the squared value as in the variance.)

Finally, in an average attack, we would expect that attack profiles would have very similar within-profile variance: they would have more or less similar ratings for the filler items and an extreme value for the target item. So, our third model-derived feature is *Profile Variance*, simply the variance associated with the profile itself.

Segment Attack Detection Model. For the segment attack model, the partitioning feature that maximizes the attack's effectiveness is the difference in ratings of items in the $P_{u,T}$ set compared to the items in $P_{u,F}$. Thus we introduce the *Filler Mean Target Difference* (FMTD) attribute. The attribute is calculated as follows:

$$FMTD_u = \left| \left(\frac{\sum_{i \in P_{u,T}} r_{u,i}}{|P_{u,T}|} \right) - \left(\frac{\sum_{k \in P_{u,F}} r_{u,k}}{|P_{u,F}|} \right) \right|$$

Target Focus Detection Model. All of the attributes thus far have concentrated on inter-profile statistics; target focus, however, concentrates on intra-profile statistics. Here we are seeking to make use of the fact that often many attack profiles are required to insert the desired bias. Only a substantial attack containing a number of targeted profiles can achieve this result. It is therefore profitable to examine the density of target items across profiles. One of the advantages of the partitioning associated with the model-based attributes described above is that a set of suspected targets are identified for each profile. For our *Target Model Focus* attribute (TMF), we calculate the degree to which the partitioning of a given profile focuses on items common to other attack partitions, and therefore measures a consensus of suspicion regarding each profile. To calculate TMF for a profile, first we define F_i, the degree of focus on a given item, and then select from the profile's target set the item that has the highest focus and use its focus value. Specifically,

$$TMF_u = \max_{j \in P_T} F_j, \text{where}$$

$$F_i = \frac{\sum_{u \in U} \Theta_{u,i}}{\sum_{u \in U} |P_{u,T}|}, \text{and}$$

$$\Theta_{u,i} = \begin{cases} 1, & \text{if } i \in P_{u,T} \\ 0, & \text{otherwise} \end{cases}$$

Although the TMF attribute focuses on model target density; it is easy to see how a similar approach could be used to incorporate other evidence of suspicious profiles for example from time series data or unsupervised detection algorithms. This type of attribute could significantly reduce the impact a malicious user could make by constraining the number of profiles they could inject before they risk the detection of their entire attack effort.

4. METHODOLOGY

The results in this paper were generated using the publicly-available Movie-Lens 100K dataset[2]. This dataset consists of 100,000 ratings on 1682 movies by 943 users. All ratings are integer values between one and five where one is the lowest (disliked) and five is the highest (most liked). Our data includes all the users who have rated at least 20 movies.

The attack detection and response experiments were conducted using a separate training and test set by partitioning the ratings data in half. The first half was used to create training data for the attack detection classifier used in later experiments. For each test the second half of the data was injected with attack profiles and then run through the classifier that had been built on the augmented first half. This approach was used since a typical cross-validation approach would be overly biased since the same movie being attacked would also be the movie being trained for.

For these experiments we use 15 detection attributes:

- 6 generic attributes: WDMA, RDMA, WDA, Length Variance, DegSim (k = 450), and DegSim' (k = 2, d = 963);
- 6 average attack model attributes (3 for push, 3 for nuke): Filler Mean Variance, Filler Mean Difference, Profile Variance;
- 2 segment attack model attributes (1 for push, 1 for nuke): FMTD; and,
- 1 target detection model attribute: TMF.

The training data was created by inserting a mix of the attack models described above for both push and nuke attacks at various filler sizes that ranged from 3% to 100%. Specifically the training data was created by inserting the first attack at a particular filler size, and generating the

[2]http://www.cs.umn.edu/research/GroupLens/data/

detection attributes for the authentic and attack profiles. This process was repeated 18 more times for additional attack models and/or filler sizes, and generating the detection attributes separately. For all these subsequent attacks, the detection attributes of only the attack profiles were then added to the original detection attribute dataset. This approach allowed for a larger attack training set to be created while minimizing over-training for larger attack sizes due to the high percentage of attack profiles that make up the training set (10.5% total across the 19 training attacks).

The segment attack is slightly different from the others in that it focuses on a particular group of items that are similar to each other and likely to be popular among a similar group of users. In our experiments, we have developed several user segments defined by preferences for movies of particular types. In these experiments, we use the Harrison Ford segment (movies with Harrison Ford as a star) as part of the training data and the Horror segment (popular horror movies) for attack testing.

4.1 Generating Recommendations

To generate recommendations, we use the standard kNN collaborative filtering algorithm based on user-to-user similarity [7]. (The scope of this paper precludes consideration of item-based algorithms.) In selecting neighbors, we use Pearson's correlation coefficient for user-user similarities and a neighborhood size $k = 20$. We also filter out all neighbors with a similarity of less than 0.1 to prevent predictions being based on very distant or negative correlations. After identifying a neighborhood, we use the following formula (from [10]) to compute the prediction for a target item i and target user u:

$$p_{u,i} = \bar{r}_u + \frac{\sum_{v \in V} sim_{u,v}(r_{v,i} - \bar{r}_v)}{\sum_{v \in V} |sim_{u,v}|}$$

where V is the set of k similar neighbors to a target user u; $r_{v,i}$ is the rating of i for neighbor v; $\bar{r}_u$ and $\bar{r}_v$ are the average ratings over all rated items for u and v, respectively; and $sim_{u,v}$ is the Pearson correlation.

We incorporate attack detection by following the lead of Chirita, et al. [6] of using a parameter PA_u, the probability that a profile u is an attack profile. Chirita's algorithm has an ad-hoc calculation of PA. They calculate the mean agreement used in the RDMA calculation on the subgroup of profiles whose average similarity over their top 25 neighbors was less than half of the maximum average top 25 similarity in the system. This value is then used as follows:

$$\mathrm{PA}_u = \begin{cases} 0, \text{if} RDMA_u < RDMA_{Avg} \\ \frac{1}{e^{\alpha}-1}(e^{\alpha \frac{RDMA_u - RDMA_{Avg}}{1 - RDMA_{Avg}}} - 1), \text{otherwise} \end{cases}$$

After the attack probability is calculated, it is used to discount the similarity of each profile in the neighborhood calculation. Profiles that are considered likely attackers will therefore be less likely to influence prediction behavior.

$$sim'_{u,v} = sim_{u,v} * (1 - PA_v)$$

where $sim_{u,v}$ is the similarity for profile u and neighbor profile v, and PA_v is the PA for profile v. This revised similarity score is used, instead of the basic Pearson correlation, for neighborhood formation. For our attack classifier, we found that best results were obtained by using a binary classifier and completely eliminating attackers from consideration, effectively setting $PA = 1$ for profiles classified as attackers.

4.2 Evaluation Metrics

There has been considerable research in the area of recommender systems evaluation [8]. Some of these concepts can also be applied to the evaluation of the security of recommender systems, but in evaluating security, we are interested not in raw performance, but rather in the change in performance induced by an attack. Our goal is to measure the effectiveness of an attack - the "win" for the attacker. The desired outcome for the attacker in a "push" attack is of course that the pushed item be more likely to be recommended after the attack than before. In the experiments reported below, we follow the lead of [12] in measuring an algorithm's stability via prediction shift. The prediction shift metric as computed in [1] measures the change in the predicted rating of an item before and after attack. Let $p(u,i)$ be the rating predicted by the system for a given user/item pair, and let $p'(u,i)$ be the prediction for the same pair after the system has been attacked.

$$\mathit{PredShift}(u,i) = p'(u,i) - p(u,i)$$

This quantity is then averaged over all of the test users to arrive at an average prediction shift for that item. These values are averaged once again over all items to derive a measure of the effectiveness of the attack. Note that we do not use the absolute value of the change. A push attack is intended to create a positive change of predicted rating, taking the absolute value of the shift would hide changes that occur in the opposite direction.

For measuring classification performance, we use the standard measurements of precision and recall. Since we are primarily interested in how well the classification algorithms detect attacks, we look at each of these metrics with respect to attack identification. Thus precision is calculated as:

$$\mathit{precision} = \frac{\#\ \mathit{true\ positives}}{(\#\ \mathit{true\ positives} + \#\ \mathit{false\ positives})}$$

$$\mathit{recall} = \frac{\#\ \mathit{true\ positives}}{(\#\ \mathit{true\ positives} + \#\ \mathit{false\ negatives})}$$

where *# true positives* is the number of attack profiles correctly identified as attacks, *# false positives* is the number of authentic profiles that were misclassified as attacks, and *# false negatives* is the number of attack profiles that were misclassified as authentic.

4.3 Experimental Setup

Based on the training data described above, kNN with $k = 9$ was used to make a binary profile classifier with $PA_u = 0$ if classified as *authentic* and $PA_u = 1$ if classified as *attack*. To classify unseen profiles, the k nearest neighbors in the training set are used to determine the class using one over Pearson correlation distance weighting. All segment attack results reflect the average over the 6 combinations of Horror segment movies. Classification results and kNN classifier were created using Weka [16].

In all experiments, to ensure the generality of the results, 50 movies were selected randomly that represented a wide range of average ratings and number of ratings. Each movie was attacked individually and the average is reported for

Table 1: Information gain for the detection attributes against push attacks.

Attribute	Random		Average		Bandwagon		Segment	
	Info Gain	Rank	Info Gain	Rank	Info Gain	Rank	Info Gain	Rank
DegSim (k = 450)	0.161	6	0.116	9	0.180	5	0.180	12
DegSim' (k = 2, d = 963)	0.103	9	0.177	7	0.101	9	0.213	10
WDA	0.233	4	0.229	3	0.234	4	0.246	5
LengthVariance	0.267	1	0.267	1	0.267	1	0.269	3
WDMA	0.248	2	0.238	2	0.248	2	0.229	8
RDMA	0.240	3	0.229	4	0.240	3	0.239	7
FillerMeanDiff*	0.064	13	0.084	13	0.064	13	0.244	6
MeanVar*	0.099	10	0.093	12	0.100	10	0.222	9
ProfileVariance*	0.083	12	0.109	10	0.086	12	0.274	2
FMTD*	0.130	7	0.189	5	0.131	7	0.276	1
FMV*	0.094	11	0.126	8	0.095	11	0.263	4
TMF*	0.194	5	0.185	6	0.174	6	0.176	13

Table 2: Information gain for the detection attributes against nuke attacks.

Attribute	Random		Average		Love/Hate	
	Info Gain	Rank	Info Gain	Rank	Info Gain	Rank
DegSim (k = 450)	0.161	6	0.111	10	0.155	11
DegSim' (k = 2, d = 963)	0.104	10	0.176	5	0.213	9
WDA	0.234	4	0.229	4	0.253	5
LengthVariance	0.267	1	0.267	1	0.267	3
WDMA	0.248	2	0.238	2	0.244	8
RDMA	0.240	3	0.229	3	0.249	7
FillerMeanDiff*	0.084	12	0.094	12	0.249	6
MeanVar*	0.109	9	0.103	11	0.200	10
ProfileVariance*	0.095	11	0.121	8	0.095	12
FMTD*	0.138	8	0.154	7	0.276	1
FMV*	0.077	13	0.069	13	0.276	1
TMF*	0.190	5	0.162	6	0.267	4

all experiments. For prediction shift experiments, we used a neighborhood size of $k = 20$ in the k-nearest-neighbor algorithm, and a sample of 50 users mirroring the overall distribution of users in terms of number of movies seen and ratings provided. The results reported below represent averages over the combinations of test users and test movies.

The Chirita et al. algorithm was also implemented for comparison purposes (with $\alpha = 10$), and run on the test set described above. Comparative results are shown in the next section. It should be noted that there are a number of methodological differences between the results reported in [6] and those shown here. The attack profiles used in [6] used 100% filler size and targeted 3 items simultaneously. In the experiments below, we concentrate on a single item and vary filler size. Also their results were limited to target movies with low average ratings and few ratings, the 50 movies we have selected represent both a wider range of average ratings and variance in rating density.

5. EXPERIMENTAL RESULTS

Information Gain Analysis

Below we present a detailed analysis of the information gain associated with the attributes discussed above. As our results below show, the information gain varies significantly across several dimensions. First we present the information gain associated with each attribute across attack models. This is followed by an analysis of the effect of filler size and attack size on the information gain of an attribute.

For our experiments each attack was inserted targeting a single movie at 5% attack size and a specific filler size. Each of the test movies was attacked at filler sizes of 3%, 5%, 10%, 20%, 40%, 60%, 80%, and 100% and the results reported are averaged over the 50 test movies and the 8 filler sizes.

Table 1 shows the average information gain (info gain) for each attribute, and its relative rank for each of the push attacks described above. The model-specific attributes shown (indicated with an '*'), were created to look for push attacks. As the results show, the LengthVar attribute is very important for distinguishing attack profiles, since few real users rate more than a small percentage of the items. The attributes with the next highest gain for average, random, and bandwagon attack are those using the "deviation from mean agreement" concept from [6]: WDMA, RDMA, and WDA. For segment attack, however, the model specific attribute FMTD, which captures the mean rating difference between the target and filler items, is the most informative. The next most informative is profile variance, which follows intuition since segment attack gives all items the same rating except the segment items. Interestingly, TMF, which uses our crude measure of which items are under attack, also has strong information gain. This suggests that further improvements in detecting likely attack targets could yield even better detection results.

The results displayed in Table 2 were obtained following the same methodology, but the model-specific attributes (indicated with an '*') were created to look for nuke attacks. The table depicts the average information gain for each attribute, and its relative rank for each of the nuke attacks described above. For average and random attacks, the relative information gain of the attributes is pretty consistent between push and nuke attacks. A closer inspection

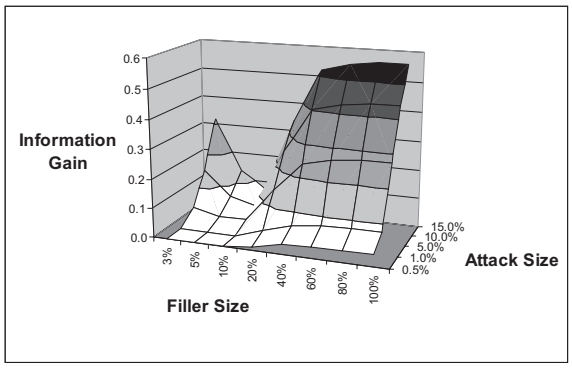

Figure 2: DegSim ($k = 450$) Information gain vs. filler size and attack size for random push attacks.

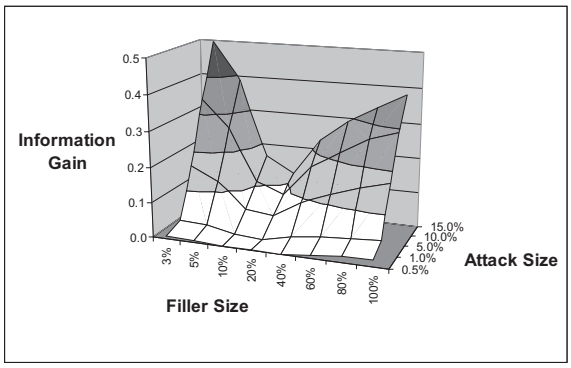

Figure 3: DegSim ($k = 450$) Information gain vs. filler size and attack size for average push attacks.

of the information gain reveals that the informativeness of the generic attributes is nearly identical. For the model based attributes, the FillerMeanDiff, MeanVar, and ProfileVariance (single target) attributes all become slightly more informative, whereas the FMTD and FMV (multiple target) attributes generally become less informative. Conceptually the reason this occurs is due in part to the distribution characteristics of the data. In our dataset, the distribution of ratings is such that there are more high ratings than low ratings as the system mean of a 3.6 rating reflects. Since the information gain of the single target attributes improves with correct selection of the actual target item, for the average and random model-specific attributes, the probability of identifying the correct target item increases. On the other hand, since the group attack model-specific attributes select all items with the user's minimum rating as the suspected target, the models also mask some of the more extreme variability in real user ratings thus decreasing the information gain for detecting nuke attacks targeting a single item.

The attribute information gain shows some interesting differences between the love/hate attack and the other nuke attacks. Due to the simplicity of this attack, a single minimum rating and all other ratings given the system maximum, it is not surprising that the variance and similarity based attributes are far more informative. The most similar attack is the segment attack without the addition of the target segment (I_S). A comparison of the information gain of the attributes detecting the segment push attack and the love/hate nuke attack reflects this intuition with the only major differences occurring in the ProfileVariance, and TargetModelFocus attributes. ProfileVariance becomes less informative due primarily to the same rating distribution reasons given earlier. The TargetModelFocus attribute on the other hand becomes more informative since it is very easy to identify the actual nuke attack target with these profiles.

To understand the informativeness of the attributes in greater depth, we experimented with the dimensions of filler size and attack size on the information gain of each attribute for each attack model. For this set of experiments, the 50 test movies were attacked at each combination of filler sizes of 3%, 5%, 10%, 20%, 40%, 60%, 80%, and 100% and attack sizes of .5%, 1%, 5%, 10%, and 15%. Figures 2 and 3 show the information gain of the DegSim attribute using $k = 450$ across the dimensions of filler size and attack size,

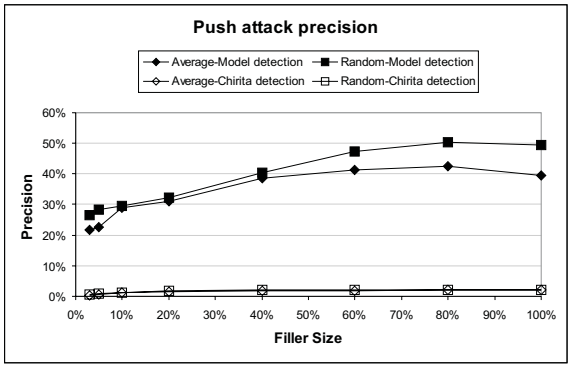

Figure 4: Precision for both classifiers against 1% average and random attacks.

thus creating an information gain surface for random and average push attacks. While the tables above provide some insight into the benefit of each of these attributes across attack models, the actual information gain of each attribute will vary greatly based on filler size and attack size as well. Furthermore, the range of filler sizes in which an attribute is most informative may also differ across attack models. A similar analysis was performed for all of the attributes and all of the models (results not included due to space constraints).

Classification Performance Analysis

Figures 4 and 5 compare the detection capabilities of our algorithm using model-specific features with the Chirita algorithm for the basic attacks: random and average. For both precision and recall, the model-specific algorithm is dominant. Precision is particularly a problem for the Chirita algorithm: many false positive identifications are made. However, as the authors point out, this is probably not too significant since discarding a few real users will not generally impact the system's recommendation performance, and our experiments showed that generally this was true. We also see that the model-specific version has better recall especially a low filler sizes: recall that the Chirita algorithm was tuned for 100% filler sizes, so this is not surprising.

Figures 6 and 7 extend these results to examine the bandwagon and segment attacks. Again a similar pattern is seen. Precision is a bit lower, especially for the segment attack, but recall is extremely high for the model-specific algorithm.

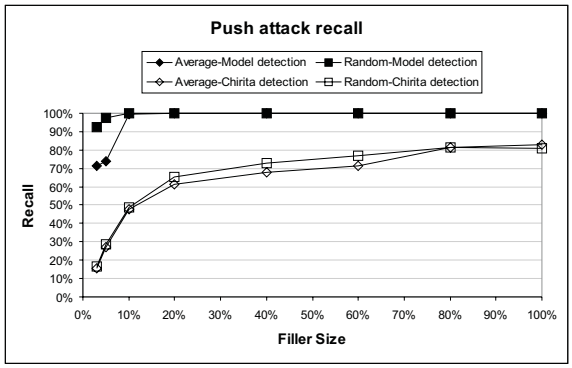

Figure 5: Recall for both classifiers against 1% average and random attacks.

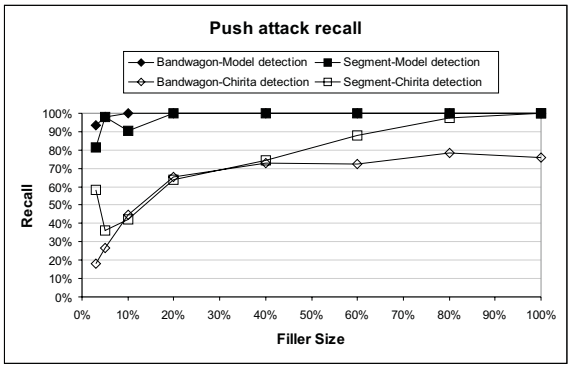

Figure 7: Recall for both classifiers against 1% bandwagon and segment attacks.

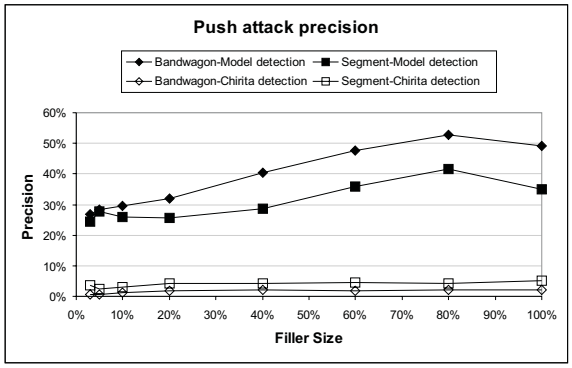

Figure 6: Precision for both classifiers against 1% bandwagon and segment attacks.

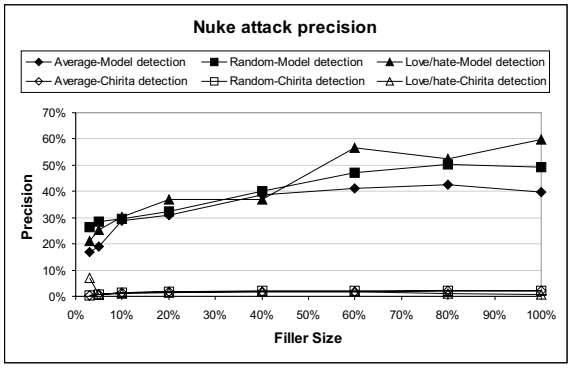

Figure 8: Precision and recall for both classifiers against 1% nuke attacks.

Chirita again suffers at low filler sizes. There is an interesting dip at 3% filler size. This occurs because the average number of user ratings is around this number. The $LengthVar$ attribute is not useful at this point because the attack profiles do not differ in length from a typical user.

Nuke attack results are shown in Figures 8 and 9. Three attacks are shown: average, random and love/hate. Again, precision is low for the Chirita algorithm and recall results are similar to those seen for the push attacks, except that the love/hate attack proves to be difficult for Chirita to detect at high filler sizes.

Robustness Analysis

The results above show that both detection schemes have some success in identifying attack profiles. While this is promising, the real proof of the concept is its impact on the recommender system itself. Can using a detection algorithm reduce the impact of an attack, forming a successful defense?

We measure the robustness of the system with the *Prediction Shift* metric discussed above on the recommendation algorithm as described in Section 4.1. We used the troublesome 3% filler size to maximize the difficulty of detection and varied the attack size, expressed here as a percentage of the original profile database: a 1% attack equals 9 attack profiles inserted into the database.

Figure 10 shows the average prediction shift for the recommendation algorithm, with and without attack profile detection, for both our model-specific as well as Chirita detection methods. This figure shows the results for only the average and random attacks. Lower prediction shifts are better: they mean that the system is more robust. Note that the unaided system responds quickly as the attacks get larger. At 5% attack, the attacked item is already rated 1.4 points higher. For the MovieLens data, this is a shift that could take an item which previously would have gotten a middling predicted rating (3.6 is the overall system average for all movies) all the way to the maximum possible predicted rating of 5. The Chirita detection algorithm, despite its lower recall in the detection experiments, still has a big impact on system robustness against the average attack, cutting the prediction shift by half or better for low attack sizes. It does less well for the random attack, for which it was not designed. Our classification approach improves on Chirita except at the very largest attack sizes. At these sizes, the attack profiles begin to alter the statistical properties of the ratings corpus, so that the attacks start to look "normal."

Figure 11 continues this analysis to the bandwagon and segment attacks. We see how significant the threat posed by the segment attack is here. At very low attack sizes, it is already having an impact equivalent to the average attack at 5% attack size. At these lower sizes, the model-based approach is actually inferior to Chirita. At 3%, the targeted attributes kick in and the segment attack is virtually neutralized until it becomes very large. Chirita shows more or less the same pattern as against the random attack.

Finally, Figure 12 shows prediction shift results with the nuke attack. The low-knowledge love/hate attack is quite

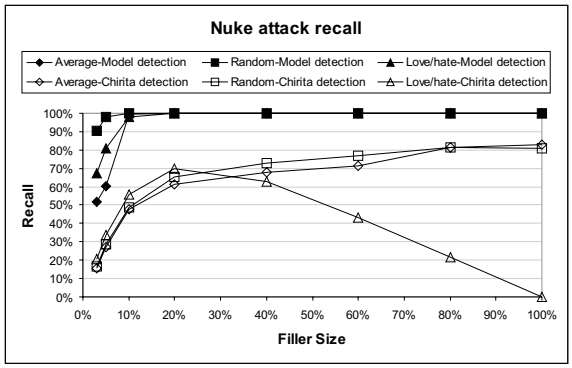

Figure 9: Precision and recall for both classifiers against 1% nuke attacks.

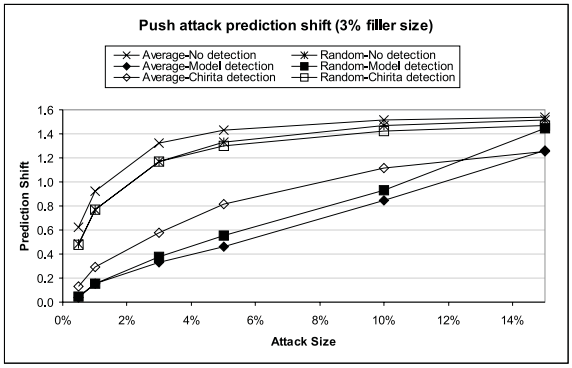

Figure 10: Prediction shift for the recommender system vs. average and random attacks with and without attack detection.

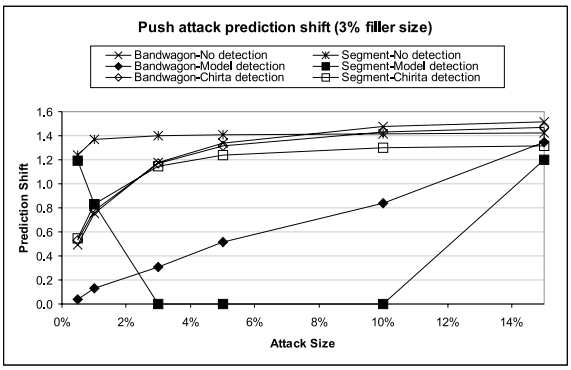

Figure 11: Prediction shift for the recommender system vs. bandwagon and segment attacks before and after attack detection.

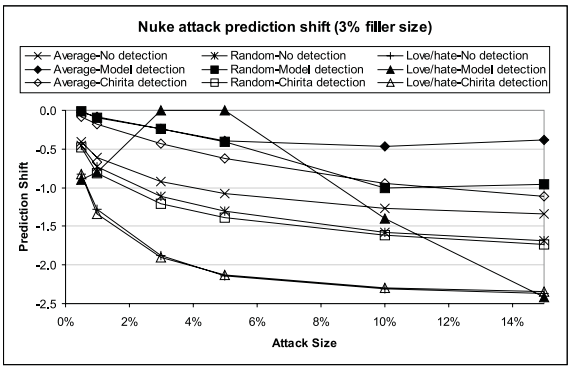

Figure 12: Prediction shift for the recommender system vs. nuke attacks before and after detection.

effective, almost as good as the average attack. Either of these attacks can reduce the predicted score of a highly-favored item (5.0 prediction) all the way to below the mean, with just a 3% attack size. A similar pattern is seen as in the previous results. The model-based approach does very well at defending against average and random attacks. It does less well with the love/hate attack, for which it must be said it has no model-specific features. Chirita again is somewhere in the middle, doing better against the love/hate attack at low and very high attack sizes, but not elsewhere.

Although not necessary for the detection model described here, we also experimented with the use of the detection attributes to identify the type of attack associated with a profile. It may be possible to use this information in weighting attributes per profile based on model suspicion potentially further improving classifier performance.

For this experiment, the same training set was used, but the attack label was replaced with the type of attack. A C4.5 classifier [14] was constructed using the Weka data mining package [16] which resulted in the tree shown in Figure 13. The identification nodes are labeled with (*# of instances classified / # of instances misclassified*. This tree was able to correctly classify instances as either Authentic, Average attack, Random attack, Bandwagon attack, Segment attack, or Love/Hate attack with 97.38% accuracy using 10 times cross-validation on the training set. The majority of misclassifications came from bandwagon attack being misclassified as random attack and vice-versa, not surprising since the random attack is a special case of the bandwagon attack.

6. CONCLUSION

Profile injection attacks are a serious threat to the robustness and trustworthiness of collaborative recommender systems and other open adaptive systems. An essential component of a robust recommender system is a mechanism to detect profiles originating from attacks so that they can be quarantined and their impact reduced.

In this paper, we have demonstrated that the information gain associated with detection attributes can vary greatly not just across attack types, but also based on filler size and attack size. In addition, we show that classifiers built using these features can be quire effective in detecting attacks, and that this detection can improve the stability of the recommender, keeping its predicted ratings steady under most attack scenarios. The segment and love/hate attack models prove to be the wiliest opponents. They can strongly impact prediction even at low attack sizes and it is precisely at these small sizes that they are difficult to detect. We are continuing to study the problem of detection for these attack types.

7. REFERENCES

[1] R. Burke, B. Mobasher, and R. Bhaumik. Limited knowledge shilling attacks in collaborative filtering

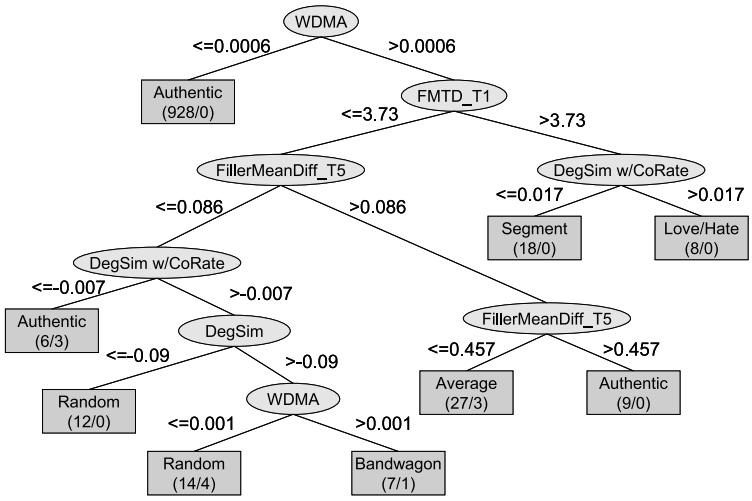

Figure 13: C4.5 decision tree used to identify attack type. T1 and T5 signify the nuke and push respectively.

systems. In *Proceedings of the 3rd IJCAI Workshop in Intelligent Techniques for Personalization*, Edinburgh, Scotland, August 2005.

[2] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik. Segment-based injection attacks against collaborative filtering recommender systems. In *Proceedings of the International Conference on Data Mining (ICDM 2005)*, Houston, December 2005.

[3] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik. Classification features for attack detection in collaborative recommender systems. In *To appear in Proceedings of The Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006)*, Philadelphia, PA, August 2006.

[4] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik. Detecting profile injection attacks in collaborative recommender systems. In *To appear in Proceedings of the IEEE Joint Conference on E-Commerce Technology and Enterprise Computing, E-Commerce and E-Services (CEC/EEE 2006)*, Palo Alto, CA, June 2006.

[5] R. Burke, B. Mobasher, R. Zabicki, and R. Bhaumik. Identifying attack models for secure recommendation. In *Beyond Personalization: A Workshop on the Next Generation of Recommender Systems*, San Diego, California, January 2005.

[6] Paul-Alexandru Chirita, Wolfgang Nejdl, and Cristian Zamfir. Preventing shilling attacks in online recommender systems. In *WIDM '05: Proceedings of the 7th annual ACM international workshop on Web information and data management*, pages 67–74, New York, NY, USA, 2005. ACM Press.

[7] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd ACM Conference on Research and Development in Information Retrieval (SIGIR'99)*, Berkeley, CA, August 1999.

[8] J. Herlocker, J. Konstan, L. G. Tervin, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.

[9] S. Lam and J. Reidl. Shilling recommender systems for fun and profit. In *Proceedings of the 13th International WWW Conference*, New York, May 2004.

[10] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams. Effective attack models for shilling item-based collaborative filtering systems. In *Proceedings of the 2005 WebKDD Workshop, held in conjuction with ACM SIGKDD'2005*, Chicago, Illinois, August 2005.

[11] B. Mobasher, R. Burke, C. Williams, and R. Bhaumik. Analysis and detection of segment-focused attacks against collaborative recommendation. In *To appear in Lecture Notes in Computer Science: Proceedings of the 2005 WebKDD Workshop*. Springer, 2006.

[12] M. O'Mahony, N. Hurley, N. Kushmerick, and G. Silvestre. Collaborative recommendation: A robustness analysis. *ACM Transactions on Internet Technology*, 4(4):344–377, 2004.

[13] M.P. O'Mahony, N.J. Hurley, and G. Silvestre. Utility-based neighbourhood formation for efficient and robust collaborative filtering. In *Proceedings of the 5th ACM Conference on Electronic Commerce (EC 04)*, pages 260–261, May 2004.

[14] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[15] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM Press, 1994.

[16] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques, 2nd Edition*. Morgan Kaufmann, San Francisco, CA, 2005.

[17] Hua-Jun Zeng 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*, May 2005.