

# Classification Features for Attack Detection in Collaborative Recommender Systems \*

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

## ABSTRACT

Collaborative recommender systems are highly vulnerable to attack. Attackers can use automated means to inject a large number of biased profiles into such a system, resulting in recommendations that favor or disfavor given items. Since collaborative recommender systems must be open to user input, it is difficult to design a system that cannot be so attacked. Researchers studying robust recommendation have therefore begun to identify types of attacks and study mechanisms for recognizing and defeating them. In this paper, we propose and study different attributes derived from user profiles for their utility in attack detection. We show that a machine learning classification approach that includes attributes derived from attack models is more successful than more generalized detection algorithms previously studied.

## Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval; H.3.4 [Information Storage and Retrieval]: Systems and Software; I.2.6 [Artificial Intelligence]: Learning

## General Terms

Experimentation, Algorithms, Security

## Keywords

collaborative filtering, recommender systems, robustness, attack detection

## 1. INTRODUCTION

Research has established the vulnerabilities of recommender systems using collaborative filtering techniques, in the face of what has been termed “shilling” or “profile injection”

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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

KDD'06, August 20–23, 2006, Philadelphia, Pennsylvania, USA.  
Copyright 2006 ACM 1-59593-339-5/06/0008 ...\$5.00.

attacks in which a malicious user enters biased profiles in order to influence the system’s behavior [4, 1, 6, 9].

Recent work in this area has focused on detecting and preventing profile injection attacks. Chirita et al. [5] proposed several metrics for analyzing rating patterns of malicious users and evaluate their potential for detecting such attacks. Su, et al. [13] developed a spreading similarity algorithm in order to detect groups of similar attackers. O’Mahony et al. [10] developed several techniques to defend against the attacks described in [6] and [9], including new strategies for neighborhood selection and similarity weight transformations. In our prior work we introduced the attack model-specific approach to profile classification that is more fully explored here. [3, 8].

To detect attacks via pattern classification, known attack models are used to build a training set where standard data mining techniques are used to build a classifier. With such an approach, the closer an attacker mimics 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 are 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 the construction of 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 particular, we are investigating a simple nearest-neighbor classification using kNN. We compare our model-based approach with that described in [5] and demonstrate improved performance, especially for smaller, more difficult to detect, attacks.

## 2. ATTACK MODELS

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* [9].

An attack model is an approach to constructing attack profiles based on knowledge of the recommender system, its

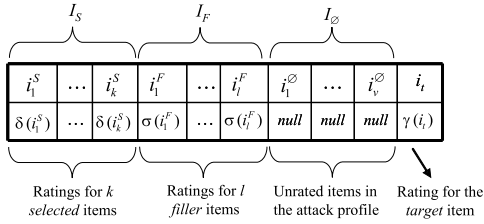


Figure 1: The general form of an attack profile.

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, where  $m$  is the total number of items in the system. The profile is partitioned in four parts. The null partition,  $I_0$ , are those items with no ratings in the profile. The single target item  $i_t$  will be given a rating as determined by the function  $\gamma$ , 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  $\delta$ . Finally, there is a set of filler items  $I_F$  whose ratings are added as specified by the function  $\sigma$ . It is the strategy for selecting items in  $I_S$  and  $I_F$  and the functions  $\gamma$ ,  $\sigma$ , and  $\delta$  that define an attack model and give it its character.

Two basic attack models, introduced originally in [6] 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  $\sigma$  generates random ratings centered around 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 [4, 1, 2] for additional details.

The *bandwagon attack* is similar to the random attack, but it uses some additional knowledge, namely the identification of a few of the most popular items in a particular domain: blockbuster movies, for example. This information is easy to obtain and not system-dependent. 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 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 make them similar to the target item, also rated highly in a push attack, and the filler items are given low ratings, making them different from 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, since the dislikes of a market segment are more dispersed than its preferences. Our final attack model, the *love/hate attack* is simple but nonetheless effective attack against both item-based and user-based algorithms. It associates a low rating with the target item and high ratings with the filler items  $I_F$ .

### 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 using 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 characteristics that 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 our attack models.

#### 3.1 Generic Attributes

We expect the overall statistical signature of attack profiles will differ significantly from that of authentic profiles. This difference comes from two sources: the rating given the target item, and the distribution of ratings among the filler items. As many researchers in the area have theorized [6, 5, 9, 7], 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 statistically from those of 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. [5] 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 places higher weight on rating deviations for sparse items. We have found this variant 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  $\bar{r}_i$  be the average of these ratings.

The WDMA attribute can be computed as follows:

$$WDMA_u = \frac{\sum_{i=0}^{n_u} \frac{|r_{u,i} - \bar{r}_i|}{l_i^2}}{n_u}$$

The second variation of the RDMA measure which we call *Weighted Degree of Agreement* (WDA) uses only the numerator of the RDMA equation, capturing the sum of the differences of the profile’s ratings from the item’s average rating divided by the item’s rating frequency. It is computed as follows:

$$WDA_u = \sum_{i=0}^{n_u} \frac{|r_{u,i} - \bar{r}_i|}{l_i}$$

This .

In addition to rating deviations, some researchers have hypothesised that attack profiles are likely to have a higher similarity with their top 25 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 [5, 11]. This hypothesis was confirmed in our earlier experiment, 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 [5].

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 the 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 a 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. We capture this idea with the measure *Length Variance* (LengthVar) that is a measure of how much the length of a given profile varies from the average length in the database.

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

where  $\bar{n}_u$  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, so we seek to augment the generic attributes with some that are designed specifically to match the characteristics of our attack models.

As shown in Section 2, attacks can be characterized based on the features of their 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 the 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. 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 selects 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 minimum, 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, then, we compute *MeanVar* for a profile  $P_u$  as follows. First 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:  $P_u - P_{u,T}$ .

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

We compute *MeanVar* twice, once where  $P_{u,T}$  contains items given the maximum rating, and once where  $P_{u,T}$  contains items given the minimum rating. 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 com-

puted 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 a single profile cannot really influence the recommender system. 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 is 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<sup>1</sup>. 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

<sup>1</sup><http://www.cs.umn.edu/research/GroupLens/data/>

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 as the same movie being attacked would also be the movie being trained for.

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 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 combined with the average attribute normalizing factor described above, allowed a larger attack training set to be created while minimizing over-training for larger attack sizes (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.

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:

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

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

where  $\# \text{ true positives}$  is the number of attack profiles correctly identified as attacks,  $\# \text{ false positives}$  is the number of authentic profiles that were misclassified, and  $\# \text{ false negatives}$  is the number of attack profiles that were misclassified.

Based on the training data described above,  $k$ NN 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  $k$ NN classifier were created using Weka [12].

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 of these movies was attacked individually and the average is

reported for all experiments.

The Chirita et al. algorithm was also implemented for comparison purposes (with  $\alpha = 10$ ), and run on the test set described above. It computes the probability that a profile  $u$  is an attack profile ( $PA_u$ ) using an ad-hoc calculation tied to how much a profile’s RDMA exceeds the system average. In the comparative results shown in the next section, it should be noted that there are a number of methodological differences between the results reported in [5] and those shown here. The attack profiles in [5] used 100% filler size and targeted 3 items simultaneously. 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, while the 50 movies we have selected represent both a wider range of average ratings and variance in rating density.

## 5. EXPERIMENTAL RESULTS

Table 1: Information gain for detection attributes.

Info. Gain	Average Rank	Attribute
$0.358 \pm 0.003$	$1 \pm 0$	WDMA
$0.33 \pm 0.008$	$2 \pm 0$	RDMA
$0.252 \pm 0.005$	$3.6 \pm 0.49$	WDA
$0.24 \pm 0.033$	$4.5 \pm 1.86$	LengthVar
$0.233 \pm 0.018$	$4.7 \pm 1$	TMF
$0.213 \pm 0.01$	$5.6 \pm 0.49$	MeanVar (nuke)
$0.197 \pm 0.005$	$6.8 \pm 0.4$	FMTD (nuke)
$0.184 \pm 0.004$	$8.3 \pm 0.78$	FMTD (push)
$0.185 \pm 0.008$	$8.5 \pm 0.5$	MeanVar (push)
$0.144 \pm 0.006$	$10 \pm 0$	FillerMeanDiff (nuke)
$0.117 \pm 0.007$	$11 \pm 0$	FillerMeanDiff (push)
$0.097 \pm 0.004$	$12.2 \pm 0.4$	DegSim <sup>7</sup>
$0.086 \pm 0.009$	$12.8 \pm 0.4$	DegSim
$0.069 \pm 0.004$	$14 \pm 0$	ProfileVariance (nuke)
$0.048 \pm 0.003$	$15 \pm 0$	ProfileVariance (push)

Table 1 shows the attributes described in the previous section and the information gain calculated for each attribute over the training data. The attributes with the highest gain are those using the “deviation from mean agreement” concept from [5]: WDMA, RDMA, and WDA. The different measures are actually useful at different filler sizes, so none really subsumes the others. The LengthVar attribute is very important for distinguishing high filler sizes, since few real users rate anything close to 100% of the available 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.

Figures 2 and 3 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 4 and 5 extend these results to examine the bandwagon and segment attacks. Again a similar pattern is seen.

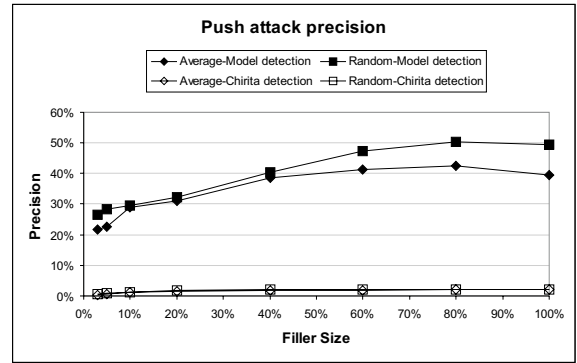


Figure 2: Classifier precision against 1% average and random attacks.

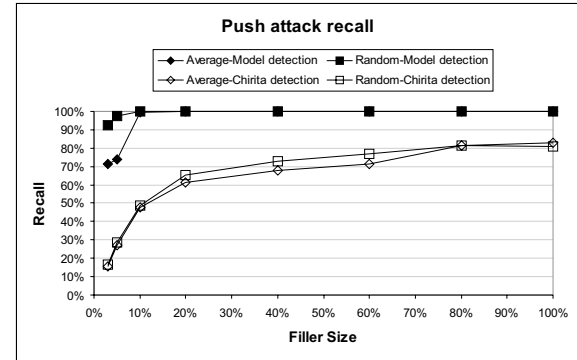


Figure 3: Classifier recall against 1% average and random attacks.

Precision is a bit lower, especially for the segment attack, but recall is extremely high for the model-specific algorithm. Chirita again suffers at low filler sizes. There is an interesting dip at 3% filler size, around the average number of ratings per user. 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 6 and 7. 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.

## 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 demonstrate a classification approach to attack detection, introducing a number of detection features based on attack models. We show that classifiers built using these features can detect attacks well to help improve the stability of a recommender under most attack scenarios. The segment and love/hate attacks prove to be the most wily opponents. They are the most effective at avoiding detection particularly at low filler sizes. We are continuing to study the problem of detection for these attacks.

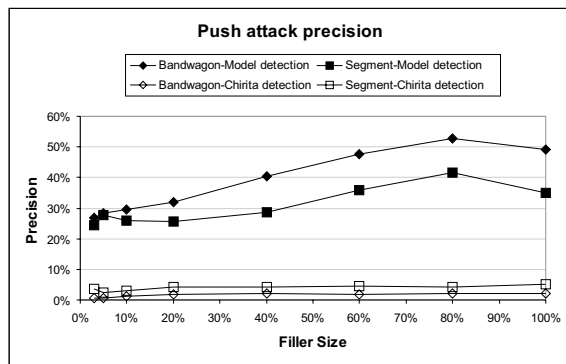


Figure 4: Classifier precision against 1% bandwagon and segment attacks.

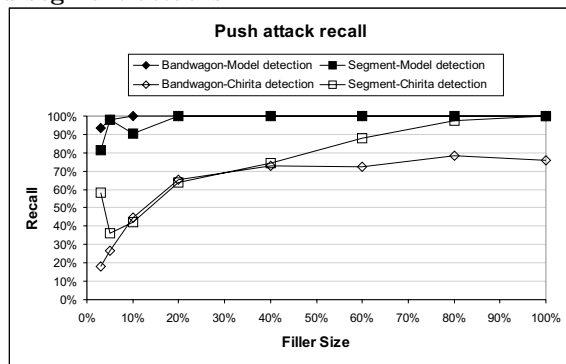


Figure 5: Classifier recall against 1% bandwagon and segment attacks.

## 7. REFERENCES

- [1] R. Burke, B. Mobasher, and R. Bhaumik. Limited knowledge shilling attacks in collaborative filtering systems. In *Proc. 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 *Proc. of the Int'l Conference on Data Mining (ICDM 2005)*, Houston, December 2005.
- [3] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik. Detecting profile injection attacks in collaborative recommender systems. To appear in *Proc. 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.
- [4] 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, CA, January 2005.
- [5] P. Chirita, W. Nejdl, and C. Zamfir. Preventing shilling attacks in online recommender systems. In *WIDM '05: Proc. of the 7th annual ACM Int'l workshop on Web information and data management*, pages 67–74, New York, NY, 2005. ACM Press.
- [6] S. Lam and J. Reidl. Shilling recommender systems for fun and profit. In *Proc. of the 13th Int'l WWW Conference*, New York, May 2004.

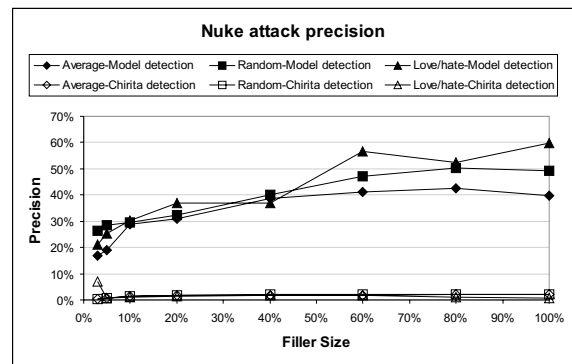


Figure 6: Classifier precision against 1% nuke attacks.

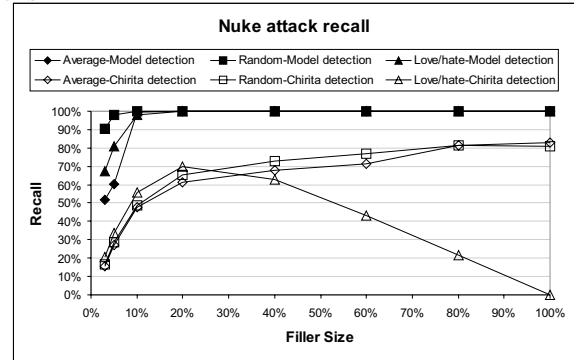


Figure 7: Classifier recall against 1% nuke attacks.

- [7] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams. Effective attack models for shilling item-based collaborative filtering systems. In *Proc. of the 2005 WebKDD Workshop*, Chicago, IL, August 2005.
- [8] B. Mobasher, R. Burke, C. Williams, and R. Bhaumik. Analysis and detection of segment-focused attacks against collaborative recommendation. To appear in *Lecture Notes in Computer Science: Proceedings of the 2005 WebKDD Workshop*. Springer, 2006.
- [9] 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.
- [10] M.P. OMahony, N.J. Hurley, and G. Silvestre. Utility-based neighbourhood formation for efficient and robust collaborative filtering. In *Proc. of the 5th ACM Conference on Electronic Commerce (EC04)*, pages 260–261, May 2004.
- [11] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proc. of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM Press, 1994.
- [12] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques, 2nd Edition*. Morgan Kaufmann, San Francisco, CA, 2005.
- [13] H. Zeng X. Su and Z. Chen. Finding group shilling in recommendation system. In *WWW 05 Proc. of the 14th international conference on World Wide Web*, May 2005.