Profile Injection Attack Detection for Securing Collaborative Recommender Systems [1]

Chad Williams
Research Advisor: Bamshad Mobasher
Center for Web Intelligence, DePaul University
School of Computer Science, Telecommunication, and Information Systems
Chicago, Illinois, USA
{cwilli43, mobasher}@cs.depaul.edu

Researchers have shown that collaborative recommender systems, the most common type of web personalization system, 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 study mechanisms for recognizing and defeating attacks. In prior work, we have introduced a variety of attributes designed to detect profile injection attacks and evaluated their combined classification performance against several well studied attack models using supervised classification techniques. In this paper, we propose and study the impact the dimensions of attack type, attack intent, filler size, and attack size have on the effectiveness of such a detection scheme. We conclude by experimentally exploring the weaknesses of a detection scheme based on supervised classification, and techniques that can be combined with this approach to address these vulnerabilities.

Key Words and Phrases: Attack Detection, Attack Models, Bias Profile Injection, Collaborative Filtering, Pattern Recognition, Profile Classification, Recommender Systems, Shilling

Dedication

This dissertation is dedicated to four people who shared with me the sacrifices required to complete it. The first is my wife, Patricia Boye-Williams, who shared in my struggles of trying to be a full-time student, husband and father. Without her emotional support and encouragement the completion of this project would not have been possible. The second is my daughter, Grace Boye-Williams, who is growing up into a wonderful little person before my eyes. Last, I would like to thank my parents, KC and Theresa Williams, for their nurturing and support throughout my life.

Acknowledgments

I owe thanks to many people, whose assistance was indispensable in completing this project. First among these are Bamshad Mobasher and Robin Burke, whose mentoring, expertise, insight, and patience have been invaluable throughout this project. I particularly owe gratitude to Bamshad Mobasher for inspiring and encouraging me to pursue an academic career in computer science. His thoroughness and promptness in reviewing my work in progress was crucial to both my development and the success of this project. Also I would like to thank Runa Bhaumik and JJ Sandvig for their contributions through our discussions in our research group. Finally, Jami Montgomery, my academic advisor, for his open feedback and suggestion to contact Bamshad Mobasher about research opportunities in the first place.

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

Contents

1. INTRODUCTION

Significant vulnerabilities have been exposed in collaborative filtering recommender systems to what have been termed "shilling" or "profile injection" attacks in which a malicious user enters biased profiles in order to influence the system's behavior [Burke et al. 2005; Burke et al. 2005; Lam and Reidl 2004; O'Mahony et al. 2004]. We use the more descriptive phrase "profile injection attacks", since promoting a particular product is only one way such an attack might be used. In a profile injection attack, an attacker interacts with a collaborative recommender system to build within it a number of profiles associated with fictitious identities with the aim of biasing the system's output. A collaborative recommender using any of the common algorithms can be exploited by attackers without a great degree of knowledge of the system. Related work has established that hybrid and model-based recommendation offers a strong defense against profile injection attacks, reduces the impact of attacks from serious body blows to the system's integrity to mere annoyances for the most part [Mobasher et al. 2006]. Such approaches should be seriously considered by implementers interested in robustness and capable of deploying them.

However, even the most robust of the systems studied have not been unaffected by profile injection attacks, and no collaborative system could be. As long as new profiles are accepted into the system and allowed to affect its output; it is possible for an attacker to perform these types of manipulations. One common defense is to simply make assembling a profile more difficult. A system may require that users create an account and perhaps respond to a captcha[2] before doing so. This increases the cost of creating bogus accounts (although with offshore data entry outsourcing available at low rates, the cost may still not be too high for some attackers). However, such measures come at a high cost for the system owner as well – they drive users away from participating in collaborative systems, systems which need user input as their life blood. In addition, such measures are totally ineffective for recommender systems based on implicit measures such as usage data mined from web logs.

There have been some recent research efforts aimed at detecting and reducing the effects of profile injection attacks. Several metrics for analyzing rating patterns of malicious users and algorithms designed specifically for detecting such attack profiles have been introduced [Chirita et al. 2005]. Other work introduced a spreading similarity algorithm that detected groups of very similar attackers when applied to a simplified attack scenario [Xue-Feng Su and Chen. 2005]. O'Mahony et al. [2004] developed several techniques to defend against the attacks described in [Lam and Reidl 2004] and [O'Mahony et al. 2004], including new strategies for neighborhood selection and similarity weight transformations. Massa and Avesani [2004] introduced a trust network approach to limit the influence of biased users. We are developing a multi-strategy approach to attack defense, including supervised and unsupervised classification approaches, time-series analysis, vulnerability analysis, and anomaly detection.

Profile classification means identifying suspicious profiles and discounting their contribution toward predictions. The success of such an approach is entirely dependent on the definition of a "suspicious" profile. In this paper, we examine approaches that detect attacks conforming to known attack models, those we have discussed above. Of course, nothing compels an attacker to produce profiles that have these characteristics. However, the attacks outlined above work well because they were created by reverse engineering the recommendation algorithms. Attacks that deviate from these patterns are therefore likely to be less effective than those that conform to them. If we can reliably detect attacks that conform to our models of effective attacks, then attackers will have to use attacks of lower effectiveness. Such attacks will have to be larger to achieve a given impact, and large attacks of any type are inherently more detectable through other techniques such as time series analysis. In Section 7 we explore this assumption and examine techniques that can be used to address this vulnerability. Through these combined techniques, we hope to minimize the potential harm profile injection can cause.

[2]http://www.captcha.net/

The primary aim of this work is to improve the robustness of collaborative recommendation to profile injection attacks by identifying and discounting attack profiles. This study focuses on extending the research communities understanding of how the dimensions of attack type, filler size, and attack size effect profile detection. In Section 2 we provide some background on the attack dimensions that will be examined throughout our experiments, and give a narrative example to motivate the need for detecting biased profile attacks. In Section 3 the attack models examined in our experiments are explained in detail. Section 4 summaries the attributes we have introduced in prior work and the intuition behind each of their design. This is followed by a detailed description of our experimental methodology and results in Sections 5 and 6. In Section 7 we explore the weaknesses of a supervised approach to attack detection, and examine other techniques than can be used in conjunction with such an approach to provide greater protection. Finally, in Section 8 we summarize the discoveries found in this work and suggest areas for future investigation.

2. BACKGROUND AND MOTIVATION

In this paper we consider attacks where the attacker's aim is to introduce a bias into a recommender system by injecting fake user ratings. These type of attacks has been termed "shilling" attacks [Burke et al. 2005; Lam and Reidl 2004; O'Mahony et al. 2004]. We prefer the phrase *profile injection attacks*, since promoting a particular product is only one way such attacks might be used. In a profile injection attack, an attacker interacts with the recommender system to build within it a number of profiles with the aim of biasing the system's output. Such profiles will be associated with fictitious identities to disguise their true source.

Collaborative recommenders have been shown to have significant vulnerabilities to profile injection attacks. Researchers have explored alternate algorithms and techniques for increasing the robustness of collaborative recommendation [Mobasher et al. 2006]. While these solutions offer significant gains in robustness, they are all still inherently vulnerable due to the open nature of collaborative filtering.

In this section, we present some of the dimensions across which such attacks must be analyzed, and discuss the basic concepts and issues that motivate our analysis of detection techniques in the rest of the paper.

2.1 Attack Dimensions

Profile injection attacks can be categorized based on the knowledge required by the attacker to mount the attack, the intent of a particular attack, and the size of the attack. From the perspective of the attacker, the best attack against a system is one that yields the biggest impact for the least amount of effort. While the knowledge and effort required for an attack is an important aspect to consider, from a detection perspective, we are more interested in how these factors combine to define the dimensions of an attack. From this perspective we are primarily interested in the dimensions of:

Attack model: The attack model, which we describe in detail in Section 3, specifies the rating characteristics of the attack profile. The model associated with an attack details the items that should be included in the attack profiles, and the strategy that should be used in assigning ratings to these items.

Attack intent: The intent of an attack describes the intent of the attacker. Two simple intents are "push" and "nuke". An attacker may insert profiles to make a product more likely ("push") or less likely ("nuke") to be recommended. Another possible aim of an attacker might be simple vandalism – to make the entire system function poorly. Our work here assumes a more focused economic motivation on the part of the attacker, namely that there is something to be gained by promoting or demoting a particular product. (Scenarios in which one product is promoted and others simultaneously attacked are outside the scope of this paper.)

	Item1	Item2	Item3	Item4	Item5	Item6	Correlation with Alice
Alice	5	2	3	3		?	
User1	2		4		4	1	-1.00
User2	3	1	3		1	2	0.76
User3	4	2	3	1		1	0.72
User4	3	3	2	1	3	1	0.21
User5		3		1	2		-1.00
User6	4	3		3	3	2	0.94
User7		5		1	5	1	-1.00
Attack1	**5**		**3**		**2**	**5**	**1.00**
Attack2	**5**	**1**	**4**		**2**	**5**	**0.89**
Attack3	**5**	**2**	**2**	**2**		**5**	**0.93**

Fig. 1. An example of a push attack favoring the target item Item6.

Profile size: The number of ratings assigned in a given attack profile is the profile size. The addition of ratings is relatively lower in cost for the attacker compared to the creating of additional profiles. However, there is the additional factor of risk at work when profiles include ratings for a large percentage of the rateable items. Real users rarely rate more than a small fraction of the rateable items in a large recommendation space. No one can read every book that is published or view every movie. So, attack profiles with many, many ratings are easy to distinguish from those of genuine users and are a reasonably certain indicator of an attack.

Attack size: The attack size is the number of profiles inserted related to an attack. We assume that a sophisticated attacker will be able to automate the profile injection process. Therefore, the number of profiles is a crucial variable because it is possible to build on-line registration schemes requiring human intervention, and by this means, the site owner can impose a cost on the creation of new profiles.

In our investigation we examine how these dimensions affect detection of profile injection attacks.

2.2 Types of Attacks

An attack against a collaborative filtering recommender system consists of a set of attack profiles, each containing biased rating data associated with a fictitious user identity, and a target item, the item that the attacker wishes the system to recommend more highly (a *push* attack), or wishes to prevent the system from recommending (a *nuke* attack). We provide two hypothetical examples that will help illustrate the vulnerability of collaborative filtering algorithms, and will serve as a motivation for the attack models, described more formally in the next section.

2.2.1 *A Push Attack Example.* Consider, as an example, a recommender system that identifies books that users might like to read using a user-based collaborative algorithm [Herlocker et al. 1999]. A user profile in this hypothetical system might consist of that user's ratings (in the scale of 1-5 with 1 being the lowest) on various books. Alice, having built up a profile from previous visits, returns to the system for new recommendations. Figure 1 shows Alice's profile along with that of seven genuine users. An attacker, Eve, has inserted attack profiles (Attack1-3) into the system, all of which give high ratings to her book labeled Item6. Eve's attack profiles may closely match the profiles of one or more of the existing users (if Eve is able to obtain or predict such information), or they may be based on average or expected ratings of items across all users.

Suppose the system is using a simplified user-based collaborative filtering approach where the predicted ratings for Alice on Item6 will be obtained by finding the closest neighbor to Alice. Without the attack profiles, the most similar user to Alice, using correlation-based

	Item1	Item2	Item3	Item4	Item5	Item6	Correlation with Alice
Alice	5	2	3	3		?	
User1	1		4		4	2	-1.00
User2	2	1	3		1	3	0.33
User3	1	2	3	1		4	-0.48
User4	1	3	2	1	3	3	-0.76
User5		3		1	2		-1.00
User6	2	3		3	3	4	-0.94
User7		5		1	5	2	-1.00
Attack1	**3**		**2**		**3**	**1**	1.00
Attack2		**3**		**2**		**1**	-1.00
Attack3	**3**		**2**		**3**	**1**	1.00

Fig. 2. An example of a nuke attack disfavoring the target item Item6.

similarity, would be User6. The prediction associated with Item6 would be 2, essentially stating that Item6 is likely to be disliked by Alice. After the attack, however, the Attack1 profile is the most similar one to Alice, and would yield a predicted rating of 5 for Item6, the opposite of what would have been predicted without the attack. So, in this example, the attack is successful, and Alice will get Item6 as a recommendation, regardless of whether this is really the best suggestion for her. She may find the suggestion inappropriate, or worse, she may take the system's advice, buy the book, and then be disappointed by the delivered product.

2.2.2 *A Nuke Attack Example.* Another possible intent besides pushing an item is to "nuke" an item (i.e., to cause it to be recommended less frequently). Perhaps Eve wants her buyers not to be recommended a book by her closest competitor. Figure 2 shows this situation. Eve has decided to influence the system so that Item6 is rarely recommended. Prior to the addition of Eve's attack profiles User2 would be regarded as the one most similar to Alice, and so the system would give Item6 a neutral rating of 3. Eve inserts attack profiles (Attack1-Attack3) into the system, all of which give low ratings to Item6, and some ratings to other items. Once these attack profiles are in place, the system would select Attack1 as the nearest neighbor, yielding a predicted rating of 1 for Item1, which lead the recommender system to switch its prediction to dislike.

Both of these examples have been greatly simplified for illustrative purposes. In real world systems both the product space and user database are much larger and more neighbors are used in prediction, but the same problem still exists.

2.3 Detecting Profile Injection Attacks

One of the main strengths of collaborative recommender systems is the ability for users with unusual tastes to get meaningful suggestions by the system identifying users with similar peculiarities. This strength is also one of the challenges in securing recommender systems. Specifically the variability of opinion makes it difficult to say with certainty whether a particular profile is an attack profile or the preferences of an eccentric user. Making it perhaps unrealistic to expect all profiles to be classified correctly. The goals for detection and response will therefore be:

—Minimize the impact of an attack,

—Reduce the likelihood of a successful attack, and

—Minimize any negative impact resulting from the addition of the detection scheme

The attacks that we outline below work well against collaborative algorithms because they were created by reverse engineering the algorithms to devise inputs with maximum impact.

Attacks that deviate from these patterns will likely be less effective than those that conform to them. Our approach to attack detection will therefore focus on recognizing attacks based on known attack models. An ideal outcome would be one in which a system could be rendered secure by making attacks against it no longer cost effective, where cost is measured in terms of the attacker's knowledge, effort, and time.

Our approach to attack detection is one that builds on multiple sources of evidence:

Profile characteristics Profiles designed as part of an attack will look very different in aggregate than genuine user profiles, simply because they are designed to be efficient – to achieve maximum change in recommendation behavior with the minimum number of profiles injected.

Critical mass A single profile that happens to match the pattern of, for example, an *average* attack may just happen to be a statistical fluke. A thousand such profiles, all of which target the same item, are probably not. Profile injection attacks need a critical mass of attack profiles in order to be successful. Such profiles will not only have suspicious characteristics but they will share a focus on a single attacked item.

Time series The entry of ratings that comprise a given user profile represent a time series. Obviously, a profile that is built up at inhuman speed (for example, 100 ratings in one minute) is almost certain to be an attack. Similarly, the set of ratings associated with a given item also can be examined as a time series. An item that shows a sudden burst of identical high or low ratings may be one that is being pushed or nuked.

Vulnerability Some items in the database are more vulnerable to attack than others. Items with very few ratings will be more volatile in the face of attack profiles, which may overwhelm the genuine ratings. Our experiments have shown that while relatively popular items are still vulnerable to attack, items that are sparsely rated are most vulnerable.

In other words, we will expect an attack to consist of a set of profiles having ratings that accrue in an atypical pattern, incorporating ratings for vulnerable items, having statistical properties matching those of a known attack model and focusing on a particular item. The more of these properties that are true of a set of profiles the more likely it is that they constitute an attack.

3. PROFILE INJECTION ATTACKS

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* [O'Mahony et al. 2004].

An *attack model* is an approach to constructing the attack profiles, based on knowledge about the recommender system's rating database, products, and/or users. 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 as depicted in Figure 3. The null partition, $I_\emptyset$, are those items with no ratings in the profile. The single target item i_t will be given a rating designed to bias its recommendations, generally this will be either the maximum (r_{max}) or minimum (r_{min}) 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 usually receives high ratings to make the profiles similar to those of users who prefer these product. Finally, there is a set of filler items I_F whose ratings are added to complete the profile. It is the strategy for selecting items in I_S and I_F and the ratings given to these items that define an attack model and give it its character.

A profile injection attack against a collaborative system, generally, consists of a number of attack profiles of the same type (i.e., based on the same attack model) added to the database

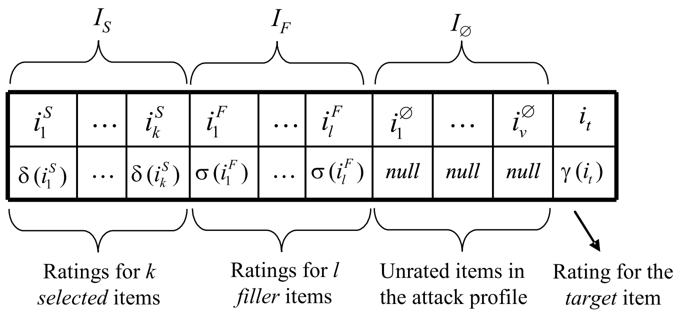

I_S
I_F
$I_\emptyset$
i_1^S … i_k^S i_1^F … i_l^F $i_1^\emptyset$ … $i_v^\emptyset$ i_t
$\delta(i_1^S)$ … $\delta(i_k^S)$ $\sigma(i_1^F)$ … $\sigma(i_l^F)$ null null null $\gamma(i_t)$
Ratings for *k* *selected* items
Ratings for *l* *filler* items
Unrated items in the attack profile
Rating for the *target* item

Fig. 3. The general form of a user profile from a profile injection attack.

of real user profiles. The goal of such an attack is to increase (in the case of a push attack) or decrease (in a nuke attack) the system's predicted rating on a target item for a given user (or a group of users). Below we describe the traits of several of the most effective known push and nuke attack models.

3.1 Push Attack Models

Two basic attack models, introduced originally in Lam and Reidl [2004] are the *random* and *average* attack models. Both of these attack models involve the generation of attack profiles using randomly assigned ratings to the filler items in the profile. In the random attack the assigned ratings are based on the overall distribution of user ratings in the database, while in the average attack the rating for each filler item is computed based on its average rating for all users.

3.1.1 *Random Attack.* Profiles created by the random attack consist of random ratings assigned to the filler items and a pre-specified rating assigned to the target item. In this attack model, the set of selected items is empty. More formally, the random attack model has the following characteristics:

—$I_S = \emptyset$;
—I_F is a set of randomly chosen filler items drawn from $I - \{i_t\}$. The rating value for each item $i \in I_F$ is drawn from a normal distribution around the mean rating value across the whole database;
—rating for $i_t = r_{max}$

The knowledge required to mount such an attack is quite minimal, especially since the overall rating mean in many systems can be determined by an outsider empirically (or, indeed, may be available directly from the system). However, as Lam and Reidl [2004] shows and our results confirm [Burke et al. 2005], the attack is not particularly effective as a push attack.

3.1.2 *Average attack.* A more powerful attack described in Lam and Reidl [2004] uses the individual mean for each item rather than the global mean (except for the pushed item). In the average attack, each assigned rating for a filler item corresponds (either exactly or approximately) to the mean rating for that item, across the users in the database who have rated it. More formally, the average attack model has the following characteristics:

—$I_S = \emptyset$;
—I_F is a set of randomly chosen filler items drawn from $I - \{i_t\}$, where the ratio of filler items, where the rating value for each item $i \in I_F$ is drawn from a normal distribution around the mean rating for i;
—rating for $i_t = r_{max}$

As in the random attack, this attack can also be used as a nuke attack by using r_{min} instead of r_{max} in the above definition. It should also be noted that the only difference between the average attack and the random attack is in the manner in which ratings are assigned to the filler items in the profile.

In addition to these standard attack models, several more sophisticated models have been studied in order to reduce knowledge requirements while still being effective. In this work we have evaluated two such models, the bandwagon and segment attacks.

3.1.3 *Bandwagon attack.* The goal of the bandwagon attack is to associate the attacked item with a small number of frequently rated items. This attack takes advantage of the Zipf's law distribution of popularity in consumer markets: a small number of items, best-seller books for example, will receive the lion's share of attention and also ratings. The attacker using this model will build attack profiles containing those items that have high visibility. Such profiles will have a good probability of being similar to a large number of users, since the high visibility items are those that many users have rated. For example, by associating her book with current best-sellers, for example, *The DaVinci Code*, Eve can ensure that her bogus profiles have a good probability of matching any given user, since so many users will have these items on their profiles. This attack can be considered to have low knowledge cost. It does not require any system-specific data, because it is usually not difficult to independently determine what the "blockbuster" products are in any product space. More formally, the bandwagon attack model has the following characteristics:

—I_S is a set of widely popular items; where each item $i \in I_S$ is given the rating r_{max}

—I_F is a set of randomly chosen filler items drawn from $I - (\{i_t\} \cup I_S)$, where the rating value for each item $i \in I_F$ is drawn from a normal distribution around the mean rating value across the whole database;

—rating for $i_t = r_{max}$

Items i_1^S through i_k^S in I_S are selected because they have been rated by a large number of users in the database. These items are assigned the maximum rating value together with the target item, i_t. The ratings for the filler items i_1^F through i_l^F in I_F are determined randomly in a similar manner as in the random attack. The bandwagon attack therefore can be viewed as an extension of the random attack. We showed in [Burke et al. 2005] that the bandwagon attack is nearly as effective as the average attack against user-based algorithms, but without the knowledge requirements of that attack. Thus, it is more practical to mount.

3.1.4 *Segment Attack.* From a cost-benefit point of view, the attacks discussed thus far are sub-optimal; they require a significant degree of system-specific knowledge to mount, and they push items to users who may not be likely purchasers. To address this, we introduced the *segment attack* model as a reduced knowledge push attack specifically designed for the item-based algorithm, but which we have also shown to be effective against the user-based algorithm [Mobasher et al. 2005; Mobasher et al. 2006].

The principle behind the segment attack, is that the best way to increase the cost/benefit of an attack is to target one's effort to those already predisposed towards one's product. In other words, it is likely that an attacker wishing to promote a particular product will be interested not in how often it is recommended to all users, but how often it is recommended to likely buyers. The segment attack model is designed to push an item to a targeted group of users with known or easily predicted preferences. For example, suppose that Eve, in our previous example, had written a fantasy book for children. She would no doubt prefer that her book be recommended to buyers who had expressed an interest in this genre, for example buyers of *Harry Potter* books, rather than buyers of books on Java programming or motorcycle repair. Eve would rightly expect that the "fantasy book buyer" segment of the market would be more likely to respond to a recommendation for her book than others. In addition, it would be to the benefit of the attacker to reduce the impact to unlikely buyers if as a consequence the broad range of the bias made the attack easier to detect.

If there is no cost to mounting a broad attack, there is no harm in pushing one's product to the broadest possible audience. However, there are two types of cost associated with broad attacks. One is that non-sequitur recommendations (children's fantasy books recommended to the reader of motorcycle books) are more likely to generate end-user complaints and rouse suspicions that an attack is underway. The second is that (as our experiments indicate below) larger, broader attacks are easier to detect by automated means. An attacker is therefore likely to opt for a smaller attack that yields the largest portion of the possible profit to be gained rather than a larger one with a small marginal utility and increased risk of detection. More formally, the segment attack model has the following characteristics:

—I_S is a set of selected items the attacker has chosen to define the segment; where all $i \in I_S$ will be given the rating r_{max}.

—I_F is a set of randomly chosen filler items; where all $i \in I_F$ will be given the rating r_{min}.

—rating for $i_t = r_{max}$

The target group of users (segment) in the segment attack model can then be defined as the set $U_S = \{up_1 \cdots up_k\}$ of user profiles in the database such that: $\forall up_j \in U_S$, $\forall i \in I_S$, $rating(up_j, i) \geq r_c$, where $rating(up_j, i)$ is the rating associated with item i in the profile up_j^S, and r_c is a pre-specified minimum rating threshold.

3.2 Nuke Attack Models

All of the attack models described above can also be used for nuking a target item. For example, as noted earlier, in the case of the random and average attack models, this can be accomplished by associating rating r_{min} with the target item i_t instead of r_{max}. However, our experimental results, presented in Section 6, suggest that attack models that are effective for pushing items are not necessarily effective for nuke attacks.

We have identified one additional attack model designed particularly for nuking: the *love/hate attack*. It is a very simple attack, with no knowledge requirements. The attack consists of attack profiles in which the target item i_t is given the minimum rating value, r_{min}, while other ratings in the filler item set are the maximum rating value, r_{max}. A variation of this attack can also be used as a push attack by switching the roles of r_{min} and r_{max}. More formally, the love/hate attack model has the following characteristics:

—$I_S = \emptyset$;

—I_F is a set of randomly chosen filler items; where all $i \in I_F$ will be given the rating r_{max}.

—rating for $i_t = r_{min}$

Clearly, the knowledge required to mount such an attack is quite minimal. Furthermore, as our results will show, it is one of the most effective nuke attacks against the user-based collaborative filtering algorithm.

3.3 Attack Summary

Table I summarizes the characteristics of the attack models discussed above. There are a couple of key similarities that are worth noting as they will factor into detection similarities as well. The bandwagon and segment attacks differ from the other attacks, in that they have a group of items ($I_S \cup i_t$) that are common across all profiles, whereas in the other models the only guaranteed common item is the target item i_t. The random and bandwagon attacks are differentiated only by the bandwagon's use of the I_S partition. The segment and love/hate attack are similar in the same way but with the ratings of the I_F and i_t items reversed. While theoretically any of these models can be used as both a push and nuke attack, we have focused on using these attacks as described in the table. These 7 attack models (4 push, 3 nuke) will be used throughout our experiments to evaluate the capabilities of our defense schemes.

Attack type	**Attack model**	I_S	I_F	$I_\emptyset$	i_t
Random	push/ nuke	Not used	Ratings assigned with normal distribution around system mean	Determined by filler size	r_{max}/ r_{min}
Average	push/ nuke	Not used	Ratings assigned with normal distribution around item mean	Determined by filler size	r_{max}/ r_{min}
Bandwagon	push	Widely popular items assigned rating r_{max}	Ratings assigned with normal distribution around system mean	Determined by filler size	r_{max}
Segment	push	Items chosen to define the segment assigned rating r_{max}	Ratings assigned with r_{min}	Determined by filler size	r_{max}
Love/ Hate	nuke	Not used	Ratings assigned with r_{max}	Determined by filler size	r_{min}

Table I. Attack model summary

4. DETECTION ATTRIBUTES FOR PROFILE CLASSIFICATION

In this section, we present a more in-depth look at our approach to attack detection and response based on profile classification: we analyze the characteristics of profiles and label each profile either as an authentic or an attack profile. Prior work in detecting attacks in collaborative filtering systems have mainly focused on ad hoc algorithms for identifying basic attack models such as the random attack [Chirita et al. 2005]. We propose an alternate technique based on more traditional supervised learning and show this approach can be effective at reducing the effects of the attacks discussed above.

Due to the sparsity and high dimensionality of the ratings data, applying a supervised learning approach to the raw data is impractical. The vast number of combinations that would be required to create an adequate training set to incorporate all attack models and all potential target items would be unrealistic. As a result, we have focused on profile analytics data and attribute reduction techniques to lower the dimensionality of the data. The training set is created as a combination of user data from the MovieLens dataset and attack profiles generated using the attack models described in Section 3. Each profile is labeled as either being part of an attack or as coming from a genuine user. (We assume that the MovieLens data is attack-free.) A binary classifier is then created based on this set of training data using the attributes described below and any profile classified as an attack is not used in predictions.

For this method, training data is created by combining a number of genuine profiles from historic data with attack profiles inserted following the attack models described above. Each profile is labeled as either being part of an attack or as coming from a genuine user. A binary classifier is then created based on this set of training data using the attributes described below and any profile classified as an attack will not be used in predictions.

The attributes we have examined come in three varieties: generic, model-specific, and intra-profile. The generic attributes, modeled on basic descriptive statistics, 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 designed to detect characteristics of profiles that are generated by specific attack models. The intra-profile attributes are designed to detect concentrations across profiles. Below we outline a set of detection attributes introduced in [Chirita et al. 2005; Burke et al. 2006b; Mobasher et al. 2006] and some additional attributes and examine their combined effectiveness at defending against a selection of attacks discussed in this work.

4.1 Generic Detection Attributes

Generic attributes are based on the hypothesis that the overall statistical signature of attack profiles will differ 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 [Lam and Reidl 2004; Chirita et al. 2005; O'Mahony et al. 2004; Mobasher et al. 2005], 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 are likely to deviate from rating patterns seen for authentic users.

For the detection classifier's data set we have used a number of generic attributes to capture these distribution differences, several of which we have extended from attributes originally proposed in [Chirita et al. 2005].
These attributes are:

—*Rating Deviation from Mean Agreement* (RDMA) [Chirita et al. 2005], is 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. The attribute is calculated as follows:

$$RDMA_u = \frac{\sum_{i=0}^{N_u} \frac{|r_{i,u} - \overline{r_i}|}{NR_i}}{N_u}$$

where N_u is the number of items user u rated, $r_{i,u}$ is the rating given by user u to item i, $\overline{r_i}$ is the average rating of item i, NR_i is the overall number of ratings in the system given to item i.

—*Weighted Degree of Agreement* (WDA), is introduced to capture 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 not weighted by the number of ratings by the user, thus only the numerator of the RDMA equation.

—*Weighted Deviation from Mean Agreement* (WDMA), designed to help identify anomalies, places a high weight on rating deviations for sparse items. We have found it to provide the highest information gain of the attributes we have studied. The WDMA attribute can be computed in the following way:

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

where U is the universe of all users u; 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; and 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.

—*Degree of Similarity with Top Neighbors* (DegSim) [Chirita et al. 2005], captures the average similarity of a profile's k nearest neighbors. As researchers have hypothesized attack profiles are likely to have a higher similarity with their top 25 closest neighbors than real users [Chirita et al. 2005; Resnick et al. 1994]. We also include a second slightly different attribute $DegSim'$, which captures the same metric as DegSim, but is based on the average similarity discounted if the neighbor shares fewer than d ratings in common. We have found this variant provides higher information gain at low filler sizes.

—*Length Variance* (LengthVar) is introduced to capture how much the length of a given profile varies from the average length in the database. If there are a large number of possible items, it is unlikely that very large profiles come from real users, who would have to enter them all manually, as opposed to a soft-bot implementing a profile injection attack. As a result, this attribute is particularly effective at detecting attacks with large filler sizes. This feature is computed as follows:

$$LengthVar_u = \frac{\left|\#ratings_u - \overline{\#ratings}\right|}{\sum_{i=0}^{N} \left(\#ratings_i - \overline{\#ratings}\right)^2}$$

where $\#ratings_u$ is the total number of ratings in the system for user u, and N is the total number of users in the system.

4.2 Model-Specific Detection Attributes

In our experiments, we have found that the generic attributes are insufficient for distinguishing attack profiles from eccentric but authentic profiles [Burke et al. 2006b; Mobasher et al. 2006]. This is especially true when the profiles are small, containing few filler items. As shown in Section 3, attacks can be characterized based on the characteristics 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.

Our detection model discovers partitions of each profile that maximize its similarity to the attack model. To model this partitioning, each profile for user u is split into three sets. The set $P_{u,T}$ contains the items in the profile that are suspected to be targets, $P_{u,F}$ contains all items within the profile that are suspected to be filler items, and $P_{u,\emptyset}$ the unrated items. Thus the intention is for $P_{u,T}$ to approximate $\{i_t\} \cup I_S$, $P_{u,F}$ to approximate I_F, and $P_{u,\emptyset}$ is equal to $I_\emptyset$. (We do not attempt to differentiate i_t from I_S.) While in this paper the suspected target item has been identified by these models alone, 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. Below we outline our model specific detection attributes for the random and average attack models, as well as a group attack model.

4.2.1 *Average Attack Model-Specific Detection Attributes.* Generation of the *average model-specific detection attributes* divides the profile into the 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. Therefore, 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 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 each possible p_{target} in the profile P_u of user u where p_{target} is from the set of items $P_{u,target}$ in P_u that are given the rating r_{target} (the maximum rating for push attack detection or the minimum rating for nuke attack detection).

$$MeanVar(p_{target}, u) = \frac{\sum\limits_{i \in (P_u - p_{target})} (r_{i,u} - \overline{r_i})^2}{|P_u| - 1}$$

where P_u is the profile of user u, p_{target} is the hypothesized target item, $r_{i,u}$ is the rating user u has given item i, $\overline{r_i}$ is the mean rating of item i across all users, and $|P_u|$ is the number of ratings in profile P_u. We then select the target t from the set $P_{u,target}$ such that $MeanVar(t, u)$ is minimized. From this optimal partitioning of $P_{u,target}$, we use $MeanVar(t, u)$ as the *Filler Mean Variance* feature for classification purposes. The item t becomes the set $P_{u,T}$ for the detection model and all other items in P_u become $P_{u,F}$.

These two partitioning sets $P_{u,T}$, and $P_{u,F}$ are used to create two sets of the following attributes (one for detecting push attacks and one for detecting nuke attacks):

—*Filler Mean Variance* (MeanVar), the partitioning metric described above.

—*Filler Mean Difference* (FillerMeanDiff), 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.)

—*Profile Variance*, capturing within-profile variance as this tends to be low compared to authentic users

4.2.2 *Random Attack Model-Specific Detection Attributes.* The random attack is also a partitioning attack, but in this case, the ratings for the filler items are chosen randomly such that their mean is the overall system average rating across all items.

As in the average attack, we can test various partitions by selecting possible targets and computing a metric for each possible choice. Since we are here aiming to detect a random attack, we use the correlation between the profile and the average rating for each item. Since the ratings are generated randomly, we would expect low correlation between the filler items and the individual ratings. (Note that this is the opposite of what we would expect for an average attack that would be very highly correlated with the item average.) We again select the most likely target t, for which the correlation between the filler and the item averages is minimized. We call this minimum the *Filler Average Correlation.*

4.2.3 *Group Attack Model-Specific Detection Attributes.* The *group attack model-specific attributes* are introduced for detecting attacks that intend to increase the separation of a target a group of items ($i_t \cup I_S$) and the filler items (I_F), such as the bandwagon and segment attacks. Unlike the average and random model-specific attributes which are designed to recognize specific characteristics within partitions, this model intends to capture differences between partitions. The basic concept is common across all attacks, but is particularly apparent in the segment and love/hate attacks.

For this model, $P_{u,T}$ is set to all items in P_u that are given the maximum rating (minimum for nuke attacks) in user u's profile, and all other items in P_u become the set $P_{u,F}$. The partitioning feature that maximizes the attack's effectiveness is the difference in ratings of items in the $i_{target} \cup I_S$ compared to the items in I_F. Thus we introduce the *Filler Mean Target Difference* (FMTD) attribute. The attribute is calculated as follows:

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

where $r_{u,i}$ is the rating given by user u to item i. The overall average $\overline{FMTD}$ is then subtracted from $FMTD_u$ as a normalizing factor. The variance of the filler items identified by the group attack partitioning is also added as an attribute *Group Filler Mean Variance* (FMV) and is calculated as follows:

$$FMV_u = \frac{\sum\limits_{i \in P_{u,F}} (r_{i,u} - \overline{r_i})^2}{|P_{u,F}|}$$

where $P_{u,F}$ is the set of items in the profile of user u that have been partitioned as filler items, $r_{i,u}$ is the rating user u has given item i, $\overline{r_i}$ is the mean rating of item i across all users, and $|P_{u,F}|$ is the number of ratings in the set $P_{u,F}$.

4.2.4 *Intra-profile Detection Attributes.* Unlike the attributes thus far which have concentrated on characteristics within a single profile, intra-profile attributes focus on statistics across profiles. As our results above show, attackers often must inject multiple profiles (attack size) in order to introduce a significant bias. Thus, if a system is attacked there are likely to be several attack profiles that target the same item. To capture this intuition, we introduce the *Target Model Focus* (TMF) attribute. This attribute leverages the partitioning identified by the model-specific attributes to detect concentrations of target items. Using these partitions the TMF attribute calculates the degree to which the partitioning of a given profile focuses on items common to other attack partitions. Thus, the TMF attribute attempts to measure the consensus of suspicion regarding each profile's most suspicious target item. To compute TMF, let $q_{i,m}$ be the total number of times each item i is included in any target set $P_{u,T}$ used in the partitioning m for the model-specific attributes. Let T_u be the union of all items identified for user u in any target set $P_{u,T}$ used by the model-specific attributes. TargetFocus is calculated for user u, item i, and model-specific partitioning m as:

$$TargetFocus(u,i,m) = \frac{q_{i,m}}{\sum\limits_{j \in I} q_{j,m}}$$

where I is the set of all items. Thus, TMF_u is taken to be the maximum value of the function $TargetFocus(u,t,m)$ across all m model-specific partitions and t in T_u.

5. METHODOLOGY

Below we describe the methodology and recommendation algorithm we have used in our experiments. This is followed by a detailed description of the metrics we have used to evaluate attributes, classification performance, and detection robustness of a detection scheme composed of the attributes described in Section 4.

5.1 Recommendation Algorithm

In this paper we focus on user-based collaborative filtering, one of the most commonly-used recommender algorithms. While other algorithms such as item-based are also widely used, we focus on user-based recommendation because its user-to-user approach simplifies interpretation of the detection effectiveness since the detection scheme is also user-based.

The standard collaborative filtering algorithm is based on user-to-user similarity [Herlocker et al. 1999]. This kNN algorithm operates by selecting the k most similar users to the target user, and formulates a prediction by combining the preferences of these users. kNN is widely used and reasonably accurate. The similarity between the target user, u, and a neighbor, v, can be calculated by the Pearson's correlation coefficient defined below:

$$sim_{u,v} = \frac{\sum\limits_{i \in I} (r_{u,i} - \bar{r}_u) * (r_{v,i} - \bar{r}_v)}{\sqrt{\sum\limits_{i \in I} (r_{u,i} - \bar{r}_u)^2} * \sqrt{\sum\limits_{i \in I} (r_{v,i} - \bar{r}_v)^2}}$$

where I is the set of all items that can be rated, $r_{u,i}$ and $r_{v,i}$ are the ratings of some item i for the target user u and a neighbor v, respectively, and $\bar{r}_u$ and $\bar{r}_v$ are the average of the ratings of u and v over those items in I that u and v respectively have in common.

Once similarities are calculated, the most similar users are selected. In our implementation, we have used a value of 20 for the neighborhood size k. 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. Once the most similar users are identified, we use the following formula to compute the prediction for an item i for target user u.

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

where V is the set of k similar users and $r_{v,i}$ is the rating of those users who have rated item i, $\bar{r}_v$ is the average rating for the target user over all rated items, and $sim_{u,v}$ is the mean-adjusted Pearson correlation described above. The formula in essence computes the degree of preference of all the neighbors weighted by their similarity and then adds this to the target user's average rating: the idea being that different users may have different "baselines" around which their ratings are distributed. If the denominator of the above equation is zero, our algorithm replaces the prediction by the average rating of user u.

We incorporate attack detection by following the lead of Chirita et al. [2005] of using a parameter PA_u, the probability that a profile u is an attack profile. 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 probability of attack (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.

5.2 Evaluation Metrics

There has been considerable research in the area of recommender systems evaluation [Herlocker et al. 2004]. Some of these concepts can also be applied to the evaluation of the security of recommender systems, but in evaluating security, the vulnerability of the recommender to attack is of more interest than the raw performance. The system's ability to recognize an attack and the resulting change in performance induced by an attack provide better insight into the robustness of the detection scheme.

5.2.1 *Attribute Evaluation Metrics.* To evaluate the benefit gained by the addition of each detection attribute, we are interested in their usefulness in distinguishing attack profiles from authentic profiles. We use one of the most well known and widely used measures for evaluating how informative an attribute is, information gain [Hunt et al. 1966]. This metric measures the increase in information associated applying a classification attribute over the expected information required to encode a set based on its entropy. For a two class domain, the information needed to decide if an arbitrary instance belongs to class P or N is defined in terms of entropy and can be calculated as follows:

$$I(p, n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

where p is the number of elements of class P and n is the number of elements of class N. Assume that applying an attribute A to a set S will partition the set into sets $\{S_1, S_2, \cdots, S_v\}$, the entropy or expected information needed to classify instances in all sub trees S_i can be calculated as follows:

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} I(p_i, n_i)$$

where p_i is the examples of class P and n_i the examples of N in the set S_i. The information gained by applying attribute could then be calculated as follows:

$$InformationGain(A) = I(p, n) - E(A)$$

where A is the attribute being evaluated.

5.2.2 *Classification Performance Metrics.* For measuring classification performance, we use the standard binary classification measurements of specificity and sensitivity. The basic definition of *specificity* and *sensitivity* can be written as:

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

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

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 *# true positives* is the number of correctly classified attack profiles, *# false positives* is the number of authentic profiles misclassified as attack profiles, and *# false negatives* is the number of attack profiles misclassified as authentic profiles. Thus *sensitivity* measures the proportion of attack profiles correctly identified, and *specificity* measures the proportion of authentic profiles correctly identified.

In our past work we have used the metrics of *precision* and *recall* for evaluating attack profile identification. The reason for the change to the metrics of specificity and sensitivity in this investigation is due to the additional focus on the impact of misclassified authentic users. In the context of binary classification, sensitivity is identical to recall, however specificity and precision provide two slightly different views of misclassification errors. Specifically the precision measure calculated as the fraction of # *true positives* (actual attacks) among all those profiles labeled as possible attacks; while it provides insight into how accurately the classifier identifies attack profiles, it does not provide insight into what percent of the authentic profiles are correctly classified, a key factor in the performance of a collaborative system. In contrast, specificity measures the percent of authentic profiles correctly classified, thus providing insight as to the portion of the original authentic profiles that are used for prediction.

In addition to these classification metrics, we are also interested in measuring the effect of discounting misclassified authentic profiles on predictive accuracy. We evaluate this impact by examining a commonly used metric for evaluating recommender predictive accuracy, *mean absolute error* (MAE). Assume that the set T is a set of ratings in a test set, then the MAE of a recommender system trained on an authentic ratings set R can be calculated as follows:

$$MAE = \frac{\sum\limits_{t \in T} |t_{u,i} - p_{u,i}|}{\|T\|}$$

where $t_{u,i}$ is a rating in T for user u and item i, $p_{u,i}$ is the predicted rating for user u and item i, and $\|T\|$ is the number of ratings in the set T. Since we are interested in the change in predictive accuracy rather than the raw accuracy, we examine ΔMAE calculated as follows:

$$\Delta MAE = MAE - MAE'$$

where *MAE'* is the MAE of the recommender trained on the authentic ratings set R' which includes only correctly classified authentic profiles.

5.2.3 *Robustness Evaluation Metrics.* In O'Mahony et al. [2004] two evaluation measures were introduced: *robustness* and *stability*. Robustness measures the performance of the system before and after an attack to determine how the attack affects the system as a whole. Stability looks at the shift in system's ratings for the attacked item induced by the attack profiles.

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 O'Mahony et al. [2004] in measuring stability via prediction shift. However, we also measure the average likelihood that a top N recommender will recommend the pushed item, the "hit ratio" [Sarwar et al. 2001]. This allows us to measure the effectiveness of the attack on the pushed item compared to all other items.

Average prediction shift is defined as follows. Let U_T and I_T be the sets of users and items, respectively, in the test data. For each user-item pair (u, i) the prediction shift denoted by $\Delta_{u,i}$, can be measured as $\Delta_{u,i} = p'_{u,i} - p_{u,i}$, where p' represents the prediction after the attack and p before. A positive value means that the attack has succeeded in making the pushed item more positively rated. The average prediction shift for an item i over all users can be computed as:

$$\Delta_i = \sum_{u \in U_T} \Delta_{u,i} / |U_T| .$$

Similarly the average prediction shift for all items tested can be computed as:

$$\bar{\Delta} = \sum_{i \in I_T} \Delta_i / |I_T| .$$

6. EXPERIMENTAL RESULTS

In our experiments we use the publicly-available Movie-Lens 100K dataset[3]. 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. In all experiments, we use a neighborhood size of 20 in the user-based k-nearest-neighbor algorithm.

To conduct our attack experiments, the dataset was split into training and test sets. Our attacks target a sample of 50 users and 50 movies. The 50 target movies were selected randomly and represent a wide range of average ratings and number of ratings. Table II shows the statistics of the 50 target movies, where cell values represent how many of these movies fall into the specified group.

	Average Rating			
Ratings	1-2	2-3	3-4	4-5
1 - 50	6	15	9	3
51 - 150			7	3
151 - 250			2	2
> 250			1	2

Table II. Statistics of Target Movies

We also randomly selected a sample of 50 target users whose mean rating mirrors the overall mean rating (which is 3.6) of all users in MovieLens database. Table III shows the statistics of the 50 target users, where cell values represent how many of these users fall into these categories.

Ratings			
20 - 50	51 - 150	151 - 250	> 250
22	16	6	6

Table III. Statistics of Target Users

Each of these target movies was attacked individually and the results reported below represent averages over the combinations of test users and test movies.

We use the metrics of information gain, sensitivity, specificity, ΔMAE, and prediction shift, as described earlier, to measure the benefit of attributes, detection performance, and robustness against various attack models. Generally, the values of these metrics are plotted against either filler size or attack size. Filler size is reported as the percentage of items in I_F of all items in $I_F \cup I_\emptyset$. The size of the attack is reported as a percentage of the total number of profiles in the system.

For all the attacks, we generated a number of attack profiles and inserted them into the system database and then generated predictions. We measure "size of attack" as a percentage of the pre-attack user count. There are approximately 1000 users in the database, so an attack size of 1% corresponds to 10 attack profiles added to the system.

In the results below, we present an overview of the vulnerabilities in user-based collaborative filtering as motivation for investigating techniques to secure these systems. This is followed by a detailed analysis of the information gain of each of the attributes described above for various attack models and attack parameters. Next we report the combined performance of these attributes at classifying push and nuke attack profiles, and the impact on prediction quality that results from adding such a classifier to user-based recommendation. Finally we present the

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

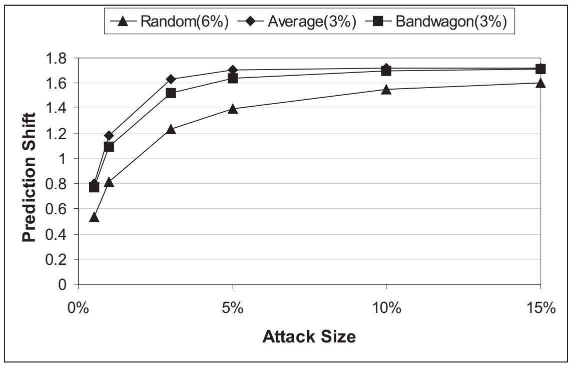


Fig. 4. Prediction shift comparison of attacks against user-based algorithm

increase in robustness from using this detection scheme with user-based collaborative filtering against both push and nuke attacks.

6.1 Vulnerability Analysis

In the results below, we present the vulnerabilities of user-based collaborative filtering against push and nuke attacks. We report some interesting differences in the effectiveness of the various attack models depending on whether they are used for nuke or push attacks.

6.1.1 *Vulnerability Against Push Attacks.* For the bandwagon attack we use 5 "bandwagon" movies. The five movies chosen were those with the most ratings in the database. In the case of the MovieLens data, these frequently-rated items are predictable box office successes including such titles as Star Wars, Return of Jedi, Titanic, etc. Obviously, this is a form of system-specific knowledge, increasing the knowledge requirements of this attack somewhat. However, we verified the general popularity of these movies using external data sources[4][5] and found they would be among anyone's list of movies likely to have been seen by many viewers.

Figure 4 shows the results of a comparative experiment examining three attack models at different attack sizes. The algorithms include the random attack (6% filler size), the average attack (3% filler size), and the bandwagon attack (using 5 frequently rated item and 3% filler size). These parameters were chosen pessimistically as they are the versions of each attack that were found to be most effective. We see that even without system-specific data an attack like the bandwagon attack can be successful at higher attack levels. The more knowledge-intensive average attack is still better, with the best performance achieved using profiles with relatively small filler sizes. The total knowledge used in the average attack is obviously quite powerful - recall that the rating scale in this domain is 1-5 with an average of 3.6, so a rating shift of 1.5 is enough to lift an average-rated movie to the top of the scale. On the other hand, the bandwagon attack is quite comparable, despite having a relatively small knowledge requirement. All that is necessary for an attacker is to identify a few items that are likely to be rated by many users. Our results on the effectiveness of the average and random attacks (provided in greater detail in [Burke et al. 2005]) agree with those of Lam and Reidl [2004], confirming their effectiveness against the user-based algorithm.

Recall that for the segment attack we are assuming the maximum benefit to the attacker will come when targeting likely buyers rather than random users. We can assume that likely buyers will be those who have previously bought similar items (we will disregard portfolio effects that are not prevalent in consumer goods, as opposed to cars, houses, etc.) The task therefore for the attacker is to associate her product with popular items considered similar. The users who have a preference for these similar items are considered the target segment.

[4]http://www.the-numbers.com/movies/records/inflation.html
[5]http://www.imdb.com/boxoffice/alltimegross

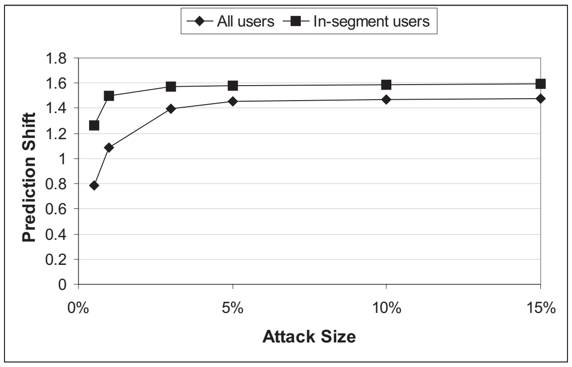


Fig. 5. Prediction shift results for the Horror Movie segment attack against the user-based algorithm with 3% filler size.

The task for the attacker in crafting a segment attack is therefore to select items similar to the target item for use as the segment portion of the attack profile I_S. In the realm of movies, we might imagine selecting movies of a similar genre or movies containing the same actors.

If we evaluate the segmented attack based on its average impact on all users, there is nothing remarkable. The attack has an effect but does not approach the numbers reached by the average attack. However, we must recall our market segment assumption: namely, that recommendations made to in-segment users are much more useful to the attacker than recommendations to other users. Our focus must therefore be with the "in-segment" users, those users who have rated the segment movies highly and presumably are desirable customers for pushed items that are similar: an attacker using the Horror segment would presumably be interested in pushing a new movie of this type.

To build our segmented attack profiles, we identified the user segment as all users who had given above average scores (4 or 5) to any three of the five selected horror movies, namely, *Alien*, *Psycho*, *The Shining*, *Jaws*, and *The Birds*.[6] For this set of five movies, we then selected all combinations of three movies that had at least 50 users support, and chose 50 of those users randomly and averaged the results.

While the segmented attack does show some impact against the system as a whole, it truly succeeds in its mission: to push the attacked movie precisely to those users defined by the segment. Clearly the segmented attack has a bigger impact than any other attack we have previously examined at small filler sizes. As previous results have shown, prediction shift results show that the segmented attack is more effective against in-segment users than even the more knowledge-intensive average attack[Burke et al. 2005a; 2005b]. These results were also confirmed with a different segment based on movies starring Harrison Ford, which for the sake brevity we do not include in this paper.

6.1.2 *Vulnerability Against Nuke Attacks.* Previous researchers have assumed that nuke attacks would be symmetric to push attacks, with the only difference being the rating given to the target item and hence the direction of the impact on predicted ratings. However, our results show that there are some interesting differences in the effectiveness of models depending on whether they are being used to push or nuke an item. The experiments below show results for nuke variations of the average and random attacks, and in addition, an attack model tailored specifically for this task, namely the *love/hate* attack.

In the love/hate attack, a number of filler items are selected and given the maximum rating while the target item is given the minimum rating. For this experiment we selected 3% of the movies randomly as the filler item set. An advantage of the love/hate attack is that it requires

[6]The list was generated from on-line sources of the popular horror films: http://www.imdb.com/chart/horror and http://www.filmsite.org/afi100thrillers1.html.

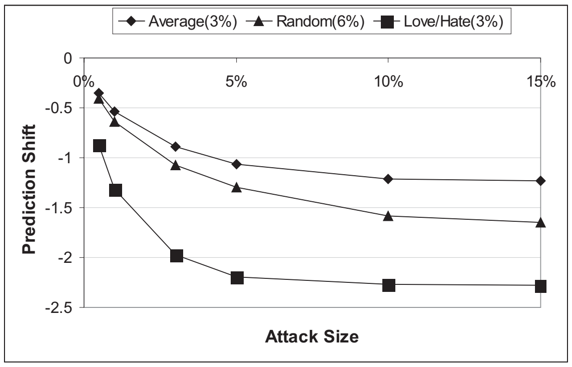


Fig. 6. Prediction Shift results for Nuke Attack

no knowledge about the system, users, or rating distribution, yet as we show it is the most effective nuke attack against the user-based algorithm.

Figure 6 shows the experimental results for all nuke attack models discussed above. Despite the minimal knowledge required for the love/hate attack, this attack proves to be the most effective at nuking items of these attacks. Among the other attacks, the random attack actually surpasses the average attack, which was not the case with in the push results discussed above.

The asymmetry between these results and the push attack data is somewhat surprising. For example, the random attack produced a positive prediction shift slightly over 1.4 for a push attack of 5%, which is much less effective than the higher knowledge average attack (1.7). However when used to nuke an item, this model is the more effective than the higher knowledge average attack. For pushing items, the average attack was the most successful, while it proved to be one of the least successful attacks for nuking items.

As previous work has shown and these results confirm, there are significant vulnerabilities in collaborative filtering [Lam and Reidl 2004; O'Mahony et al. 2004]. While we have concentrated above on user-based collaborative recommendation, other work has shown similar vulnerabilities exist in other model-based collaborative recommendation techniques as well [Mobasher et al. 2005; Mobasher et al. 2006]. Although we evaluate the effectiveness of our detection schemes at protecting user-based collaborative filtering, these same techniques are applicable to securing other forms of collaborative recommendation systems as well.

6.2 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 how informative the attributes are.

6.2.1 *Information Gain vs. Attack Model.* 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 IV 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

Table IV. Information gain for the detection attributes against push attacks.

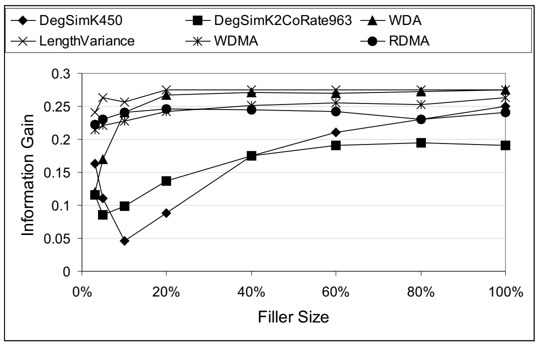
Attribute	Random		Average		Bandwagon		Segment	
	Info Gain	Rank	Info Gain	Rank	Info Gain	Rank	Info Gain	Rank
DegSimK450	0.161	6	0.116	9	0.180	5	0.180	12
DegSimK2CoRate963	0.103	9	0.177	7	0.101	9	0.213	10
WDA	0.233	4	0.229	3	0.234	4	0.246	5
LengthVariance	0.267	1	0.267	1	0.267	1	0.269	3
WDMA	0.248	2	0.238	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
FillerAverageCorrelation*	0.128	8	0.104	11	0.125	8	0.190	11
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
TargetModelFocus	0.194	5	0.185	6	0.174	6	0.176	13

Table V. Information gain for the detection attributes against nuke attacks.

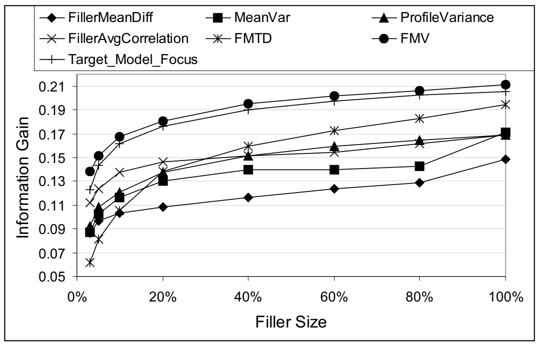
Attribute	Random		Average		Love/Hate	
	Info Gain	Rank	Info Gain	Rank	Info Gain	Rank
DegSimK450	0.161	6	0.111	10	0.155	11
DegSimK2CoRate963	0.104	10	0.176	5	0.213	9
WDA	0.234	4	0.229	4	0.253	5
LengthVariance	0.267	1	0.267	1	0.267	3
WDMA	0.248	2	0.238	2	0.244	8
RDMA	0.240	3	0.229	3	0.249	7
FillerMeanDiff*	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
FillerAverageCorrelation*	0.147	7	0.120	9	0.094	13
FMTD*	0.138	8	0.154	7	0.276	1
FMV*	0.077	13	0.069	13	0.276	1
TargetModelFocus	0.190	5	0.162	6	0.267	4

from [Chirita et al. 2005]: 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 V 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 benefit of the attributes is pretty consistent between push and nuke attacks. A closer inspection of the information gain reveals that the benefit of the generic attributes is nearly identical. For the model based attributes, the FillerMeanDiff, MeanVar, FillerAverageCorrelation, 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



(a) Generic Attributes



(b) Model-Specific Attributes

Fig. 7. Comparison of information gain vs. filler size for 5% push attacks.

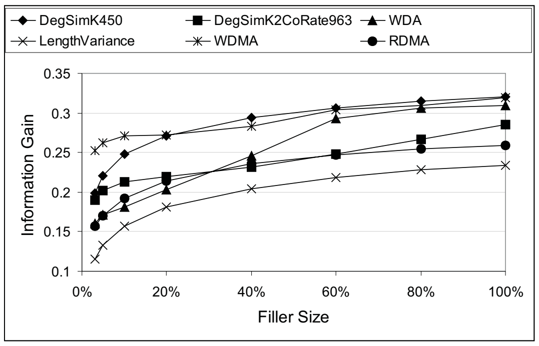
the other hand, since the group attack model-specific attributes select all items with the user's minimum rating as the suspected target, the models also mask some of the more extreme variability in real user ratings thus decreasing the information gain for detecting nuke attacks targeting a single item.

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 in detecting the segment push attack and the love/hate nuke attack reflects this intuition with the only major differences occurring in the ProfileVariance, FillerAverageCorrelation, and TargetModelFocus attributes. ProfileVariance and FillerAverageCorrelation become 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.

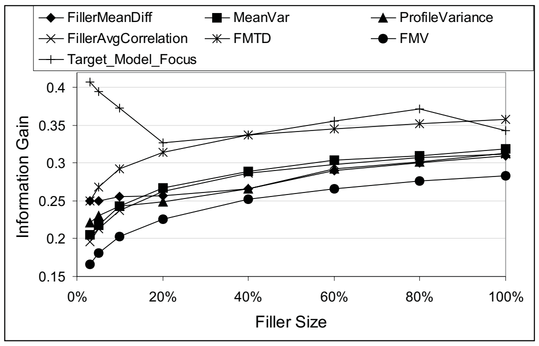
6.2.2 *Information Gain vs. Filler Size.* For our filler size analysis each attack was inserted targeting a single movie at 5% attack size and a specific attack model. Each of the test movies was attacked for each of the attack models, with the model-specific attributes created to look for either a push or nuke attack depending on the purpose of attack injected. The results reported are averaged over the 50 test movies and 4 push models (or 3 nuke models).

Figure 7 depicts the average information gain of the attributes across all push models for an attack size of 5% for various filler sizes. The two charts depict the information gain associated with the generic attributes (left) and model-based attributes (right). As the results show, in general the highest information gain for the attributes is obtained at the maximum filler size. This supports our hypothesis that as the rating sample associated with a profile grows, abnormal rating trends become more apparent. For the model based attributes in particular this recognition grows logarithmically such that attack profiles with small filler sizes are hard to distinguish from authentic profiles. For the generic attributes, a similar trend is seen for the WDA, WDMA, RDMA, and LengthVariance attributes; although the information gain degrade is much less substantial than the model based attributes for all except the WDA attribute. The two attributes based on DegSim however, exhibit a dip in information gain for filler sizes between 5% and 40%. The information gain of these two attributes will be evaluated in much greater detail in Section 6.2.3.

Next we present our results on the information gain of these attributes against nuke attacks over various filler sizes. Figure 8 depicts the average information gain of the attributes across all nuke models for an attack size of 5% for various filler sizes. The two charts depict the infor-



(a) Generic Attributes



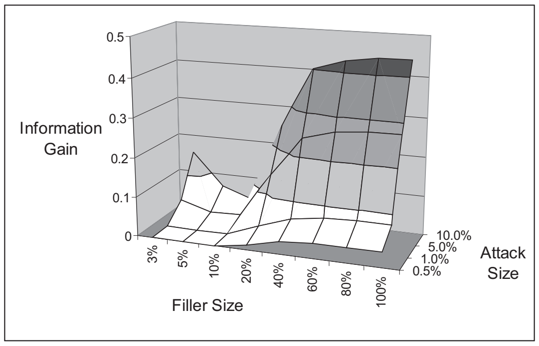
(b) Model-Specific Attributes

Fig. 8. Comparison of information gain vs. filler size for 5% nuke attacks.

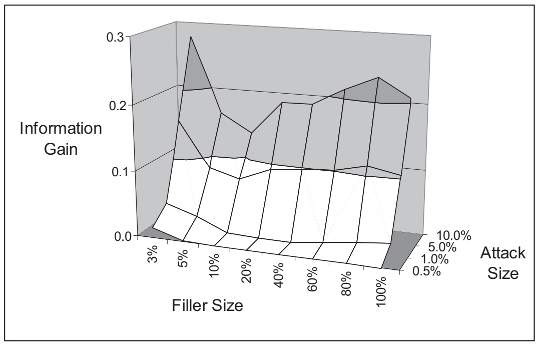
mation gain associated with the generic attributes (left) and model-based attributes (right). Like the push results, in general the highest information gain for the attributes is obtained at the maximum filler size due to unusual trends becoming more apparent with a larger sample, with the notable exception of the target model focus attribute. For small filler sizes, given the rarity of very low ratings in the MovieLens dataset, the likely target of a nuke attack becomes much more apparent as reflected in the target model focus' information gain. Even for the small profile rating sample associated with these low filler sizes, the intra-profile trends make this attribute by far the most informative. Another interesting result is that while in general the generic attributes become slightly less informative than they were for push attacks at low filler sizes, the model based attributes are general more informative across the entire filler range than they were for push attacks. This trend is primarily due to low ratings being much more uncommon than high ratings in our dataset thus making the models more accurate at selecting the suspected target item to match the actual target item.

6.2.3 *Information Gain Surface Analysis.* To understand the benefit of the attributes in greater depth, we experimented with the effect 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 9, 10, and 11 show the information gain of the DegSim ($k = 450$) and DegSim' ($k = 2$, $d = 963$) attributes across the dimensions of filler size and attack size, which we term the information gain surface, for each of the push and nuke attacks described above. As these results show, while the averaged results above provide some insight into the benefit of each of these attributes across attack model, the actual information gain of each attribute will vary greatly based on filler size and attack size as well. Furthermore, as these charts show the filler size where an attribute is most informative may also differ across attack models as shown in [Burke et al. 2006a]. The results related to these two attributes are described in more detail here to further explain the reason two attributes based on DegSim are used. A similar analysis was performed for all of the attributes and all of the models and similar trends in variance across the dimensions of filler size, attack size, and attack model were found as well (results not included).

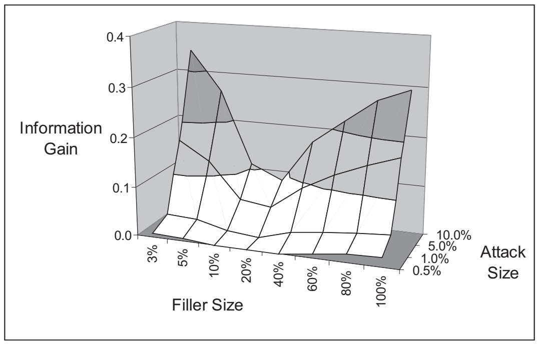
The selection of k and d for the DegSim and DegSim' were selected by experimentally evaluating the information gain of each of these parameters through a comparison of information gain surfaces as shown in Figures 9, 10, and 11. While there were areas where other combinations performed slightly better, the combination of the two attributes based on DegSim ($k = 450$) and DegSim' ($k = 2$, $d = 963$) provided the most complete coverage across the entire filler size range and attack sizes from .5% to 10%. Conceptually the DegSim attribute captures the profiles similarity to about half of the system's profiles, while the DegSim' attribute



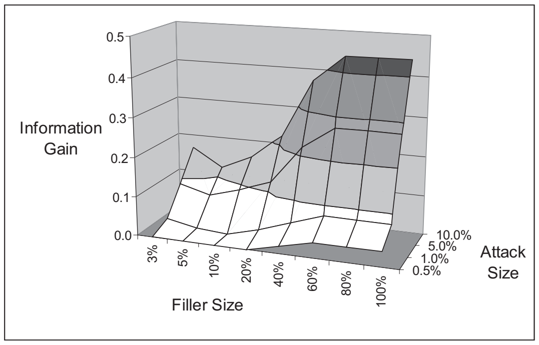
(a) Random Attack - DegSimK450



(b) Random Attack - DegSimK2CoRate963



(c) Average Attack - DegSimK450



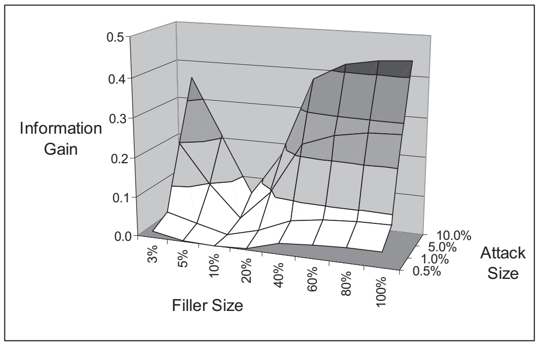
(d) Average Attack - DegSimK2CoRate963

Fig. 9. Comparison of information gain vs. filler size and attack size for random and average push attacks.

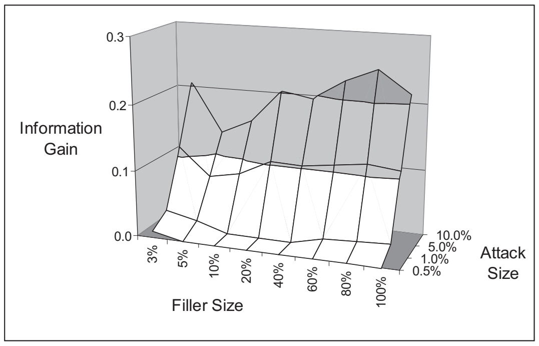
captures the profile's similarity to its immediate neighbors corate discounted by half the items. The actual value of these parameters will likely vary with datasets, but the selection will likely reflect the same selection criteria.

Specifically the DegSim attribute was found to provide good information gain at filler sizes higher than 40% for attacks that use random ratings for filler items such as the random (both push and nuke), and bandwagon attacks. Intuitively for random and bandwagon attack this would occur due to these profiles decreasing the similarity to the majority of profiles due to not correlating with item averages. At low filler sizes, however, the opposite is true; attacks that encode more knowledge in either their filler items or in the segment (I_S) are easier to detect by the DegSim attribute. Specifically the DegSim attribute is highly informative at low filler sizes for the average (push and nuke), bandwagon, and segment attacks. Conceptually this is due to the few items in the profiles being given very similar ratings. The love/hate attack is interesting in this regard, in that at low attack sizes this generalization holds, but at attack sizes greater than 5% it becomes more informative than the corated version. The reason for this is likely due to the similarity between attack profiles combined with the sparsity of minimum ratings making the attack profiles have a greater influence on the attribute as more profiles are added. For all attacks the non-corated DegSim attribute struggles across the mid filler sizes that correspond to the number of ratings provided by the majority of users in our dataset.

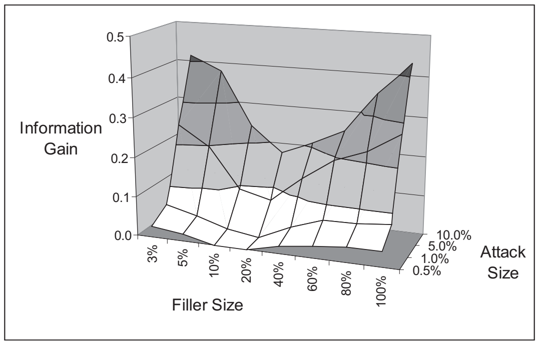
The corating of the DegSim' attribute helps address the weaknesses in coverage of the DegSim attribute. Across all 7 attack models studied in this paper (4 push and 3 nuke), the DegSim' attribute provides significantly better information gain for mid-range filler sizes (between 10% and 40%). At high filler sizes the DegSim' attribute is most effective for the average, segment, and love/hate attacks. For all three cases this occurs due to the similarity



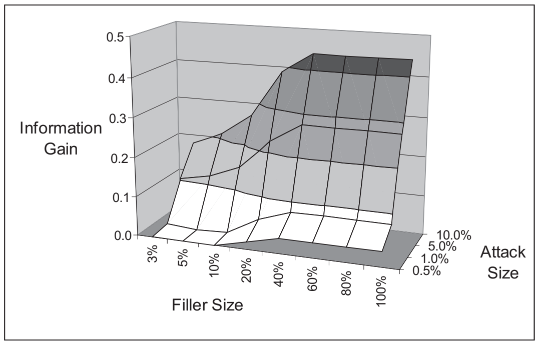
(a) Bandwagon Attack - DegSimK450



(b) Bandwagon Attack - DegSimK2CoRate963



(c) Segment Attack - DegSimK450



(d) Segment Attack - DegSimK2CoRate963

Fig. 10. Comparison of information gain vs. filler size and attack size for bandwagon and segment push attacks.

between the immediate neighbors (usually also attack profiles) being far higher than in most authentic profiles when adjusted for the number of corated items. For segment and love/hate attack this trivially true, since all corated items would be given the same rating. For average attack this occurs since the attack models do not deviate significantly from item averages, whereas real users personal tastes tend to vary on at least some subset of items.

6.3 Classification Performance

In this section we analyze how well a classifier built on the attributes described above performs at detecting attack profiles. The results below we examine the performance of the classifier across the dimensions of filler size and attack size for the various push attacks described above followed by a similar analysis of the nuke attack models.

The classification 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.

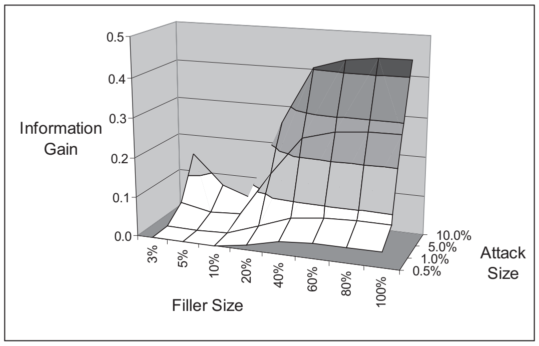
For these experiments we use 15 detection attributes:

—6 generic attributes: WDMA, RDMA, WDA, Length Variance, DegSim (k = 450), and DegSim' (k = 2, d = 963);

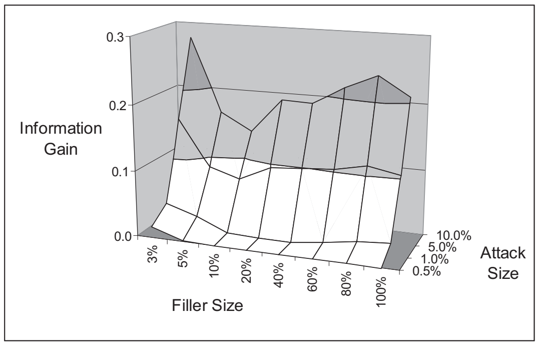
—6 average attack model attributes (3 for push, 3 for nuke): Filler Mean Variance, Filler Mean Difference, Profile Variance;

—2 segment attack model attributes (1 for push, 1 for nuke): FMTD; and,

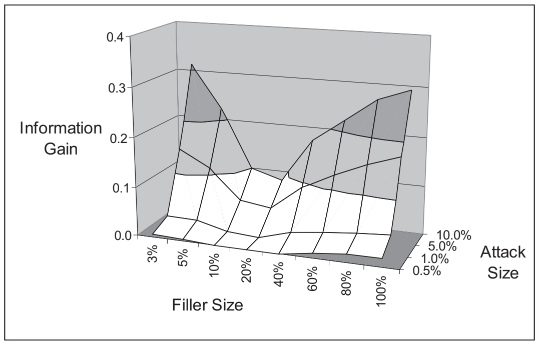
—1 target detection model attribute: TMF.



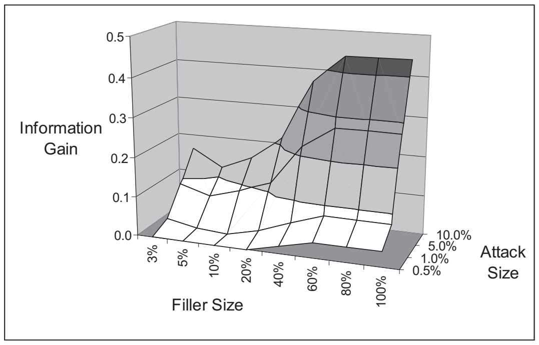
(a) Random Attack - DegSimK450



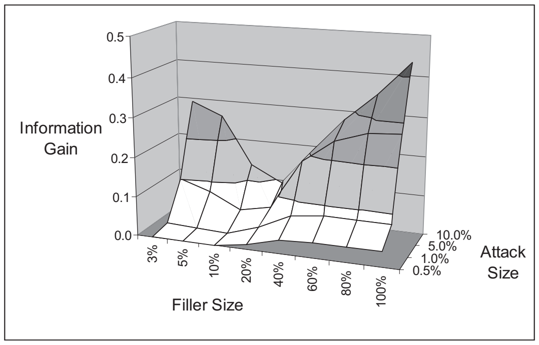
(b) Random Attack - DegSimK2CoRate963



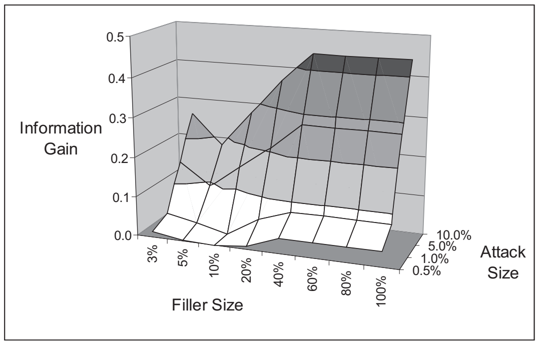
(c) Average Attack - DegSimK450



(d) Average Attack - DegSimK2CoRate963



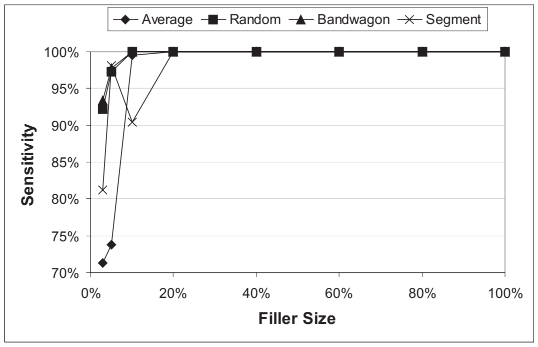
(e) Love/Hate Attack - DegSimK450



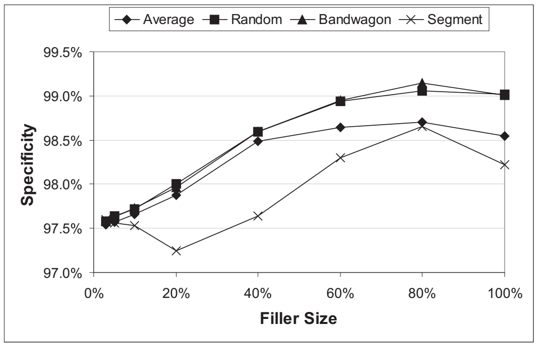
(f) Love/Hate Attack - DegSimK2CoRate963

Fig. 11. Comparison of information gain vs. filler size and attack size for nuke attacks.

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 due to the high percentage of attack profiles that make up the training set (10.5% total across the 19 training attacks).



(a) Sensitivity



(b) Specificity

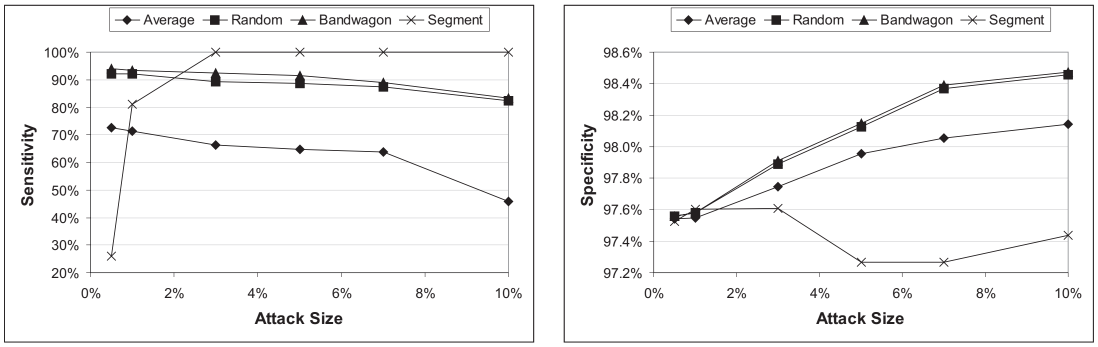
Fig. 12. Comparison of classifier performance vs. filler size for 1% push attacks.

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.

Below we present the classification performance results for push and nuke attacks. For each of these attack groups, we analyze the results across the dimensions of filler size and attack size. This is followed by a brief discussion of the impact of any misclassifications on the recommender's predictive accuracy.

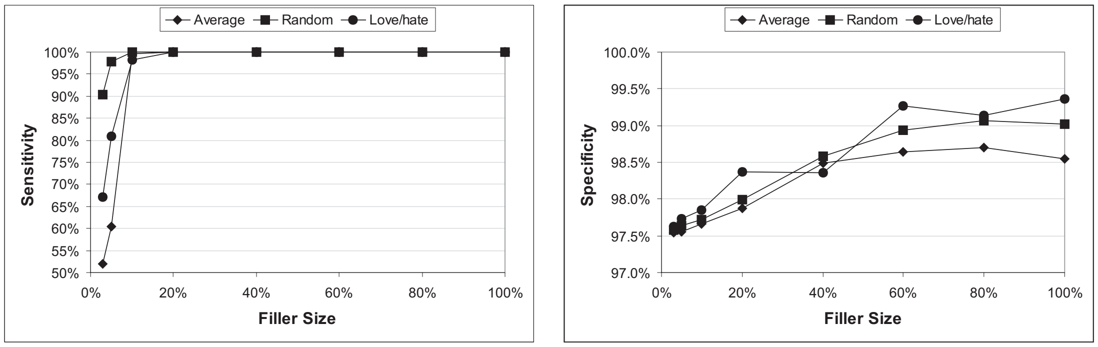
6.3.1 *Classification Performance Against Push Attacks.* To analyze the classification performance vs. the dimension of filler size, attack size was fixed at 1% and each of the push attacks were inserted individually across a range of filler sizes. Figure 12 compares the detection capabilities of our algorithm for each of the push attacks at 1% attack size across various filler sizes. As the sensitivity results show, the random and bandwagon attacks are easily detected at even low filler sizes. This is not particularly surprising given that these attacks encode very little knowledge in their attack models, thus making their lack of similarity with authentic profiles more apparent. The average attack however, is more difficult to detect at low filler sizes, but as the profile size increases, the abnormally high correlation to item averages becomes more apparent making it easy to detect as well. The segment attack also is more difficult to detect a low filler sizes as it is unclear whether the similarity of ratings is due to lack of differentiation or a more abnormal pattern associated with an attack. As the specificity results show, very few authentic profiles get misclassified for any of the attack models across the entire filler range.

A comparison of these same performance metrics vs. attack size is shown in Figure 13. For this experiment filler size was fixed at 3% and attack size was varied. The filler size of 3% was picked pessimistically as it was the profile size that was hardest to detect in our results above. The results show some interesting trends. The sensitivity of the classifier for average, random, and bandwagon attack is best at low attack sizes, but slowly degrades as attack size gets bigger. The specificity against these attacks, however, improves slightly, but significantly, with attack size. The reason for this is likely due to the saturation of attack profiles at high attack sizes making the abnormal rating trends seem more common, which has also prevented more eccentric authentic profiles from being classified as attacks. The sensitivity results for the segment attack display a strikingly different trend than the other attacks. The identification of these attacks gets progressively better as the biased profiles saturate the database. A key reason for this difference from the other attacks is the much smaller variance between attack



(a) Sensitivity (b) Specificity

Fig. 13. Comparison of classifier performance vs. attack size for push attacks at 3% filler size.



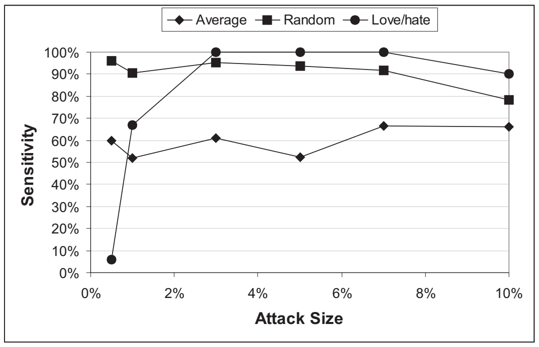
(a) Sensitivity (b) Specificity

Fig. 14. Comparison of classifier performance vs. filler size for 1% nuke attacks.

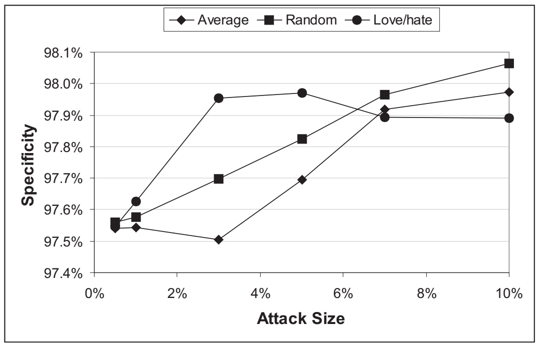
profiles created by this model. As our previous results showed related to Figure 10, for this model the information gain of the DegSim attribute for low filler sizes is much higher than any other push attack. However it is also important to note that for very low attack sizes (.5%) the identification of the segment attack is very poor.

6.3.2 *Classification Performance Against Nuke Attacks.* To analyze the classification performance against nuke attacks, we used the same methodology as the push attack experiments above. For analyzing filler size, attack size was fixed at 1% and each of the nuke attacks were inserted individually at various filler sizes. Figure 14 compares the detection capabilities of our algorithm for each of the nuke attacks at 1% attack size across various filler sizes. As the sensitivity results show, the identification trends for the random and average attacks are very similar to the results for detecting these two attacks models when used for push attacks. The main difference is a larger decrease in accuracy at low filler sizes for both models. The love/hate attack like the other attacks also proves more difficult to detect at low filler sizes.

Next we fix filler size at 3% and vary nuke attack size, the results shown in Figure 15. For random attack the same trends in sensitivity emerge, a slight degrade in performance as attack size grows. For average attack, the sensitivity remains relatively stable, although slightly lower for small attack sizes. The similarities between the love/hate and segment attack become more apparent in the love/hate sensitivity results. Like the segment attack, the love/hate attack is very difficult to detect at low attack sizes, but as the attack size grows, the information gain of the TargetModelFocus attribute becomes more pronounced making detection far easier. The



(a) Sensitivity



(b) Specificity

Fig. 15. Comparison of classifier performance vs. attack size for nuke attacks at 3% filler size.

specificity results for all models are similar to the push results seen above.

6.3.3 *Impact Of Misclassifications.* In the context of detection, the effect of specificity being less than 100% means some authentic users are not being included in collaborative prediction. From the results above, it is clear that although very high, the specificity of the profile classifier is not 100%, meaning some authentic profiles will not be used in prediction. Since the accuracy of collaborative prediction often depends on the size of the user base, one possible impact of misclassifying authentic profiles would be lower predictive accuracy. In evaluating the impact of adding the profile classifier to the recommender, of interest is the change in predictive accuracy when no attack is present. This is evaluated experimentally by comparing the predictive accuracy of the recommender with and without the profile classifier.

For this evaluation the authentic users that made up the training set for the classifier were used as the training set for the base recommender and the second half of the data was used as the test set for both implementations. When the classifier was applied to this set the specificity was just over 97% indicating about 3% of the authentic profiles were not used in creating predictions. As described in the Section 5.2, we introduce the ΔMAE metric to evaluate this impact. To determine this, the predictive accuracy of each algorithm was measured by MAE with and without the detection algorithm. The system without detection had an MAE of 0.7742 and with detection 0.7772. Thus the ΔMAE was -0.003, which indicates a slight degrade in performance, however this difference is not statistically significant based on a 95% confidence interval. Thus detection can be added without a significant impact to predictive accuracy.

6.3.4 *Attack Model Identification.* 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 [Quinlan 1993] was constructed using the Weka data mining package [Witten and Frank 2005] which resulted in the tree shown in Figure 16. 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.

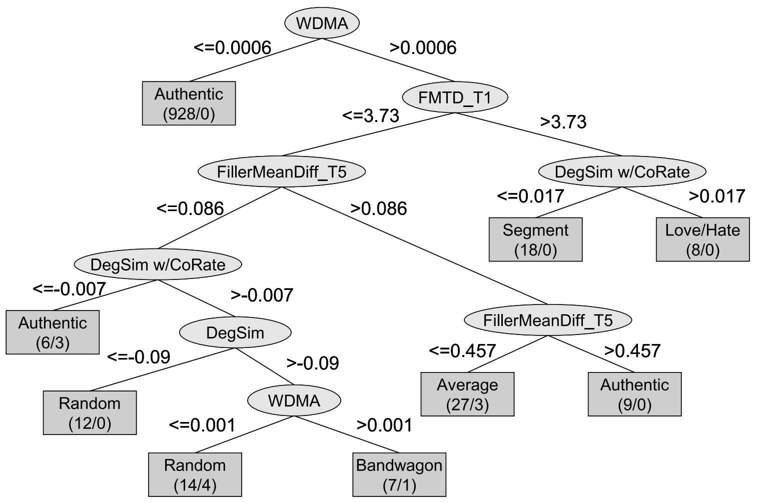


Fig. 16. C4.5 decision tree used to identify attack type. T1 and T5 signify the model based attributes trained for nuke or push attacks respectively.

6.4 Robustness Analysis

The results above show that both detection algorithms have some success in identifying profiles as belonging to an attack. While this is promising, the real proof of the concept is in 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 5.2. 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.

The detection algorithm introduced in Chirita et al. [2005] was also implemented for comparison purposes (with $\alpha = 10$), and run on the test set described above. Comparative results are shown below. It should be noted that there are a number of methodological differences between the results reported in [Chirita et al. 2005] and those shown here. The attack profiles used in [Chirita et al. 2005] 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.

6.4.1 *Robustness Comparison Against Push Attacks.* Figure 17 shows the average prediction shift for the recommendation algorithm unaided and with the attack profiles discounted (or rejected) by either the model-specific or Chirita algorithms. This figure shows only 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 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."

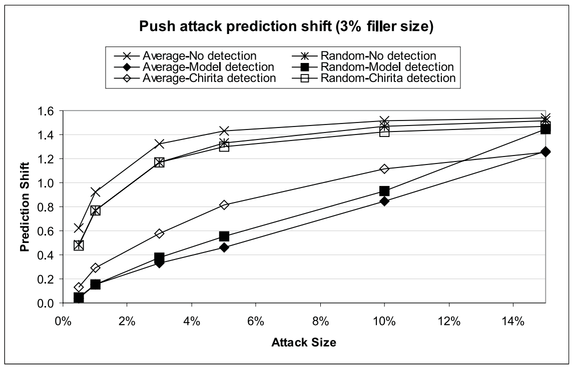


Fig. 17. Prediction shift for the recommender system vs. average and random attacks with and without attack detection.

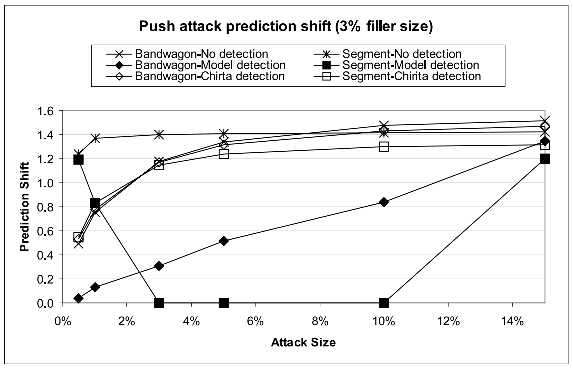


Fig. 18. Prediction shift for the recommender system vs. bandwagon and segment attacks before and after attack detection.

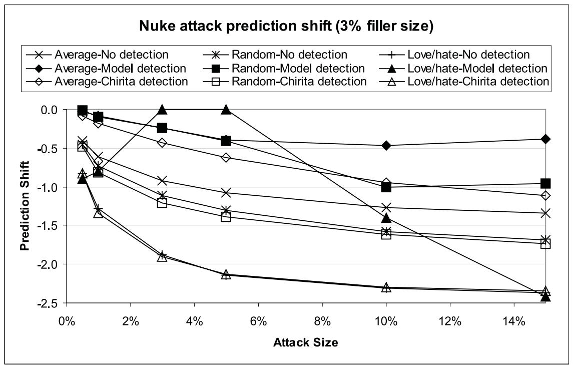


Fig. 19. Prediction shift for the recommender system vs. nuke attacks before and after attack detection.

Figure 18 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.

6.4.2 *Robustness Comparison Against Nuke Attacks.* Finally, Figure 19 shows prediction shift results with the nuke attack. The low-knowledge love/hate attack is quite 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.

7. DEFENSE AGAINST UNKNOWN ATTACKS

As our results above and prior work have shown, attacks that closely follow one of the models mentioned above can be detected and their impact can be significantly reduced [Burke et al. 2006b; Mobasher et al. 2006]. A more challenging problem will likely be ensuring robustness against unknown attacks as profile classification alone may be insufficient. Unlike traditional classification problems where patterns are observed and learned, in this context there is a competitive aspect since attackers are motivated to actively look for ways to beat the classifier. In the section below we examine potential ways an attacker might try to modify their attacks to avoid detection and evaluate the detection classifier's ability to detect such attacks. This is followed by a discussion of techniques that may be used to improve robustness, including additional detection techniques. Given the competitive dynamic of this problem, a solution will likely have to combine multiple detection approaches in order to ensure robustness. Below we outline two such approaches based on rating distribution and time series analysis . We envision combining the techniques above with other detection techniques to create a comprehensive detection framework.

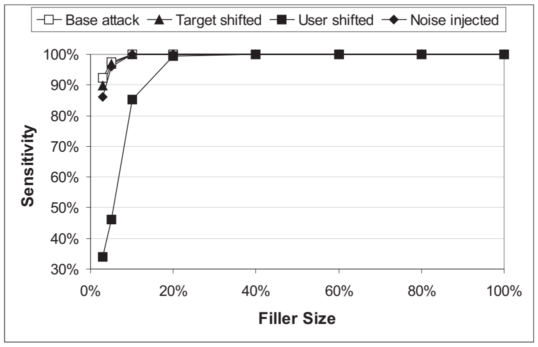
7.1 Obfuscated Attack Models

For systems with detection schemes in place, attackers will be motivated to deviate from these known models to avoid detection. To explore this problem, we have examined three ways existing attack models might be obfuscated to make their detection more difficult: *noise injection*, *user shifting* and *target shifting* [Williams et al. 2006].

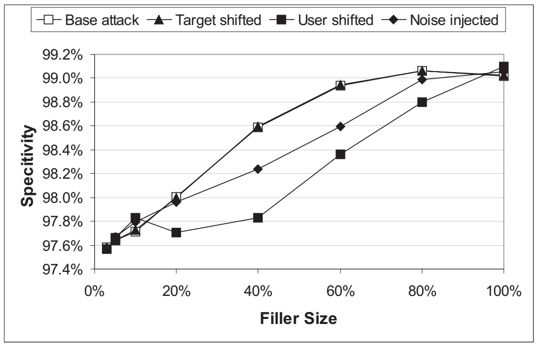
7.1.1 *Noise Injection.* – involves adding a Gaussian distributed random number multiplied by α to each rating within a set of attack profile items O_{ni}; where O_{ni} is any subset of $I_F \cup I_S$ to be obfuscated and α is a constant multiplier governing the amount of noise to be added. This noise can be used to blur the profile signatures that are often associated with known attack models. For example, the abnormally high correlation that is associated with profiles of an average attack could be reduced by this technique while still maintaining a strong enough similarity between the profiles and real system users.

7.1.2 *User Shifting.* – involves incrementing or decrementing (shifting) all ratings for a subset of items per attack profile in order to reduce the similarity between attack users. More formally, for all items i in O_s, $r'_{i,u} = r_{i,u} + \mathit{shift}(u, O_s)$ where O_s is any subset of $I_F \cup I_S$ to be obfuscated, $r_{i,u}$ is the original assigned rating given to item i by attack profile u, $r'_{i,u}$ is the rating assigned to item i by the obfuscated attack profile u, and $\mathit{shift}(u, O_s)$ is a function governing the amount to either increment or decrement all ratings within set O_s for profile u. This technique results in a portion of the base attack model ratings deviating for each attack profile. As a result, the distribution signature for the profile can deviate from the profile signature usually associated with the base attack model. This technique can also be used to reduce the similarity between attack profiles that often occurs in the reverse-engineered attack models.

7.1.3 *Target Shifting.* – for a push attack is simply shifting the rating given to the target item from the maximum rating to a rating one step lower, or in the case of nuke attacks



(a) Sensitivity



(b) Specificity

Fig. 20. Classification results for 1% obfuscated random push attacks.

increasing the target rating to one step above the lowest rating. Although a minor change, this has a key effect. Since all reverse-engineered models dictate giving the target item the highest or lowest rating, any profile that does not include these ratings is likely to be less suspect. Naturally, profiles that are not as extreme in their preference will generate less bias in the attacked system (and our experiments bear this out). However, in many practical scenarios, for example, trying to push an item with low ratings, a target shifted attack may be almost as effective as an ordinary one.

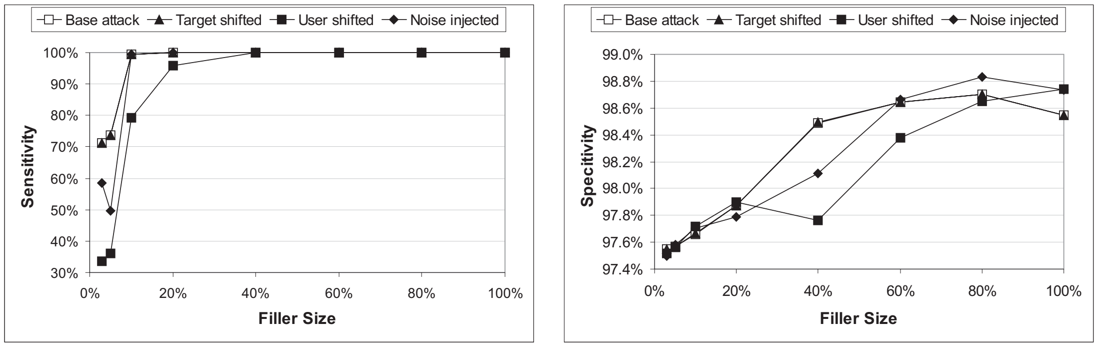
While there are numerous ways a profile may be constructed to avoid detection, we focus on these to illustrate the detection challenges that can occur with even minor changes to existing models.

7.2 Experiments With Obfuscated Attack Models

To evaluate the obfuscation methods discussed above we have examined these techniques on the average and random attack models for both push and nuke attacks. For the user shift technique, for both models we shifted all of the filler items, and we used a Gaussian distributed random number for shift amount. For the noise injection technique we add noise to all of the filler items using a Gaussian distributed random number multiplied by 0.2.

7.2.1 *Classification Performance Against Obfuscated Attacks.* In our first set of experiments we compare the attack detection model's ability to detect the obfuscated attacks compared to the base attacks (standard non-obfuscated attacks). As Figure 21 depicts the target shifting obfuscation has little impact on the detection of average attack. The user shifted and noise injection techniques were much more successful particularly at lower filler sizes where the recall degraded over 37% for average attack. (Results for the random attack were similar.) Thus as the number of ratings increase, the patterns that distinguish an attacker would become more apparent. The same trends emerged for both average and random nuke attacks (results omitted). Recall of the nuke average attack dropped by over 30% for user shifting and noise injection, while recall of random attack degraded by over 50%. Once again target shifting alone was not particularly effective at disguising either of these attacks. Target shifting may be more significant for models such as segment attack since attributes designed to detect these attacks focus on target/filler rating separation [Mobasher et al. 2006]. We intend to investigate obfuscating these types of attacks in future work.

7.2.2 *Robustness Against Obfuscated Attacks.* We also examined the impact on prediction shift due to deviating from the reverse-engineered attacks to avoid detection. We compared the prediction shift of base attacks and obfuscated attacks on a system without detection. Figure 22 depicts the maximum prediction shift found for each attack across all filler sizes with the black bars capturing the results against a system without detection and the gray



(a) Sensitivity (b) Specificity

Fig. 21. Classification results for 1% obfuscated average push attacks.

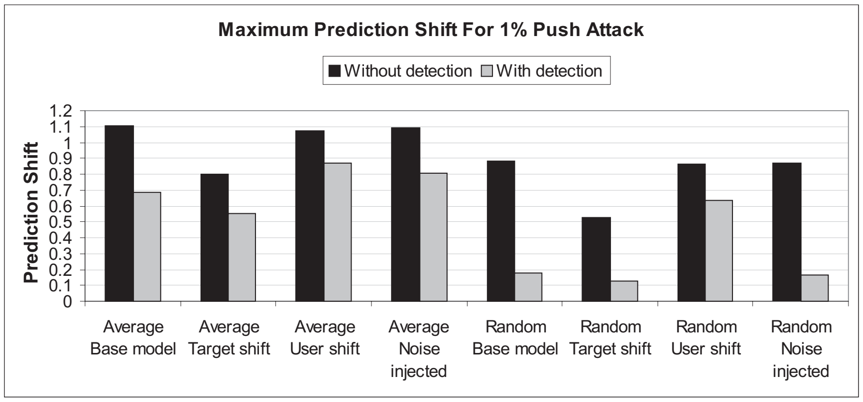
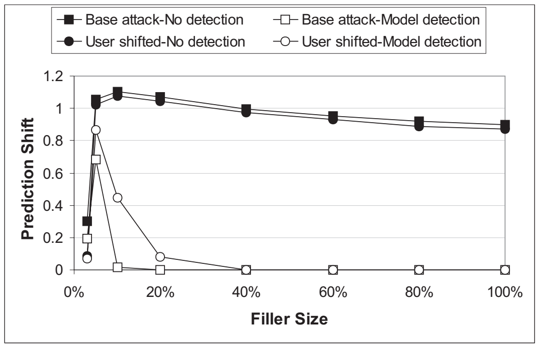


Fig. 22. Maximum prediction shift for each push attack at 1% attack size across all filler sizes.

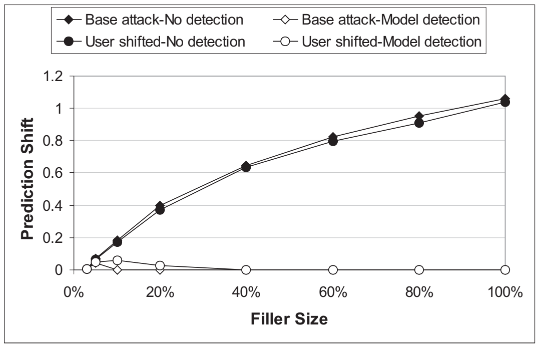
bars the results against a system with detection (we used filler sizes between 3% and 10%). As the results show, the user-shifted and noise-injected versions are nearly as effective as the non-obfuscated versions without detection for both attack models at their most effective filler sizes. This means an attacker can mount an effective attack using the obfuscation techniques with reduced chance of detection.

For both average and random attacks the user shifting obfuscation is the most effective against a system that uses detection as seen in the gray bars in Figure 22. Noise injection, however, is more effective than the base attack against a system with detection for average attack, but the obfuscated version is slightly less effective for random attack. Intuitively, this makes sense since the random attack already is an attack based on noise, and its lack of correlation to item averages is one of the features that aides in its detection; additional noise being added is unlikely to improve the correlation.

The classification and prediction shift results indicate that, when combined with detection, average and random attacks at lower filler sizes pose the most risk. To reduce the effect of these attacks at lower filler sizes, one approach would be to discount profiles that have fewer items in their profile. Herlocker *et al.* introduced such a variation in [Herlocker et al. 1999] that discounts similarity between profiles that have fewer than 50 co-rated items by $n/50$ where n is the number of co-rated items. While this modification was proposed originally to improve prediction quality, it has some interesting effects on changing the characteristics of effective attacks as well. As Figure 23 shows, while the average and random attacks are about as effective against the co-rate discounted version as they are against the basic version at high



(a) Basic kNN



(b) Corate discounted kNN

Fig. 23. Prediction shift results for 1% obfuscated average push attacks.

filler sizes, at low filler sizes their impact is far less. When combined with the detection model outlined above the largest prediction shift achieved by any of the attacks described above is only .06 compared to the .86 shift achieved against basic kNN. This combination may not be as effective against attacks that focus specifically on popular items, since they are designed to increase the likelihood of co-rating, but it does appear to add significant robustness for the attacks studied in this paper.

7.3 Anomaly Detection

In this section we describe two alternate approaches to attack detection we introduced in [Bhaumik et al. 2006] that do not rely on profile classification. Instead these techniques are based on an item-based approach to detection that identifies what items may be under attack based on rating activity related to the item. Below we present two **Statistical Process Control** (SPC) techniques for detecting items which are under attack: X-bar control limit and Confidence Interval control limit. Our second approach attempts to identify time intervals of rating activity that may suggest an item is under attack.

Statistical process control charts have been used in manufacturing to detect whether a process is out-of control [Shewart 1931]. In general, a SPC is composed of two phases. In the first phase the technique estimates the process parameters from historical events and then uses these parameters to detect out-of-control anomalies for recent events. Recommender systems which collect ratings from users can also be thought of as a similar process. The rating patterns can be monitored for abnormal rating trends that deviate from the past distribution of items. Our results show that both SPC approaches work well in identifying suspicious activity related to items which are under attack. For time interval detection, our results show this technique performs well at identifying suspicious time intervals, over which an item is under attack.

In this section we describe two SPC techniques for detecting items under attack, as well as a time-series technique for detecting time intervals an item is under attack. The detailed algorithms of these techniques, we have shown to be successful for detecting items and intervals under attack, appear below [Bhaumik et al. 2006]. These techniques offer an alternate approach to detection that can likely be combined with the profile detection approach discussed above, for a more comprehensive defense for collaborative systems.

7.3.1 *Statistical Process Control.* SPC is often used for long term monitoring of feature values related to the process. Once a feature of interest has been chosen or constructed, the distribution of this feature can be estimated and future observations can be automatically monitored. Control charts are used routinely to monitor quality in SPC. Figure 24 displays an example of a control chart in the context of collaborative filtering where observations are average rating of items, which we assume are not under attack. Two other horizontal lines,

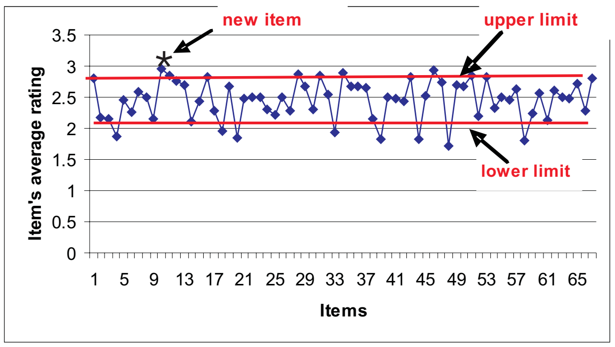


Fig. 24. Example of a control chart

called upper limit and lower limit are chosen so that almost all of the data points will fall within these limits as long as there is no threat in the system. This figure depicts that a new item's average rating is outside the upper limit. This indicates an anomaly that may be an attack. In the following section we describe two different control limits as our detection scheme.

X-bar control limit. One commonly used types of control chart is the X-Bar chart which plots how far away from the average value of a process the current measurement falls. According to [Shewart 1931], values that fall outside of three sigma standard deviation of the average have a valid cause and are labeled as being out of statistical control.

In the context of recommender systems, let us consider the case where we have to identify whether a new item is under attack by analyzing past data. Suppose we have collected k items with similar rating distribution in the same category from our database. Let n_i be the number of users who have rated an item i. According to [Shewart 1931] we can define our upper $(U\bar{x})$ and lower $(L\bar{x})$ control limits as follows:

$$U\bar{x} = \bar{X} + \frac{A*\bar{S}}{c_4(n)\sqrt{n}}$$

$$L\bar{x} = \bar{X} - \frac{A*\bar{S}}{c_4(n)\sqrt{n}}$$

where $\bar{X}$ is the grand mean rating of k items, and $\bar{S}$ is the average standard deviation which can be computed as:

$$\bar{S} = \sum_{i=1}^{k} s_i$$

with s_i being the standard deviation of each item. As n_i is different for each item i, n can be taken as the average of all n_i. The auxiliary function $c_4(n) = \sqrt{\frac{2}{n-1}}\Gamma(\frac{n}{2})\Gamma(\frac{(n-1)}{2})$, where $\Gamma(t)$ is a complete gamma function which is expressed as $(t-1)!$. When $n >= 25$, $c_4(n)\sqrt{n}$ can be approximated by $\sqrt{(n-.5)}$ and A is a constant value which determines the upper and lower limit (SPSS 2002). Thus when A is set to 3 , we get 3-sigma limit. We set $U\bar{x}$ and $L\bar{x}$ as a signal threshold. A new item is likely under attack, if the average rating is greater than $U\bar{x}$ or less than $L\bar{x}$.

Confidence Interval Control Limit. The Central Limit Theorem is one of the most important theorems in statistical theory [Ott 1992]. It states that distribution of the sample mean becomes more normalized as the sample size increases. This means that we can use the normal distribution to describe the sample mean from any population, even non-normal ones, if we have a large enough sample. The general rule of thumb is that you need a sample of at least 30 observations for the Central Limit Theorem to apply (i.e., for the distribution of the sample mean to be reasonably approximated with the normal distribution). A confidence

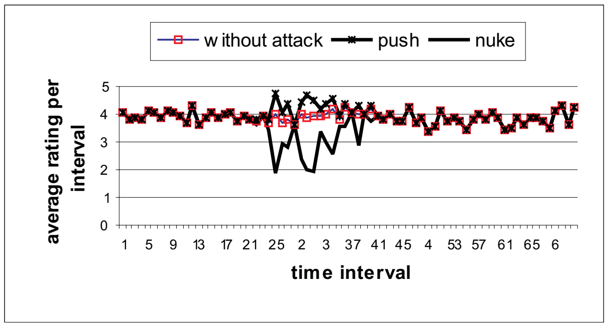


Fig. 25. A time series chart

interval, or interval estimate, is a range of values that contains the population mean with a level of confidence that the researcher chooses. For example, a 95% confidence interval would be a range of values that has a 95% chance of containing the population mean.

Suppose we have collected a set of k items with similar rating distribution in the same category, and $\bar{x}_1$,$\bar{x}_2$,$\cdots$,$\bar{x}_k$ are the mean rating of these k items. The upper ($U\bar{x}$) and lower ($L\bar{x}$) control limits for these sample means can be written as:

$$U\bar{x} = \bar{X} + \frac{C*\sigma}{\sqrt{k}}$$

$$L\bar{x} = \bar{X} - \frac{C*\sigma}{\sqrt{k}}$$

where $\bar{X}$ and σ is the mean and standard deviation of $\bar{x}_i$'s. The value of C is essentially the z-value for the normal distribution. For example, the value is 1.96 for a 95% confidence coefficient.

In a movie recommender system, the upper and lower boundaries of the confidence interval are considered as the signal threshold for push and nuke attacks respectively. If our confidence coefficient is set to .95, we are 95% sure that all the item averages will fall inside these limits and when an average rating of an item is outside of these limits, we consider the ratings related to this item suspicious.

7.3.2 *Time Interval Detection Scheme.* The normal behavior of a recommender system can be characterized by a series of observations over time. When an attack occurs, it is essential to detect the occurrences of abnormal activity as quickly as possible, before significant performance degradation. This can be done by continuously monitoring the system for deviations from the past behavior patterns. In a movie recommender system the owner could be warned of a possible attack by identifying the time period during which abnormal rating behavior occurred for an item. Most anomaly detection algorithms require a set of training data without bias for training and they implicitly assume that anomalies can be treated as patterns not observed before. Distributions of new data are then compared to the distributions obtained from the training data and differences between the distributions indicate an attack.

In the context of recommender systems, we can monitor an item's ratings over a period of time. A sudden jump in an item's mean rating may indicate a suspicious pattern. One can compare the average rating for this item by collecting ratings over a period of time, assuming there are no biased ratings. When new ratings are observed, a system can compare the current average rating to the data collected before. Figure 25 shows an example of a time series pattern before and after an attack in a movie recommender system. The upper and lower curves show the rating pattern of an item after push and nuke attack respectively, whereas the middle curve shows the rating pattern without any attack.

Our time series data can be expressed as a sequence of $\bar{x}_i^t : t = 1, 2, ...$ where t is a time variable and each $\bar{x}_i^t$ is the average rating of an item i at a particular time t. Suppose ${\mu_i}^k$ and σ_i^k are the mean and standard deviation estimated from the ratings collected from a trusted source for the first k-th interval for an item i. Our algorithm is based on calculating the probability of observing the mean rating for the new time interval. If it is outside a pre-specified threshold, then it deviates significantly from the rating population indicating an attack. Now if $\bar{x}_i^t$ is the average rating for an interval t after the k-th interval, our conditions for detecting an attack interval t is:

$$\bar{x}_i^t > {\mu_i}^k + Z_{\frac{\alpha}{2}} \frac{\sigma_i^k}{\sqrt{n}}$$

and

$$\bar{x}_i^t < {\mu_i}^k - Z_{\frac{\alpha}{2}} \frac{\sigma_i^k}{\sqrt{n}}$$

for push and nuke attack respectively. The parameter n is the total number of ratings for the first k-th interval of an item i. The value for $Z_{\frac{\alpha}{2}}$ is obtained from a normal distribution table for a particular value of α. This algorithm essentially detects the time period over which an item is potentially under attack.

7.4 Experiments With Anomaly Detection

In this section, we present an empirical evaluation of the anomaly detection methods described above and show these techniques can be quite successful in identifying items under attack and time periods in which attacks take place. We outline the experimental methodology and metrics used in evaluating these techniques. Finally, we present a selection of our results related to anomaly detection which are extended in [Bhaumik et al. 2006].

7.4.1 *Anomaly Detection Experimental Methodology.*

Like the experiments above, we have used the publicly-available Movie-Lens 100k dataset[7] for our experiments. For all the attacks, we generated a number of attack profiles and inserted them into the system database and then evaluate each algorithm. We propose examining items against the distributions of items with similar characteristics which we term categories. The goal of this categorization is to make the distributions within each of the categories more similar within the underlying populations. The items that makeup each of these categories are then used to create the process control model for other items within the same category. We have categorized items in the following way.

First we defined two characteristics of movies, density (# of ratings) and average rating in the following way.

—low density (LD): # of ratings between 25 and 40
—medium density (MD): # of ratings between 80 and 120
—high density (HD): # of ratings between 200 and 300
—low average rating (LR): average rating less than 3.0
—high average rating (HR): average rating greater than 3.0

Then we partitioned our dataset into five different categories LDLR, LDHR, MDLR, MDHR, and HDHR. For example, category LDLR contains movies which are LD and LR. Table VI shows the statistics of the different categories computed from the MovieLens dataset. The category HDLR which is high density and low average rating has not been analyzed here due to insufficient examples in the Movie-Lens 100k dataset.

Evaluation Metrics. In order to validate our results we have considered two performance metrics, precision and recall. In addition to investigating the trade offs between these metrics, we seek to investigate how the size of attacks affects the performance. In our experiments,

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

Category	Average # of Ratings	Average Rating
HDHR	245.68	3.82
LDLR	31.26	2.61
LDHR	32.37	3.5
MDHR	97.5	3.54
MDLR	87.45	2.68

Table VI. Rating distribution for all categories of movies

precision and recall have been measured differently depending on what's being identified. The basic definition of recall and precision can be written as:

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

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

In statistical control limit algorithms, we are mainly interested in detecting movies which are under attack. So *# true positives* is the number of movies correctly identified as under attack, *# false positives* is the number of movies that were misclassified as under attack, and *# false negatives* is the number of movies which are under attack that are misclassified.

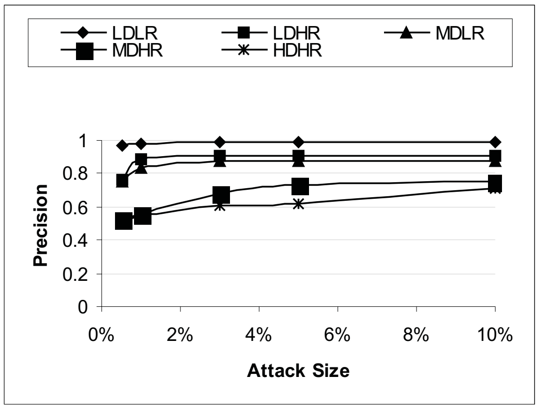
On the other hand, in the case of time interval detection, we are interested in detecting the time interval during which an attack occurred on an item. So *# true positives* is the number of time intervals correctly identified as being under attack, *# false positives* is the number of time intervals that were misclassified as attacks, and *# false negatives* is the number of time intervals that were misclassified as no attack.

Category	Training	Test
HDHR	50	30
LDLR	50	30
LDHR	50	50
MDHR	50	50
MDLR	30	14

Table VII. Total number of movies selected from each category in training and testing phases

In Section 5.2.2, there was a discussion as to why the metrics of sensitivity and specificity were more appropriate for our user classification results. As such, it seems appropriate to explain why those metrics were more appropriate for profile classification, but precision and recall are more appropriate for this context. The reason for this difference lies in the cost of misclassifications. In the context of profile classifications, the cost of misclassifying an authentic profile is its exclusion from contributing to predictions, thus potentially impacting the recommender's prediction performance. By contrast, in the context of detecting items under attack and time periods of attack, the cost associated with misclassifying authentic activity as attack activity is less severe since these classifications are indicators of suspicious activity rather than exclusion indicators. As such the metric of precision gives a better gauge of how informative is an indication of suspicion.

Methodology for detection via control limits. In SPC, the process parameters are estimated using historical data. This process is accomplished in two stages, training and testing. In the training stage, we use the historical ratings to estimate the upper and lower control limits. In the testing stage, we compare the new item's average rating with these limits. If the current average rating is outside of the boundaries we consider that an attack. Table VII shows the number of movies selected during training and testing phases. In the training phase, we used the ratings for all movies in the training set to compute the control limits.



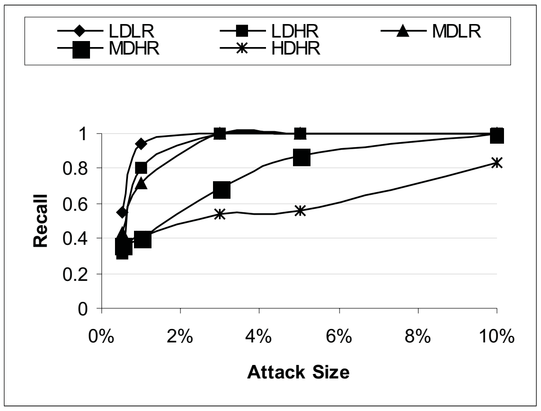


Fig. 26. The Precision and Recall graphs for all categories, varying push attack sizes using X-Bar control limits (sigma limit set to 3)

Our evaluation has been done in two phases. In the first phase, we calculated the average rating for each movie in the test set and checked whether it lies within the control limits. We assumed that the MovieLens dataset had no biased ratings for these movies and considered the base rating activity to have *no attack*. We then calculated the false positives, which are the number of *no attack* movies that were misclassified as under attack by our detection algorithm. In the second phase we generated attacks for all the movies in the test set. Two types of attacks were considered: push and nuke. For push attacks we gave a maximum possible rating 5 and for nuke attacks we gave a minimum possible rating 1. The average rating of each movie was then computed and checked whether it fell within the control limits. We then computed true positives, the number of movies correctly identified as under attack and false negatives, the number of movies which are under attack that were misclassified as *no attack* movies. Precision and recall have then been computed using the formula in the evaluation section.

Methodology for time interval detection. For the time interval detection algorithm, we relied upon the time-stamped ratings which were collected in the MovieLens dataset over a seven month period. Our main objective here is to detect the time interval over which an attack is made. The original dataset was sorted by the time-stamps given in the MovieLens dataset, and broken into 72 intervals, where each interval consists of 3 days.

For each category, we selected movies from the test set shown in Table VII. For each test movie, first we obtained ratings from sorted time-stamp data and computed mean and standard deviation prior to the t-th interval, which we assume contains no biased ratings. We set t to 20, which is equivalent to two months. Our assumption here is that the system owner has collected data from a trusted source prior to the t-th interval, which is considered as historical data without any biased ratings.

An attack (push or nuke) was then generated and inserted between the t-th and $(t + 20)$-th interval chosen at random times. We choose a long period of attack (20 intervals) so that an attacker can easily disguise himself as a genuine user and will not be easily detectable. During this time, we identified the time intervals as *attack* or *no attack* depending on whether our system generates an attack at that time interval or not. For each subsequent interval starting at t-th interval, we computed the average rating of the movie. If the average rating deviate significantly from the distribution of historical data, we considered this interval as a suspicious one. At this stage, we calculated the precision and recall for detecting attack intervals for each movie and averaged over all test movies.

7.4.2 *Anomaly Detection Results.* In our first set of experiments we built a predictive model for different categories of movies using SPC algorithms. Test items were then classified as either an item under attack or not under attack. Figure 26 shows the results for all categories

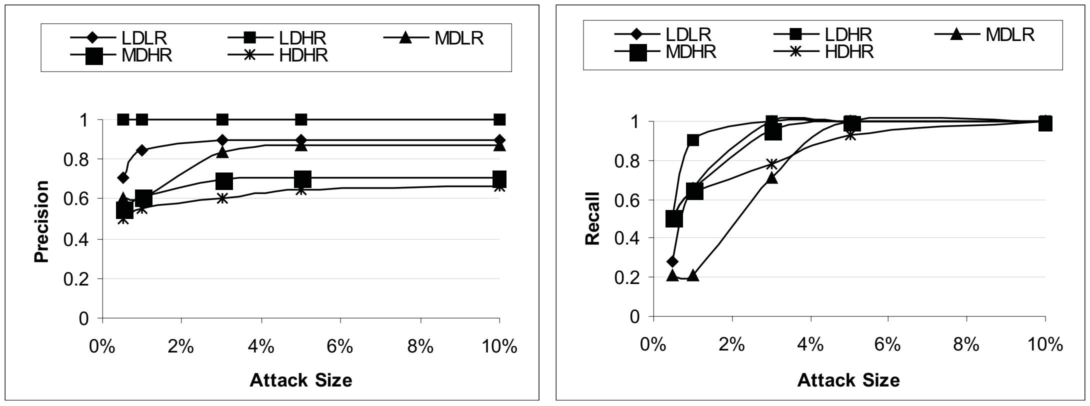


Fig. 27. The Precision and Recall graphs for all categories, varying nuke attack sizes using X-Bar control limits (sigma limit set to 3)

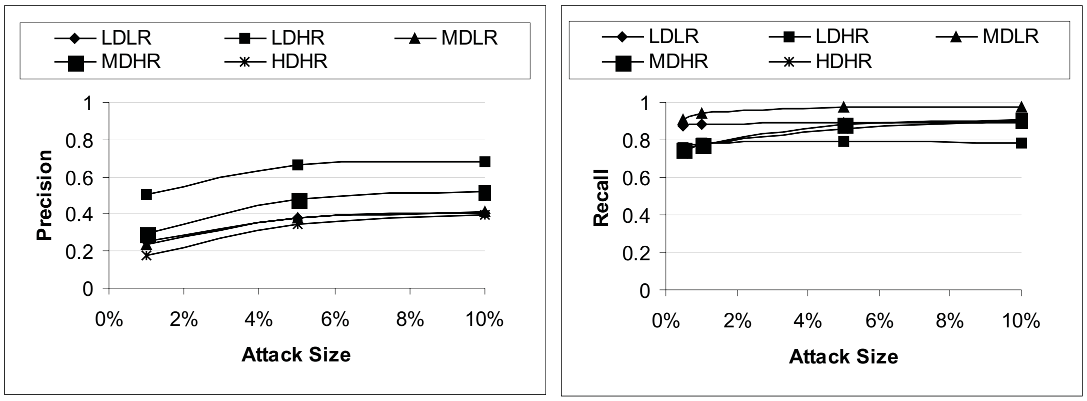


Fig. 28. The Precision and Recall graphs for all categories, varying push attack sizes using time series algorithm (α set to .05)

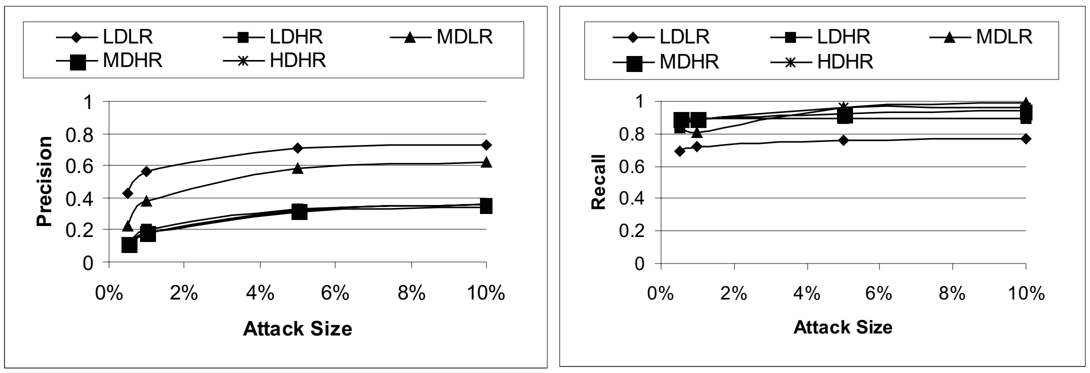


Fig. 29. The Precision and Recall graphs for all categories, varying nuke attack sizes using time series algorithm (α set to .05)

of movies at different attack sizes in a push attack, using X-Bar algorithm where sigma limit is set to 3. The recall chart shows that at lower attack sizes precision and recall are low for both the HDHR and MDHR categories. This observation is consistent with our conjecture that if an item is already highly rated then it is hard to distinguish from other items in this category after an attack. On the other hand the recall measures for LDLR, LDHR, and MDLR are 100% at 3% attack size. The lower the densities or average rating, the higher the recall values at different attack sizes. Similar results were obtained (not shown here) for the confidence interval control limit algorithm. Precision values in X-bar algorithm are much higher than Confidence Interval algorithm indicating X-bar algorithm works well in terms of classifying correctly no attack items, although both algorithms work for identifying suspicious items. As the figures depict, the performances also vary with different categories of items. This is consistent with our conjecture as the number of ratings (density) decrease, the algorithms detect the suspicious items more easily.

The next aspect we examined was the effectiveness of detection in the face of nuke attacks against all categories of movies. Figure 27 shows the results for all categories of movies varying attack sizes in a nuke attack using X-Bar algorithm where sigma limit is set to 3. The precision increases as the attack size increases, indicating this algorithm produces fewer false positives at higher attack sizes. The recall chart shows that the detection rate is very high even at 3% attack size for all categories. At lower attack sizes low density movies are more detectable than higher densities against nuke attack. It is reasonable to assume that the higher the densities, the lower the chance of decreasing average rating below the lower limit at lower attack sizes. The results depicted here confirm that this algorithm is also effective at detecting nuke attacks and the performance varies with different categories. The same trend has been obtained for the Confidence Interval algorithm not shown here.

The main objective of the time series algorithm is to detect a possible attack by identifying time intervals during which an item's ratings are significantly different from what is expected. In this experiment, first we obtained ratings from sorted time-stamp data for each item in the test dataset and computed mean and standard deviation prior to the t-th interval, which we assume contains no biased ratings and consider this as our historical data. An attack (push or nuke) was then generated and inserted between the t-th and $(t + 20)$-th interval chosen at random time. Now for each subsequent interval starting at t-th interval, we compute the average rating of the movie. If the average rating deviates significantly from the distribution of historical data, we flag this interval as a suspicious one.

The overall effect of this algorithm against all categories of movies are shown in Figure 28 against a push attack. The time interval detection rate for highly rated items is low at small attack sizes which indicate that it is very hard to detect the attack interval against a push attack. On the other hand, the results are opposite in nature against a nuke attack which is depicted in Figure 29. As expected, the highly rated items are easily detectable against a nuke attack.

The time interval results show that the period in which attacks occur can also be identified effectively. This approach offers some particularly valuable benefits by not only identifying the target of the attack and the type of attack (push/nuke), but also identifies the time interval over which the bias was injected. This combination of data would greatly improve a system's ability to triangulate on the most suspicious profiles. This technique could be combined with profile classification to further weight the suspicion of profiles that contributed to an item during a suspected attack interval. For very large datasets with far more users than items, profile analysis is likely to be resource intensive; thus it is easy to see the benefit of being able to narrow the focus of such tasks. One of the side effects of time based detection is forcing an attacker to spread out their attack over a longer period of time in order to avoid detection.

In the experiments above, we have shown that there are some significant differences in the detection performance over different groups of items based on their rating density and their average ratings. In particular, with the techniques described above, the items that seem most likely to be the target of a push attack (LDLR, LDHR, and MDLR) are effectively detected at even low attack sizes. For nuke attacks the detection of likely targets (LDHR and MDHR) is also fairly robust. Across the SPC detection schemes, detection was weakest for the HDHR group. However, as these items are the most densely rated, they are inherently the most robust to injected bias.

The performance evaluation indicated that the interval-based approach generally fairs well in comparison to SPC, even at lower attack sizes for detecting attacks. But precision was much lower in interval-based approach than in SPC. However, these two types of algorithms may be useful in different contexts. The interval-based method focuses on the trends rather than the absolute rating values during specific snapshots. Thus, it is useful for monitoring newly added items for which we expect high variance in the future rating distributions. On the other hand the SPC method works well for items that have well established distribution for which significant changes in a short time interval may be better indicators of a possible attack. As noted earlier, however, we do not foresee these algorithms to be used alone for attack detection. Rather, they should be used together with other approaches to detection, such as profile classification, in the context of comprehensive detection framework.

In addition to the direct benefits described above, all three of these techniques offer a crucial difference to profile classification alone; they are profile independent. Profile classification is fundamentally based on detecting profile traits researchers consider suspicious. While profiles that exhibit these traits might be the most damaging, there are likely ways to deviate from these patterns and still inject bias. Unlike traditional classification problems where patterns are observed and learned, in this context there is a competitive aspect since attackers are likely to actively look for ways to beat the classifier. Given this dynamic, detection schemes that combine multiple detection techniques that examine different aspects of the collaborative data are likely to offer significant advantages in robustness over schemes that rely on a single aspect. We envision combining the techniques outlined in this paper with other detection techniques to create a comprehensive detection framework.

8. CONCLUSIONS

In this section we highlight the discoveries made in this investigation and areas for future work.

8.1 Discussion

This paper has shown several key findings in the area of profile injection attack detection. In our analysis of the vulnerabilities of user-based recommendation to push and nuke attacks, we demonstrated that the characteristics that make an attack effective at introducing a positive bias are not necessarily the same as those required for creating a negative bias. While this is an interesting finding in itself, in the context of attack detection, this has the implication that a detection scheme could likely benefit from weighting detection features by the type of suspected attack.

In our analysis of the effects of the dimensions of attacks on the information gain of detection attributes, we showed that the relative information gain of attributes is far more complex than previous work has considered when evaluating the benefits of detection attributes. As a result, if a detection scheme had a priori knowledge of the dimensions of an attack, it could likely enhance its classification performance by adjusting the weighting of attributes to be optimal for the attack dimensions. While it is unlikely for a system to know the exact information about an attack, if a reasonable guess of these dimensions were incorporated in the attribute weighting this may yield significant improvements in classification performance.

As our results demonstrate, it is likely that all of these dimensions can be determined with reasonable accuracy. The easiest of these is the filler size, as it can be closely approximated by the profile size which is known. We also show the attack model being used can be fairly accurately identified for known attack models. While the benefit of such an approach is less clear for unknown or obfuscated attacks, it may be worth further investigation to see if such an attack model hint could improve classification. The last dimension, attack size, can be approximated through a combination of our intra-profile attributes and time series analysis.

In addition to these findings, we empirically evaluated the vulnerability of our supervised classification detection approach to models that deviate from known attacks. As we show, while techniques such as corate discounting can aid in reducing the impact of attacks at the most vulnerable filler sizes, additional approaches which are profile independent are likely needed. The use of approaches such as the SPC and time series techniques discussed will likely be needed to further increase robustness.

8.2 Future Work

The findings in this paper uncover a number of problems that still exist in the area of profile injection attack detection. For instance incorporating the impact of the attack dimensions on information gain to create a more intelligent weighting of detection attributes is likely to yield significant gains to the results presented here. In addition to fine tuning the profile detection component, a framework that combines the techniques discussed here into a more comprehensive detection scheme needs to be developed. Another area worth examining is whether applying detection attributes in an unsupervised fashion can improve robustness to unknown attacks. Genetic algorithms could offer another way of approaching the problem of protecting against unknown attack models. Theoretically a genetic algorithm could be used to probe the weaknesses of a detection scheme and establish an estimated upper bound on the impact of an attacker given some assumptions. In addition, these findings could be used to supplement the training set of the detection model, or used to adaptively fine tune the detection system. Finally, the vulnerabilities and detection schemes described in this work related to collaborative filtering with explicit numeric ratings, also likely exist in collaborative systems that use implicit feedback like web usage data, or free form textual feedback.

Users' trust in a collaborative recommender system will in general be affected by many factors, and the trustworthiness of a system, its ability to earn and deserve that trust, is likewise a multi-faceted problem. However, an important contributor to users' trust will be their perception that the recommender system really does what it claims to do, which is to represent even-handedly the tastes of a large cross-section of users, rather than to serve the ends of a few unscrupulous attackers. Progress in understanding these attacks and their effects on collaborative algorithms and advancements in the detection of attacks all constitute progress toward trustworthy recommender systems.

REFERENCES

Bhaumik, R., Williams, C., Mobasher, B., and Burke, R. 2006. Securing collaborative filtering against malicious attacks through anomaly detection. In *To appear in Proceedings of the National Conference on Artificial Intelligence (AAAI 2006): Workshop on Intelligent Techniques for Web Personalization*. Boston, Massachusetts.

Burke, R., Mobasher, B., and Bhaumik, R. 2005. Limited knowledge shilling attacks in collaborative filtering systems. In *Proceedings of the 3rd IJCAI Workshop in Intelligent Techniques for Personalization*. Edinburgh, Scotland.

Burke, R., Mobasher, B., Williams, C., and Bhaumik, R. 2005a. Collaborative recommendation vulnerability to focused bias injection attacks. In *International Conference on Data Mining: Workshop on Privacy and Security Aspects of Data Mining (ICDM 2005)*. Houston, Texas.

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

Burke, R., Mobasher, B., Williams, C., and Bhaumik, R. 2006a. 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.

Burke, R., Mobasher, B., Williams, C., and Bhaumik, R. 2006b. 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.

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

Chirita, P.-A., Nejdl, W., and Zamfir, C. 2005. Preventing shilling attacks in online recommender systems. In *WIDM '05: Proceedings of the 7th annual ACM international workshop on Web information and data management*. ACM Press, New York, NY, USA, 67–74.

Herlocker, J., Konstan, J., Borchers, A., and Riedl, J. 1999. 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.

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

Hunt, E. B., Marin, J., and Stone, P. T. 1966. *Experiments in Induction*. Academic Press, New York, NY, USA.

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

Massa, P. and Avesani, P. 2004. Trust-aware collaborative filtering for recommender systems. *Lecture Notes in Computer Science 3290*, 492–508.

Mobasher, B., Burke, R., Bhaumik, R., and Williams, C. 2005. 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.

Mobasher, B., Burke, R., and Sandvig, J. 2006. Model-based collaborative filtering as a defense against profile injection attacks. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI'06)*. Boston, Massachusetts.

Mobasher, B., Burke, R., Williams, C., and Bhaumik, R. 2006. 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.

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

O'Mahony, M., Hurley, N., and Silvestre, G. 2004. An evaluation of neighbourhood formation on the performance of collaborative filtering. *AI Review, Kluwer Academic Publishers*.

Ott, R. L. 1992. *An Introduction to Statistical Methods and Data Analysis*. Duxbury.

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

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. 1994. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*. ACM Press, 175–186.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International World Wide Web Conference*. Hong Kong.

Shewart, W. A. 1931. *Economic Control of Quality of manufactured Product*. Van Nostrand.

Williams, C., Mobasher, B., Burke, R., Sandvig, J., and Bhaumik, R. 2006. Detection of obfuscated attacks in collaborative recommender systems. In *Proceedings of the Workshop on Recommender Systems at the 17th European Conference on Artificial Intelligence (ECAI 2006)*.

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

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