Towards Trustworthy Recommender Systems: An Analysis of Attack Models and Algorithm Robustness

BAMSHAD MOBASHER, ROBIN BURKE, RUNA BHAUMIK, and CHAD WILLIAMS
Center for Web Intelligence
School of Computer Science, Telecommunication and Information Systems
DePaul University, Chicago, Illinois
(mobasher,rburke,rbhaumik,cwilli43)@cs.depaul.edu

Publicly-accessible adaptive systems such as collaborative recommender systems present a security problem. Attackers, who cannot be readily distinguished from ordinary users, may inject biased profiles in an attempt to force a system to "adapt" in a manner advantageous to them. Such attacks may lead to a degradation of user trust in the objectivity and accuracy of the system. Recent research has begun to examine the vulnerabilities and robustness of different collaborative recommendation techniques in the face of "profile injection" attacks. In this paper, we outline some of the major issues in building secure recommender systems, concentrating in particular on the modeling of attacks and their impact on various recommendation algorithms. We introduce several new attack models and perform extensive simulation-based evaluation to show which attack models are most successful against common recommendation techniques. We consider both the overall impact on the ability of the system to make accurate predictions, as well as the degree of knowledge about the system required by the attacker to mount a realistic attack. Our study shows that both user-based and item-based algorithms are highly vulnerable to specific attack models, but that hybrid algorithms may provide a higher degree of robustness. Finally, we develop a novel classification-based framework for detecting attack profiles and show that it can be effective in neutralizing some attack types.

Categories and Subject Descriptors: H3.3 [**Information Storage and Retrieval**]: Information Filtering

Additional Key Words and Phrases: Profile Injection Attacks, Collaborative Filtering, Recommender Systems, Shilling, Attack Detection

1. INTRODUCTION

Collaborative filtering recommender systems are based on the types of recommendation behavior that occurs in our everyday social interactions: people share their opinions about their likes and we decide whether or not to act on them [Herlocker et al. 2006]. Collaborative filtering (CF) has the advantage that such interactions can be scaled to groups of thousands or even millions, far more than could possibly meaningfully share opinions in virtually any other way. However, everyday social recommendation has an advantage that collaborative systems lack, which is the giver of recommendations has a known stable

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

identity on which receivers of recommendations can rely. Over time, you may come to discount the recommendations of a friend whose tastes have been shown to be incompatible. Anonymous or pseudonymous users of an on-line system, on the other hand, can multiply their profiles and identities nearly indefinitely.

It is quite clear that an adaptive system that depends on profiles built by anonymous unauthenticated users, like most Web-based recommender systems, is subject to manipulation. In the most extreme case, we might imagine a system that has nothing in it except profiles injected by an attacker. Obviously, the attacker can make the system produce any recommendation behavior he or she desires. Indeed, recent work has shown that surprisingly modest attacks are sufficient to manipulate the behavior of the most-commonly used recommendation algorithms [O'Mahony et al. 2004; Lam and Riedl 2004; Burke et al. 2005; Mobasher et al. 2005].

The theoretical basis for the vulnerabilities in collaborative recommendation has been well-established. Much of this work relates to earlier research on the impact of biased noise on classification accuracy. In particular, the formal framework introduced in [O'Mahony et al. 2004] extends the noise-free Probably Approximately Correct (PAC) [Haussler 1990] model of [Albert and Aha 1991] for k-nearest neighbor classification to handle biased class noise.

The vulnerabilities of collaborative recommender systems to attacks have led to a number of recent studies focusing on the notion of "trust" in recommendation from different perspectives. One such perspective involves the explicit calculation and integration of trust and reputation in the recommendation framework, as exemplified by recent work on "Trust-aware" collaborative filtering [Massa and Avesani 2006]. In the latter study, the authors consider a framework which allows for the elicitation of trust values among users. The filtering process is informed by the reputation of users computed by propagating trust values.

Another perspective is focused mainly on the notion of trust from a more global perspective, i.e., the trust users can place in the accuracy of recommendations produced by the system. From this point of view, the vulnerabilities of collaborative recommender systems are studied in terms of the robustness of such systems in the face of malicious attacks. This is also the perspective used in the present work.

Previous work on the robustness of recommender systems has also examined a number of attack models that are simple to formulate, but perhaps impractical from an attacker's perspective. For example, O'Mahony and colleagues use an attack that draws attack profiles directly from the rating database [O'Mahony et al. 2004]. This definition of an attack makes possible their formal analysis of the problem, allowing them to assume that the attributes of the attack profiles are noise-free. Lam and Riedl use attacks that assume the attacker has quite extensive knowledge of the distribution of ratings (average and deviation) across all items [Lam and Riedl 2004]. O'Donovan and Smyth [O'Donovan and Smyth 2006] show that trust-based collaborative filtering algorithms can be even more susceptible to profile injection attacks since attacking profiles can actually reinforce mutual opinions during the trust building process. This underlines the importance of studying the properties of typical attack models which, in turn, can lead to better automated attack detection algorithms, as well as to more robust recommendation algorithms.

Our work examines the problem of profile injection from a practical standpoint. In particular, we are interested in attacks that can be mounted with minimum knowledge of the ratings distribution. We are interested in whether different recommendation algorithms offer differing degrees of robustness against attack, whether it is possible for an attacker to craft attacks that are tailored to exploit the weaknesses of each algorithm, whether there are attacks effective against all common algorithms, and whether attacks can be detected and rendered harmless.

We know that in the theoretical limit any attack that injects biased profiles will be effective against any algorithm if it is of sufficiently large size (that is, contains enough profiles and ratings). Therefore, the question of robustness cannot be decoupled from the question of detection. All algorithms are vulnerable at some level of attack strength, but very large attacks will by necessity have distinct and recognizable signatures. Therefore, a robust algorithm will be one that, not only minimizes the impact of attacks on system behavior, but also requires such a large attack that the attacker's aims become obvious. We believe that sound detection will be based on an understanding of the vulnerabilities of the algorithms and the modeling of attacks that will be most effective against them. If we can model and recognize the most effective attacks, attackers will be forced to use methods that by definition have less impact and therefore require larger and more recognizable attack signatures. Furthermore, modeling of the most effective attack types allows us to derive the characteristic features of attack profiles which can, in turn, be used for detecting and neutralizing such profiles.

One important consideration is the recommendation algorithm itself. User-based collaborative filtering [Herlocker et al. 1999] is the classic formulation of the collaborative recommendation model, where the most proximal neighbors to a target user are selected and their profiles are used as a basis for predicting ratings for items as yet unrated by that target user. There are numerous other formulations of the collaborative technique, including model-based techniques in which a model is learned from the user profiles and then applied to a new user's profile. A model-based variation of standard user-based model is called *item-based collaborative filtering* [Sarwar et al. 2001], which works by comparing item similarities based on their pattern of ratings across users. Bayesian networks, association rules, and latent semantic analysis are just a few of the other techniques that have been applied to the problem of collaborative recommendation [Breese et al. 1998; Mobasher et al. 2001; Billsus and Pazzani 2000]. In this paper, we focus specifically on the standard user-based and item-based collaborative filtering algorithms and their vulnerabilities to different types of attacks. (See [Mobasher et al. 2006] for a study that examines attacks against several model-based recommendation algorithms.)

This paper begins with an examination of algorithms for collaborative recommendation and of models of attacks against them, both those used in prior literature and some that we have developed. A general formal framework is established in Section 3 and the special cases of each attack type are defined. We also discuss the question of evaluation: how to define the property of robustness that we are interested in measuring with respect to the different algorithms and attack models.

In Section 5 we present detailed experimental results comparing the effectiveness of different attack models across a variety of recommendation algorithms. We first show the impact of various push attack models (i.e., attacks designed to increase the probability that an item is recommended) on the user-based algorithm, then examine the more robust item-based algorithm. The effectiveness of these attack models for push attacks are then compared to their effectiveness for nuke attacks (attacks that are designed to reduce the probability of a target item being recommended).

Our study will show that both user-based and item-based algorithms are susceptible to low knowledge attacks, i.e., attacks that require minimal knowledge of the system and user profiles. These algorithms' vulnerabilities to push attacks have been the focus of previous work. We also explore the robustness of these algorithms in the face of nuke attacks. This examination uncovers some surprising differences in each algorithm's response to different attack models depending on the type of attack. We also introduce two effective reduced knowledge attacks for nuking items.

Another class of recommender system uses algorithms based not on user ratings alone but also information about the items themselves. A content-based recommender, for example, induces a classifier based on a particular user's ratings, and then produces recommendations on that basis [Lang 1995; Mooney and Roy 1999]. A knowledge-based recommender reasons about the fit between a user's need and the features of available products [Burke 2000]. An effective response to the problem of biased ratings may be to combine the use of content with the use of collaborative information. Hybrid recommendation, combining multiple recommenders of different types, is therefore a promising approach for securing recommender systems.

In Section 6, we provide a detailed analysis of a hybrid recommendation algorithm which extends the standard item-based collaborative filtering by integrating semantic knowledge about items into the computation of item similarities. Our empirical analysis of the semantically enhanced hybrid algorithm shows that it can be effective at reducing the impact of profile injection attacks, and thus has promise as a potential solution to this problem.

Finally, in Section 7, we present a set of generic, as well as model-specific features, based on the statistical characteristics of attack profiles, which can be used for detecting and neutralizing attacks. The results in our study of attack detection show that simple classification learning, based on these features, is a promising approach for defending against profile injection attacks.

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 Riedl 2004; O'Mahony et al. 2004]. We prefer the phrase *profile injection attacks*, since promoting a particular product is only one way such attack 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.

Our overall aim is to identify different types of profile injection attacks, to study their characteristics and their impact on common collaborative filtering recommendation algorithms, and to develop techniques for defending recommender systems against them. 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 attack models and algorithms 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. There are various ways that the effort required to mount an attack can be evaluated, but in this paper, we will emphasize the issue of knowledge: what does the attacker have to know in order to launch a particular attack? Knowledge that is specific to particular system, such as the algorithm that it uses and/or the details of the ratings distribution within, can be considered more difficult to obtain than general knowledge about products, for example what books are best-sellers. System owners can take steps to avoid disclosure of the first type of information but not the second. We use a relatively informal distinction between two types of attack based on knowledge:

High-knowledge attack:. A high-knowledge attack is one that requires very detailed knowledge the ratings distribution in a recommender system's database. Some attacks, for example, require that the attacker know the mean rating and standard deviation for every item. These would be classified as high-knowledge.

Low-knowledge attack:. A low-knowledge attack is one that one requires system-independent knowledge such as might be obtained by consulting public information sources.

A second dimension of an attack is the intent of an 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.) We are concerned primarily with the "win" for the attacker: the change in the predicted rating of the attacked item. Our metrics for measuring the impact of attacks are described in detail in Section 4.

The size of an attack can be measured in several ways. We look at both the number of profiles being added by the attacker and the number of ratings that are supplied in each profile. 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. The addition of ratings is relatively lower in cost. However, there is the additional factor of risk at work when profiles include ratings for a large percentage of the rateable items. Real human users never 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.

2.2 Types of Attacks

An attack against a collaborative filtering recommender system consists of a set of attack profiles, each contained biased rating data associated with a fictitious user identity, and including 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.

	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
Correlation with Item6	0.85	-0.55	0.00	0.48	-0.59		

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

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 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.

On the other hand, if a system is using an item-based collaborative filtering approach, then the predicted rating for Item6 will be determined by comparing the rating vector for Item6 with those of the other items. This algorithm does not lend itself to an attack as obvious as the previous one, since Eve does not have control over ratings given by other users to any given item. However, if Eve can obtain some knowledge about the rating distributions for some items, this can make a successful attack more likely. In the example of Figure 1, for instance, Eve knows that Item1 is a popular item among a significant group of users to which Alice also belongs. By designing the attack profiles so that high ratings are associated with both Item1 and Item6, Eve can attempt to increase the similarity of

	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
Correlation with Item6	-0.64	-0.41	0.34	0.13	-0.28		

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

these two items, resulting in a higher likelihood that Alice (and the rest of the targeted group) will receive Item6 as a recommendation. Indeed, as the example portrays, such an attack is highly successful regardless of whether the system is using an item-based or a user-based algorithm. This latter observation illustrates the motivation behind one of the new attack models we introduce and analyze in this paper, namely the *segment attack*.

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.

Interestingly, this attack is not effective against the item-based algorithm. The prediction is the same (3) before and after the attack. Previous studies [Lam and Riedl 2004] have suggested that item-based collaborative algorithms are generally more robust against profile injection attacks than their user-based counter-parts. In this paper, we show that more sophisticated attack models can, in fact, have a dramatic impact on item-based algorithms as well.

3. ATTACK MODELS

In this section, we begin by presenting a general formal framework for specifying attack models and attack profiles. We then turn our attention to several specific attack models that we have introduced or studied in this work. In each case, we use our formal framework to define the attack model and briefly discuss its properties and its characteristics. Later, in Section 5, we present our detailed experimental results corresponding to these attack models.

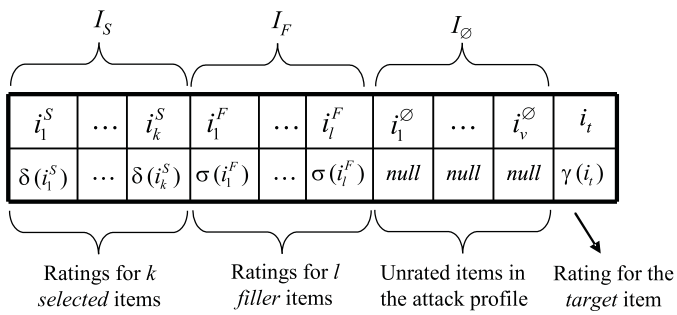

Fig. 3. The general form of a user profile from a profile injection attack based on Definitions 1 and 2.

3.1 Profile Injection Attacks: A Formal Framework

Let I be a set of items, U a set of users, R a set of rating values, and $UP = \{up_1, up_2, \cdots, up_d\}$ a set of user profiles, where each up_i is a set of pairs $\langle i, r\rangle$, where $i \in I$ and $r \in R \cup \{\texttt{null}\}$, with `null` representing a missing or undefined rating.

A *recommender system* can be viewed as a mapping $S : \mathbf{2}^{UP} \times U \times I \rightarrow R \cup \{\texttt{null}\}$, assigning rating values to pairs of users and items. More specifically, in the usual context of collaborative recommendations, given a *target item*, $i_t \in I$, whose rating will be extrapolated for a *target user*, $u_t \in U$, and a set of user profiles $P \in UP$, $S(P, u_t, i_t)$ "predicts" a rating value for u_t on item i_t.

A profile-injection attack against a recommender system consists of a set of profiles added to the system by the attacker. The generic form of these profiles is shown in Figure 3. We can think of each profile as identifying four sets of items: a singleton target item i_t, a set of selected items with particular characteristics determined by the attacker I_S, a set of filler items usually chosen randomly I_F, and a set of unrated items $I_\emptyset$. Attack models can be defined by the methods by which they identify the *selected items*, the proportion of the remaining items that are used as *filler items*, and the way that specific ratings are assigned to each of these sets of items and to the target item. The set of selected items represents a small group of items that have been selected because of their association with the target item (or a targeted segment of users). For some attacks, this set is empty. On the other hand, the set of filler items represent a group of randomly selected items in the database which are assigned ratings within the attack profile. Since the selected item set is small, the size of each profile (total number of ratings) is determined mostly by the size of the filler item set. In our experimental results, we report filler size as a proportion of the size of I (i.e., the set of all items).

DEFINITION 1. An *attack model* is a 4-tuple $\mathcal{M} = \langle \chi, \delta, \sigma, \gamma\rangle$, where:

—$\chi(i_t$,I,U, $\Phi) = \langle I_S, I_F, I_\emptyset\rangle$ is choice function which given a target item i_t, the set of all items I, the set of all users U, and a set of parameters Φ, partitions the set I, such that I_S is a set of *selected items* determined based on pre-specified parameters in Φ; I_F is a set of randomly selected *filler items*, based on a pre-specified random variable in Φ; and $I_\emptyset = I - (I_S \cup I_F \cup \{i_t\})$ is the set of *unrated items*;

—$\delta : I \rightarrow R$ and $\sigma : I \rightarrow R$ are mappings of the elements of I to rating values, used respectively to give ratings to the sets I_S and I_F; and

—$\gamma : \{i_t\} \rightarrow R$ is mapping of I to a (pre-specified) target rating value, used for the target item i_t.

The set of parameters Φ used in the choice function χ are specific to the particular attack model. The set of selected items, I_S, specified by χ, may be determined according to a number of factors which we have generically combined into the parameter set Φ for the sake of presentation simplicity. These factors may include the distribution of rating values among items or users, the likelihood that a particular item is highly or frequently rated, or the expected characteristics associated with a particular segment of users. The specific parameters used for each specific attack model will be presented below.

DEFINITION 2. An *attack profile based on an attack model* $\mathcal{M}$, is a set of item-rating pairs $ap(\mathcal{M}) = P_S \cup P_F \cup P_t \cup P_\emptyset$, where:

—$\mathcal{M} = \langle \chi, \delta, \sigma, \gamma \rangle$ is an attack model;

—$P_S = \{\langle i, r\rangle \mid i \in I_S, r \in R, \delta(i) = r\}$;

—$P_F = \{\langle i, r\rangle \mid i \in I_F, r \in R, \sigma(i) = r\}$;

—$P_t = \{\langle i_t, r_t\rangle\}$, where $r_t \in R$ and $\gamma(i) = r_t$;

—$P_\emptyset = \{\langle i, r\rangle \mid i \in I_\emptyset, r = \texttt{null}\}$.

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 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). The basic elements of a profile injection attack are expressed more formally in the following definition.

DEFINITION 3. A *profile injection attack of size* n (an attack of size n, for short) against a recommender system S consists of a set $AP^n_{\mathcal{M}} = \{ap_1(\mathcal{M}), \cdots, ap_n(\mathcal{M})\}$ of attack profiles based on an attack model $\mathcal{M}$, added to the database of user profiles UP. A *push attack*, is an attack, $AP^n_{\mathcal{M}}$ such that for a given target user u_t and a target item i_t, $S(UP \cup AP^n_{\mathcal{M}}, u_t, i_t) > S(UP, u_t, i_t)$. On the other hand, a *nuke attack*, $AP^n_{\mathcal{M}}$, is such that $S(UP \cup AP^n_{\mathcal{M}}, u_t, i_t) < S(UP, u_t, i_t)$.

We next focus our attention on a number of specific attack models, several of which have been identified for the first time in this work, and discuss some of their characteristics.

3.2 Push Attack Models

Two basic attack models, introduced originally in Lam and Riedl [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.

3.2.1 *Random Attack.* As noted above, the random attack profiles 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 is defined as follows.

DEFINITION 4. The *random attack model* is an attack model,

$$\mathcal{M}_{rand} = \langle \chi_{rand}, \delta_{rand}, \sigma_{rand}, \gamma_{rand} \rangle,$$

with 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, $f = |I_F|/|I - \{i_t\}|$ is a pre-determined parameter specified in χ_{rand};

—$\forall i \in I_F, \sigma(i) \sim \mathcal{N}(\bar{r}_I, s_I)$, where $\bar{r}_I$ and s_I are the mean and standard deviation of ratings for all items in I, i.e., 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;

—$\gamma(i_t) = r_{max}$.

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). The execution cost involved, however, can be substantial, since this attack usually involves assigning ratings to every item in each attack profile. Furthermore, as Lam and Riedl [2004] shows and our results confirm [Burke et al. 2005], the attack is not particularly effective.

3.2.2 *Average attack.* A more powerful attack described in Lam and Riedl [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. Formally, the average attack model can be described as follows.

DEFINITION 5. The *average attack model* is an attack model,

$$\mathcal{M}_{avg} = \langle \chi_{avg}, \delta_{avg}, \sigma_{avg}, \gamma_{avg} \rangle,$$

with 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, $f = |I_F|/|I - \{i_t\}|$ is a pre-determined parameter specified in χ_{avg};

—$\forall i \in I_F, \sigma(i) \sim \mathcal{N}(\bar{r}_i, s_i)$, where $\bar{r}_i$ and s_i are the mean and standard deviation of ratings for item i across all users, i.e., the rating value for each item $i \in I_F$ is drawn from a normal distribution around the mean rating for i;

—$\gamma(i_t) = r_{max}$

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. Figure 4 depicts the general form for both the random and the average attacks.

In addition to the effort involved in producing the ratings, the average attack also has considerable knowledge cost of order $|I_F|$ (the number of filler items in the attack profile). Our experiments, however, have shown that, in the case of user-based collaborative filtering algorithms, the average attack can be just as successful even when using a small filler item set. i.e., by assigning the average ratings to only a small subset of items in the database.

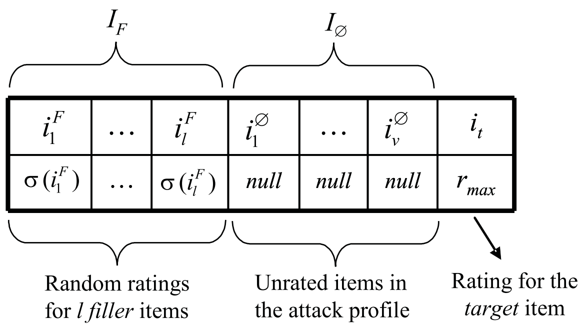

Fig. 4. An attack profile based on the random or the average attack model. See Definitions 4 and 5.

Thus, the knowledge requirements for this attack can be substantially reduced [Burke et al. 2005]. This attack model, however, is not, as effective against an item-based collaborative algorithm, as we will show in Section 5 below.

The characteristics of these standard attack models raise several immediate questions. If the average attack is impractically knowledge-intensive, then perhaps it is not as much of a threat as we might image. Can variants of this attack be found that require less knowledge on the part of the attacker? If the item-based algorithm is relatively unaffected by the average attack, is switching to this algorithm a simple and effective defense against profile injection attacks? To answer these questions, we experimented with some additional attack models: the bandwagon and segment attacks.

3.2.3 *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.

DEFINITION 6. The *bandwagon attack model* is an attack model,

$$\mathcal{M}_{bw} = \langle \chi_{bw}, \delta_{bw}, \sigma_{bw}, \gamma_{bw} \rangle,$$

with the following characteristics:

—I_S is a set of items that are likely to be densely-rated, as determined by χ_{bw}. I.e., I_S is chosen to maximize the likelihood that for each $i \in I_S$, $|\{\langle i, r\rangle \in up_j \mid up_j \in UP, r \neq \texttt{null}\}|$ will be high.

—$\forall i \in I_S, \delta(i) = r_{max}$, where r_{max} is the maximum rating value in R.

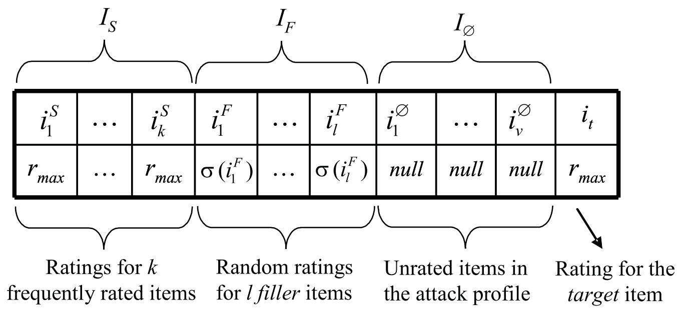

Fig. 5. A *Bandwagon* attack profile.

—I_F is a set of randomly chosen filler items drawn from $I - (\{i_t\} \cup I_S)$, where the ratio of filler items, $f = |I_F|/|I - \{i_t\}|$ is a pre-determined parameter specified in Φ_{bw};

—$\forall i \in I_F, \sigma(i) \sim \mathcal{N}(\bar{r}_I, s_I)$, where $\bar{r}_I$ and s_I are the mean and standard deviation of ratings for all items in I, i.e., 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;

—$\gamma(i_t) = r_{max}$

Figure 5 depicts a typical attack profile for the bandwagon attack. 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. Note that the items in I_S are given positive ratings. Examination of ratings data in the movie domain showed that densely-rated items are also generally highly-rated. There were no "negative blockbusters," movies that many people disliked; negative ratings tended to be more highly dispersed, and in general, there are fewer negative than positive ratings. This could be because movie-goers will tend to have a selection bias towards movies that they will like, or it could be that they will tend not to rate disliked movies as often.

We showed in [Burke et al. 2005] that the bandwagon attack can still be successful even when only a small set of the filler items are assigned ratings. As we show in Section 5, 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. However, as in the case of the average attack, it falls short when used against an item-based algorithm.

3.2.4 *Segment Attack.* Previous work [Lam and Riedl 2004] concluded that item-based algorithms were more robust than user-based ones and the average attack has been found to be most effective. From a cost-benefit point of view, however, such attacks 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 introduce the *segment attack* model as a reduced knowledge push attack specifically designed for the item-based algorithms [Mobasher et al. 2005].

It is a basic truism of marketing that the best way to increase the impact of a promotional activity 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.

We can frame this intuition as a question of utility. We assume that the attacker has a particular item i that she wants recommended more highly because she has a personal stake in the success of this product. The attacker receives some positive utility or profit p_i each time i is purchased. Let us denote the event that a recommendation of product i is made to a user u, by $R_{u,i}$ and the event that a user buys an item by $B_{u,i}$. The probability that a user will purchase i if it is recommended we can describe as a conditional probability: $P(B_{u,i}|R_{u,i})$. Over all users U that visit the system over some time period, the expected profit would be

$$P = \sum_{u \in U} p_i * P(R_{u,i}) * P(B_{u,i}|R_{u,i})$$

The attacker of a recommender system hopes to increase her profit by increasing $P(R_{u,i})$, the probability that the system will recommend the item to a given user.

However, preferences for most consumer items are not uniformly distributed over the population of buyers. For many products, there will be users (like a "Harry Potter" buyers) who would be susceptible to following a recommendation for a related item (another fantasy book for children) and others who would not. In other words, there will be some segment of users S that are distinguished from the rest of the user population $N = U - S$, by being likely recommendation followers:

$$\forall s \in S, \forall n \in N, P(B_{s,i}|R_{s,i}) \gg P(B_{n,i}|R_{n,i}))$$

Let us consider an extreme case of a niche market in which $P(B_{n,i}|R_{n,i})$ is zero. The only customers worth recommending to are those in the segment S. Everyone else will ignore the recommendation. It is in the attacker's interest to make sure that the attacker item is recommended to the segment users; it does not matter what happens to the rest of the population. The attacker will be only interested in manipulating the quantity $P(R_{s,i})$. In other words, the quantity that matters to an attacker may not be the overall impact of an attack, but rather its impact on a segment of the market distinguished as likely buyers. This may even be true if $P(B_{n,i}|R_{n,i}) > 0$ because these out-of-segment buyers contribute relatively little to the expected utility compared to the in-segment ones.

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

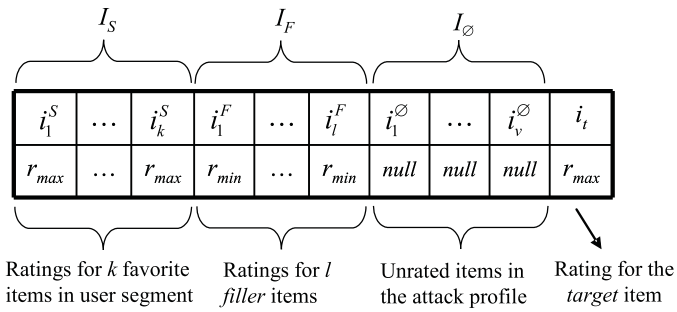

Fig. 6. A *Segment* attack profile.

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.

The segment attack model is formally defined as follows.

DEFINITION 7. [Segment Attack] The *segment attack model* is an attack model, $\mathcal{M}_{seg} = \langle \chi_{seg}, \delta_{seg}, \sigma_{seg}, \gamma_{seg} \rangle$, with the following characteristics:

—I_S is a set of selected items specified in χ_{seg} which the attacker has chosen to define the segment.
—$\forall i \in I_S, \delta(i) = r_{max}$, where r_{max} is the maximum rating value in R.
—I_F is a set of randomly chosen filler items as in Definition 6.
—$\forall i \in I_F, \sigma(i) = r_{min}$, where r_{min} is the minimum rating value in R;
—$\gamma(i_t) = r_{max}$, is the rating for the target item.

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.

Figure 6 depicts a typical attack profile based on the segment attack model. The selected segment items, i_1^S through i_k^S in I_S represent the items that are (likely to be) favored by the targeted segment of users. These items are assigned the maximum rating value together with the target item. To provide the maximum impact on the item-based CF algorithm, the minimum rating was given to the filler items, thus maximizing the variations of item similarities used in the item-based algorithm.

The detailed experimental results for this attack model are presented in Section 5. The results show that this attack model is quite effective against both item-based and user-based collaborative filtering.

3.3 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 5, suggest that attack models that are effective for pushing items are not necessarily effective for nuke attacks.

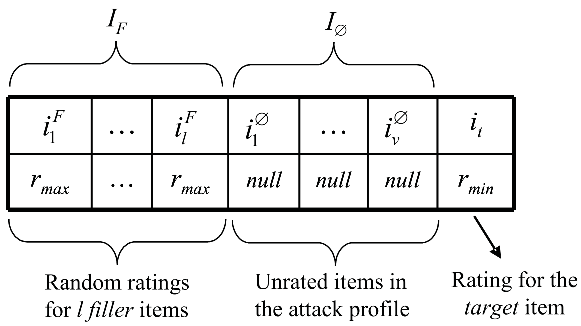

Fig. 7. An attack profile, based on the *Love/Hate* attack model.

We have identified two additional attack models designed particularly for nuking items: "love/hate" and "reverse bandwagon". They have proved to be particularly effective against the user-based and item-based algorithms, respectively. Both of them can be considered to have reduced knowledge cost, as they do not require any system-specific data. However reverse bandwagon does require some general knowledge of product domain to be able to effectively select low rated items that will have a significant number of ratings. These attack models are described in more detail below.

3.3.1 *Love/Hate attack.* The *love/hate attack* 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}.

The formal definition for this attack model is given below.

DEFINITION 8. [Love/Hate Attack] The *love/hate attack model* is an attack model,

$$\mathcal{M}_{lh} = \langle \chi_{lh}, \delta_{lh}, \sigma_{lh}, \gamma_{lh} \rangle,$$

with the following characteristics:

—$I_S = \emptyset$;
—I_F is a set of randomly chosen filler items as in Definition 6.
—$\forall i \in I_F, \sigma(i) = r_{max}$, where r_{max} is the maximum rating value in R;
—$\gamma(i_t) = r_{min}$, where r_{min} is the minimum rating value in R.

Figure 7 depicts a typical attack profile based on the love/hate attack model. Clearly, the knowledge required to mount such an attack is quite minimal. Furthermore, as our results will show, although this attack is not effective at producing recommender bias when used as a push attack, it is one of the most effective as a nuke attack against the user-based CF algorithm.

3.3.2 *Reverse Bandwagon attack.* The *reverse bandwagon* attack is a variation of the bandwagon attack, discussed above, in which the selected items are those that tend to be rated poorly by many users. These items are assigned low ratings together with the target item. Thus, the target item is associated with widely disliked items, increasing the probability that the system would generate low predicted ratings for that item.

The bandwagon attack takes advantage of the fact that high rated items also tend to be very popular. Low rated items, on the other hand, tend to have sparser ratings, making it more challenging to select items to be included in the attack that also have enough ratings to make a significant impact. The item with the lowest average rating in the system might be rated by only a few users. To build an attack model that has a significant impact a large number of items would need to be known to have poor ratings, thus increasing the knowledge required for the attack.

The reverse bandwagon was designed to reduce the knowledge required by selecting only a handful of known unpopular items. An attacker using this model, would select items that are widely known for having poor ratings. For example, in the movie domain, these may be box office flops that had been highly promoted prior to their openings.

This attack model is defined formally as follows.

DEFINITION 9. [Reverse Bandwagon Attack] The *reverse bandwagon attack model* is an attack model,

$$\mathcal{M}_{rbw} = \langle \chi_{rbw}, \delta_{rbw}, \sigma_{rbw}, \gamma_{rbw} \rangle,$$

with the following characteristics:

—I_S is a set of selected densely-rated items as in Definition 6 with the additional requirement that they have below average ratings, that is, $\forall i \in I_S, \bar{r}_i \leq \bar{r}_I$, where $\bar{r}_i$ is the mean of rating for item i across all users and $\bar{r}_I$ is the overall mean rating in the whole database.

—$\forall i \in I_S, \delta(i) = r_{min}$, where r_{min} is the minimum rating value in R.

—I_F is a set of randomly chosen filler items as in Definition 6

—$\forall i \in I_F, \sigma(i) \sim \mathcal{N}(\bar{r}_I, s_I)$, where $\bar{r}_I$ and s_I are the mean and standard deviation of ratings for all items in I, i.e., 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;

—$\gamma(i_t) = r_{min}$, is the rating for the target item.

Figure 5 can also be used to describe the reverse bandwagon attack, as defined here. However, in the case of this attack model, the roles of r_{min} and r_{max} are switched in Figure 5. In Section 5, we show that although this attack is not as effective as the more knowledge intensive average attack for nuking items in the user-based system, it is a very effective nuke attack against item-based recommender systems.

3.4 Summary of Attack Models

Table I summarizes the attack models described so far based on the characteristics of the attack profile partitions identified in Definition 1 and whether they are used for pushing or nuking items.

4. RECOMMENDATION ALGORITHMS AND EVALUATION METRICS

In this paper we focus on the most commonly-used algorithms for collaborative filtering, namely user-based and item-based. Previous work had suggested that item-based collaborative filtering might provide significant robustness compared to the user-based algorithm, but, as this paper shows, the item-based algorithm also is still vulnerable in the face of some of the attacks we introduced in the previous section. We believe that hybrid recommender systems that rely on a combination of user profiles and semantic knowledge about

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}
Reverse Bandwagon	nuke	Widely disliked items assigned rating r_{max}	Ratings assigned with normal distribution around system mean	Determined by filler size	r_{min}

Table I. Attack model summary

the domain may provide a higher degree of robustness against profile injection attacks, and hence a potential solution to the problem addressed by this work. We, therefore, also introduce a hybrid algorithm that extends the more robust item-based system by combining rating similarity with semantic similarity measures.

In the rest of this section we provide the details of the standard CF algorithms we have used in our experiments. We also briefly discuss the semantically enhanced hybrid algorithm, but leave the detailed description of that algorithms to Section 6.

4.1 User-Based Collaborative Filtering

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.

4.2 Item-Based Collaborative Filtering

Item-based collaborative filtering works by comparing items based on their pattern of ratings across users. Again, a nearest-neighbor approach can be used. The kNN algorithm attempts to find k similar items that are co-rated by different users similarly.

For our purpose we have adopted the adjusted cosine similarity measure introduced by Sarwar et al. [2001]. The adjusted cosine similarity formula is given by:

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

where $r_{u,i}$ represents the rating of user u on item i, and $\bar{r}_u$ is the average of the user u's ratings as before. After computing the similarity between items we select a set of k most similar items to the target item and generate a predicted value by using the following formula:

$$p_{u,i} = \frac{\sum\limits_{j \in J} r_{u,j} * sim_{i,j}}{\sum\limits_{j \in J} sim_{i,j}}$$

where J is the set of k similar items, $r_{u,j}$ is the prediction for the user on item j, and $sim_{i,j}$ is the similarity between items i and j as defined above. We consider a neighborhood of size 20 and ignore items with negative similarity. 'The idea here is to use the user's own ratings for the similar items to extrapolate the prediction for the target item. As in the case of user-based algorithm, if the denominator of the above equation is zero, our algorithm replaces the prediction by average rating of that user u.

4.3 Semantically Enhanced Collaborative Filtering

It seems clear that hybrid recommendation should offer something of a defense against profile injection attacks. A system that has multiple recommendation components, only one of which is collaborative, does not rely solely on profile data and is therefore buffered, to some degree, from the manipulation of that data. Or, it may be that an attacker will have to attack all of the components in order to be successful. A useful analogy can be seen in the Google search engine.[1] Its PageRank algorithm combines both knowledge-based (keyword) comparison and collaborative (link-based) authority measures. [Brin and Page 1998] As a result, it is relatively immune from attacks that manipulate the content of web pages, and attackers must attempt to manipulate the authority mechanism as well as the knowledge-based algorithm in order to bias its results.

There are many different types of hybrids that can be built with a collaborative recommendation component. (See [Burke 2002] for a survey of different hybrid designs.) For the purposes of our study, we sought a hybrid that would enable us to adjust the degree of dependence on the collaborative part, thereby giving a quantitative notion of the tradeoff between the degree of hybridization and the protection against attack. Thus, we chose to use a weighted hybrid, a design that combines the predictions of multiple components into a single score using a weighted sum. In this paper, we report on results using a knowledge-based / item-based collaborative weighted hybrid. We plan to explore other hybrid designs, including those using content-based recommendation, in our future work.

Our design for knowledge-based / collaborative weighted hybrid recommendation algorithm is known as *semantically-enhanced collaborative filtering*. [Jin and Mobasher 2003; Mobasher et al. 2004]. The knowledge-based component of the system uses structured semantic knowledge about items based on domain-specific reference ontologies to calculate these content similarities. The semantic content similarities among items are then combined with the rating similarity among items to produce the final predictions. Our hybrid design, is therefore, an extension of the item-based collaborative filtering algorithm. Further details of the algorithm are provided, along with the results in Section 6.

4.4 Evaluation Metrics

There has been considerable research in the area of recommender systems evaluation [J.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, we are interested not in raw performance, but rather in the change in performance induced by an attack. 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:

[1] www.google.com

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

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

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

Note that a strong prediction shift is not a guarantee that an item will be recommended – it is possible that other items' scores are affected by an attack as well or that the item scores so low to begin with that even a significant shift does not promote it to "recommended" status. Thus, in order to measure the effectiveness of the attack on the pushed item compared to other items, we introduce the hit ratio metric. Let R_u be the set of top N recommendations for user u. If the target item appears in R_u, for user u, the scoring function H_{ui} has value 1, otherwise it is zero. Hit ratio for an item i is given by

$$HitRatio_i = \sum_{u \in U_T} H_{ui} / \left| U_T \right| .$$

Likewise average hit ratio can then calculated as the sum of the hit ratio for each item i following an attack on i across all items divided by the number of items:

$$\overline{HitRatio} = \sum_{i \in I_T} HitRatio_i / \left| I_T \right| .$$

For nuke attacks, where the purpose is to decrease the predicted rating of an item, average rank is used. Average rank captures the relative predicted rating of a target item following an attack. This measure better captures differences in negative shift since for nuke attacks, target items commonly fall out of traditional hit ratio windows making hit ratio differences insignificant. Let T_u be the set of predicted ratings for unrated items for user u. For each attack on item i let $Rank_{ui}$ be defined as the position of item i in the set T_u sorted descending based on predicted rating. Likewise $AvgRank$ can then be calculated as the sum of $Rank_{ui}$ across all users u divided by the number of users:

$$\overline{AvgRank_i} = \sum_{u \in U} Rank_{ui} / \left| U \right| .$$

We plan to explore other metrics based on recommendation behavior, such as the bin-based techniques used in Lam and Riedl [2004] and others, in our future work.

5. COMPARATIVE RESULTS: USER-BASED AND ITEM-BASED ALGORITHMS

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

In all experiments, we use a neighborhood size of 20 in the k-nearest-neighbor algorithms for user-based, item-based and semantically enhanced hybrid systems.

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

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.

Ratings	Average Rating 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 prediction shift, hit ratio, and average rank, as described earlier, to measure the relative performance of various attack models. Generally, the values of these metrics are plotted against the size of the attack 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 the vulnerabilities of both user-based and item-based collaborative filtering against push attacks. Next we report how these two algorithms respond to nuke attacks and some interesting differences in the effectiveness of the various attack models depending on whether they are used for nuke or push attacks.

5.1 Push Attacks Against User-Based Collaborative Filtering

The average attack was shown to be highly successful in prior work and our initial investigations also indicated that this was the case. However, the knowledge requirements for the average attack are substantial. The attacker must collect mean rating information for every item in the system. A natural question to ask is what is the dependence between the

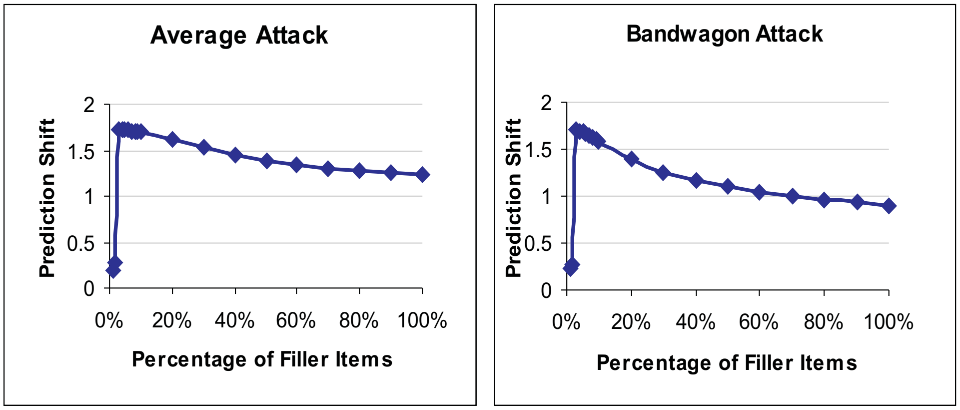

Fig. 8. Prediction Shift vs. Filler Size: User-Based algorithm.

power of the attack and the amount of knowledge behind it? Can we reduce the amount of knowledge used to generate the attack and still be successful?

To investigate this question, we varied the "filler size", $|I_F|$ (see Definition 1 and Figure 3). This is the number of ratings for the filler items added to fill out the attack profile, and thus is directly related to the amount of effort and, in the case of the average attack, knowledge required to mount the attack.

To magnify the impact of this manipulation, we used a large attack size of 15%. Figure 8 shows the rather surprising results of these experiments. As the amount of the knowledge increases to around 3%, the prediction shift rises sharply to around 1.6 but after this point it drops off gradually ending at 1.3 with a filler size of 100%. A similar effect was seen at smaller attack sizes.

This would appear to be a consequence also of Zipf's law: most users will not have rated more than a small fraction of the product space; a person can only see so many movies. An attacker, therefore, only needs to use part of the product space to make the attack effective. An attack has to achieve a balance between coverage (including enough movies so that the attack profiles will have some chance of being similar to a large number of users) and generality (every movie that is included creates the possibility that the profile will be dissimilar to any given user.) What is surprising is that the optimum of this trade-off appears to come with so few ratings.

These results also show that a similar phenomenon can be observed in the bandwagon attack. Recall that in this case the attacker does not need to know anything system-specific, merely a list of items that are likely to be densely rated. The attacker selects k such items and rates them highly along with the target item. The filler size, in this case, is the proportion of the remaining items that are assigned random ratings based on the overall data distribution across the whole database (see Figure 5). 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. The attack profiles consist of high ratings given to these popular titles in conjunction with high ratings for the pushed movie. Figure 8 shows the effect of filler size on the effectiveness of this attack. In this particular experiment, we selected only a single popular movie as the "bandwagon" movie with which to associate the target and used an attack size of 10% (i.e., the number of attack profiles were about

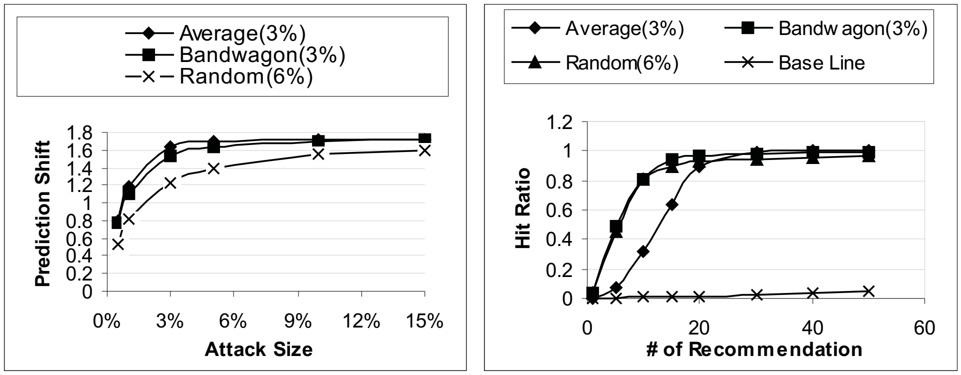

Fig. 9. Comparison of attacks against user-based algorithm, prediction shift (left) and hit ratio(right). The baseline in the right panel indicates the hit ratio results prior to attack

10% of the size of the original database).

For the bandwagon attack, the best results were obtained by using a 3% filler size (set I_F containing ratings for approximately 3% of the movies in the database.) This number seems to correspond closely to the average number of movies per user in the database. For subsequent experiments with this attack model we use 5 "bandwagon" movies. The five movies chosen were those with the most ratings in the database. 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[3],[4] and found they would be among anyone's list of movies likely to have been seen by many viewers.

Figure 9 shows the results of a comparative experiment examining three algorithms at different attack sizes. The algorithms include the average attack (3% filler size), the bandwagon attack (using 1 frequently rated item and 3% filler size), and the random attack (6% 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 minimal knowledge requirement. All that is necessary for an attacker is to identify a few items that are likely to be rated by many users.

5.2 Push Attacks Against Item-Based Collaborative Filtering

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 Riedl [2004], confirming their effectiveness against the user-based algorithm. We can also confirm that these attacks are less

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

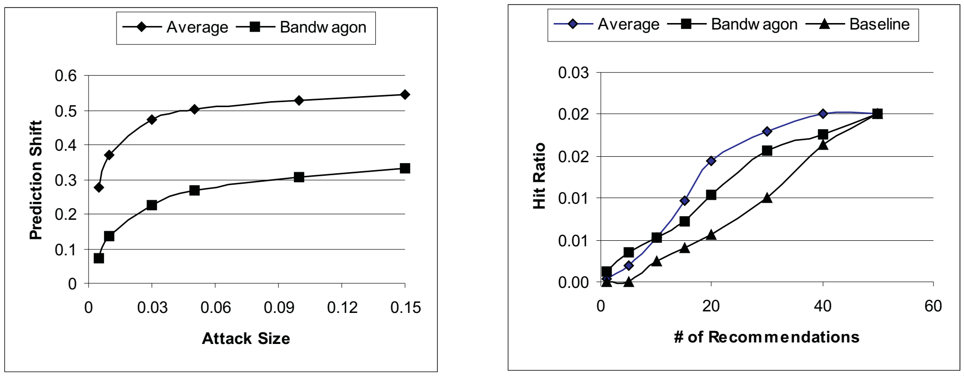

Fig. 10. Prediction Shift(left) and HitRatio(right) results for Average and Bandwagon Attack against Item-Based Collaborative Filtering. The Baseline in the right panel indicates the hit ratio results prior to attack

effective against an item-based formulation of collaborative recommendation. Figure 10 shows the results for this condition. Note that the prediction shift curves are significantly lower. The difference in the hit ratio curves are particularly dramatic. The attack curves are only slightly different from the pre-attack baseline, and never exceed 0.03. Compare this to Figure 9, where hit ratio nears 1.0 for all attacks around recommendation set size of 20.

Unlike the average, random and bandwagon attacks, the segment attack was designed specifically to impact an item-based algorithm. It aims to increase the column-by-column similarity of the target item with the user's preferred items. If the target item is consider similar to something that the user likes, then its predicted rating will be high – the goal of the push attack.

Recall that 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. 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,

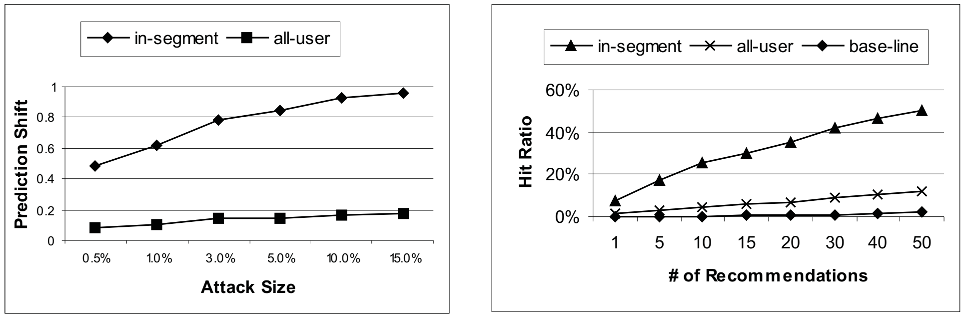

Fig. 11. Prediction shift and hit ratio results for the Horror Movie Segment in item-based algorithm.

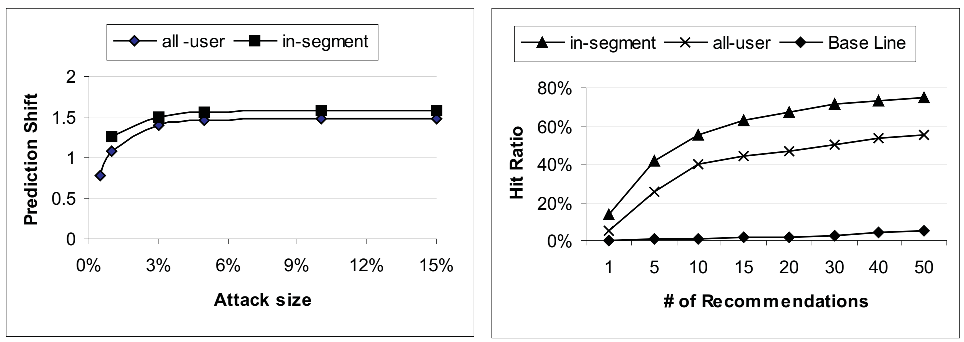

Fig. 12. Prediction shift and hit ratio results for the Horror Movie Segment in user-based algorithm.

namely, *Alien*, *Psycho*, *The Shining*, *Jaws*, and *The Birds*.[5] 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.

The power of the segmented attack is emphasized in Figure 11 in which the impact of the attack is compared within the targeted user segment and within the set of all users. The left panel in the figure shows the comparison in terms of prediction shift and varying attack sizes, while the right panel depicts the hit ratio at 1% attack.[6] 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. Indeed, in the case of in-segment users, the hit ratio is much higher than average attack. The chart also depicts the effect of hit ratio before any attack. Clearly the segmented attack has a bigger impact than any other attack we have previously examined against item-based algorithm. Our prediction shift results show that the segmented attack is more effective against in-segment users than even the more knowledge-intensive average attack for the item-based collaborative algorithm. 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

[5]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.

[6]Note that our previous hit ratio figures have used 10% attack size. Here we see comparable performance with 1/10 of the number of attack profiles.

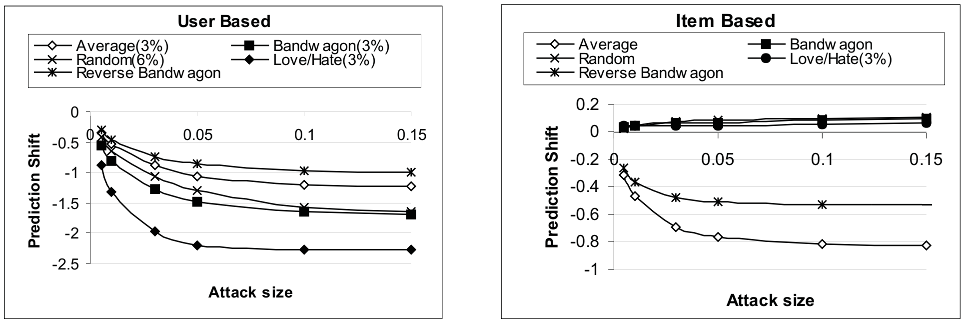

Fig. 13. Prediction Shift Results for Nuke Attack

paper.

Although designed specifically as an attack against the item-based algorithm, it turns out that the segment attack is also effective against the user-based algorithm. See Figure 12. The prediction shift for the in-segment users here is almost as good as the average attack results shown in Figure 12.

5.3 Nuke Attacks Against Item-Based and User-Based Algorithms

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, two attack models tailored specifically for this task, namely the *love/hate* and the *reverse bandwagon* attacks.

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 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.

The reverse bandwagon attack is tailored for the item-based algorithm. The item-based algorithm extrapolates from the user's own ratings of items and therefore, the predictions are based on what items in the user's profile are determined to be similar to the target item. The aim of the attacker therefore must be to push the target item closer to items that the user does not like. Instead of associating an item that we want the user to like with other well-liked items, we associate the item we want to nuke with other generally-disliked ones. The knowledge needed to mount a reverse bandwagon attack would seem to be greater than the ordinary bandwagon attack. We know that there is considerably less general agreement on disliked movies than liked ones and there are fewer system-external resources for identifying exactly what these densely-rated but disliked items might be.

The items with the lowest average rating that meet a minimum threshold in terms of the number of user ratings in the system are selected as the selected item set, as described in detail in section 2.2. Our experiments were conducted using $|I_S| = 25$ with a minimum of 10 users rating each movie.

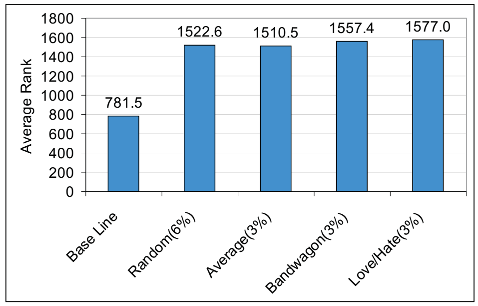

Fig. 14. Average Rank Results for Nuke Attacks (User-Based algorithm)

Figure 13 shows the experimental results for all attack models at 10% attack sizes, with the user-based algorithm on the left and the item-based on the right. Despite the minimal knowledge required for the love/hate attack, this attack proves to be the most effective at nuking items of these attacks against the user-based algorithm. Among the other attacks, the bandwagon 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 love/hate attack produced a positive prediction shift slightly over 1.0 for a push attack of 10% against the user-based algorithm, which is much less effective than even random attack. However when used to nuke an item against the user-based algorithm, this model is by far the most effective model we have tried, with a prediction shift of almost twice that of the 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. Bandwagon attack on the other hand performed nearly as well as average attack in pushing items, and had superior overall performance for nuking, despite its lower knowledge requirement.

The item-based proved far more robust overall. The average attack is the most successful nuke attack here with reverse bandwagon close behind. However, note the difference in scale from the left half of the figure: the average attack can do no better than about -0.8 against the item-based algorithm, whereas the best attack against user-based (love/hate) had a prediction shift of over -2.

The asymmetries between push and nuke continue as we examine the item-based results. The random and love/hate attacks are poor performers for push attacks, but as nuke attacks, they actually fail completely to produce the desired effect. Reverse bandwagon (but not bandwagon) proves to be a reasonable low-knowledge attack model for a nuke attack against the item-based algorithm.

Hit ratio, which was used as an alternate metric for push attacks, make less sense as a measure of the effectiveness of a nuke attack. A nuked item will quickly drop out of the retrieval set windows over which hit ratio is measured, making hit ratio differences insignificant. So, for this condition, we use the *average rank* metric, measuring at what rank the target item appears in the retrieval set. A successful nuke attack will increase the average rank of the target item, pushing it out of consideration for most users.

In Figure 14, we see average rank results for the user-based algorithm comparing the five attacks (at 10% attack) against a pre-attack baseline. The results confirm the prediction shift findings with the love/hate attack showing the largest impact, but with the other attacks perhaps differing from each other somewhat less than the prediction shift results would indicate.

6. SEMANTICALLY ENHANCED HYBRID COLLABORATIVE RECOMMENDATION

In this section, we focus our attention on a knowledge-based / collaborative hybrid recommendation algorithm which, we believe, represents a potential solution to the profile injection attack problem. The reason this algorithm is more robust against such attacks is that it relies not only on user profiles, but also on semantic knowledge of the domain and items, in order to make predictions. It is therefore less affected by the injection of bogus user profiles into the system. The main question, however, is whether such a hybrid algorithm can be as accurate or effective as the standard algorithms based purely on user profiles. We show, in this section, that the proper combination of user-based and semantic knowledge, can not only ensure accurate predictions in par with standard CF algorithms, but it can also dramatically reduce the impact of profile injection attacks.

6.1 The Hybrid Recommendation Algorithm

Our semantically enhanced collaborative recommendation algorithm is a hybrid algorithm that integrates semantic information with item based collaborative recommendation [Jin and Mobasher 2003; Mobasher et al. 2004]. Item-based recommendation relies on the similarity of ratings between items. This hybrid approach extends item-based similarity by combining it with content based similarity.

The algorithm uses structured semantic knowledge about items in domain specific reference ontologies to calculate these content similarities. For example, in the movie domain, a reference domain-specific ontology may contain classes such as **movie**, **actor** and **director** along with their attributes. The attribute of the **movie** class, include *case*, *genre*, *synopsis*, *director*, etc.

In order to facilitate the computation of item similarities, generally, the extracted class instances will need to be converted into a vector representation. In our case, the values of semantic attributes associated with class instances are collected into a relational table whose rows represent the n items, and whose columns correspond to each of the extracted attributes. The final result is a $n \times d$ matrix S, where d is the total number of unique semantic attributes called the *attribute matrix*.

To reduce noise and collapse highly correlated attributes, Latent Semantic Indexing (LSI) is used on the attribute matrix to reduce dimensionality [Berry et al. 1995]. Singular Value Decomposition (SVD), a well-known technique used in LSI, is applied to perform matrix decomposition. In our case, we perform SVD on the attribute matrix $S_{n \times d}$ by decomposing it into three matrices:

$$S_{n\times d} = U_{n\times r} \bullet \Sigma_{r\times r} \bullet V_{r\times d}$$

where U and V are two orthogonal matrices; r is the rank of matrix S, and Σ is a diagonal matrix of size $r \times r$, where its diagonal entries contain all singular values of matrix S and are stored in decreasing order. One advantage of SVD is that it provides the best lower rank approximation of the original matrix S [Berry et al. 1995]. We can reduce the diagonal matrix Σ into a lower-rank diagonal matrix $\Sigma_{k\times k}$ by only keeping k ($k < r$) largest values.

Accordingly, we reduce U to U' and V to V'. Then the matrix $S' = U' \bullet \Sigma' \bullet V'$ is the rank-k approximation of the original matrix S.

In the above process, U' consists of the first k columns of the matrix U corresponding to the k highest order singular values. In the resulting attribute matrix, S', each item is, thus, represented by a set of k latent variables, instead of the original d attributes.

Once the reduced dimension attribute matrix has been obtained, the semantic similarities are then integrated into the standard item-based framework. The semantic similarity and user-item rating similarity are then combined into a single similarity measure, as a linear combination of the two similarities to perform item-based collaborative recommendation.

The semantic similarity measure $\mathit{SemSim}(i_p, i_q)$, for a pair of items i_p and i_q, is computed using the standard vector-based cosine similarity on the reduced semantic space. This process can be viewed as multiplying the matrix S' by its transpose and normalizing each corresponding row and column vector by its norm. This results in a $n \times n$ square matrix in which an entry i, j corresponds to the semantic similarity of items i and j.

Similarly, we compute item similarities based on the user-item matrix M. We employ the adjusted cosine similarity, described earlier, in order to take into account the variances in user ratings. We denote the rating (or usage) similarity between two items i_p and i_q as $\mathit{RateSim}(i_p, i_q)$.

Finally, for each pair of items i_p and i_q, we combine these two similarity measures to get *CombinedSim* as their linear combination:

$$\mathit{CombinedSim}(i_p, i_q) = (1 - \alpha) \cdot \mathit{SemSim}(i_p, i_q) + \alpha \cdot \mathit{RateSim}(i_p, i_q)$$

where α is a *semantic combination parameter* specifying the weight of semantic similarity in the combined measure. If $\alpha = 1$, then $\mathit{CombinedSim}(i_p, i_q) = \mathit{RateSim}(i_p, i_q)$, in other words we have the standard item-based recommendation. On the other hand, if $\alpha = 0$, then only the semantic similarity is used which, essentially, results in a form of content-based recommendation. The appropriate value for α is found by performing sensitivity analysis for the particular data set as shown in our experimental results in the remainder of this section.

6.2 Push Attacks Against Semantically Enhanced Hybrid Algorithm

Since our hybrid algorithm is an extension of the item-based collaborative recommendation, in these experiments we focus on comparing the robustness of the hybrid to that of the item-based algorithm.

To build the hybrid system, we obtained semantic data for movies using the methodology described in Mobasher et al. [2004]. Specifically, an agent was used to extract movie instances from the Internet Movie Database (www.imdb.com). Semantic attributes such as movie title, release year, director(s), cast, genre, and plot were extracted for each instance. The attributes were then used to form a binary attribute vector with continuous data types discretized. Singular value decomposition was then used to reduce the attribute vectors from 2762 to 60 dimensions. Experiments were conducted using the same target items and user sets as described earlier for horror segment attack and average attack at a filler sizes of 100%.

Using a 10% attack size to examine the effectiveness of the hybrid algorithm, Figure 15 shows, as expected, that α can be adjusted to decrease the impact of a profile injection attack for both segment attack and the more traditional average attack. However the more interesting aspect of these results is that the integration of semantic information greatly

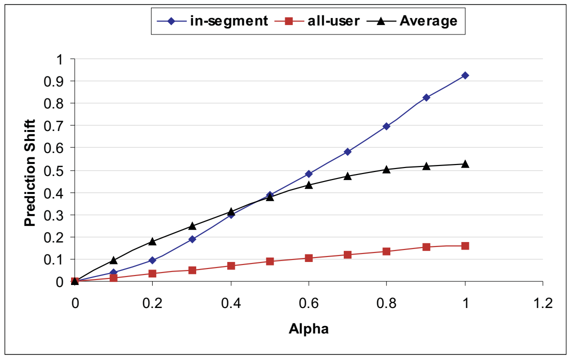

Fig. 15. Semantically enhanced algorithm: comparison of segment and average attack.

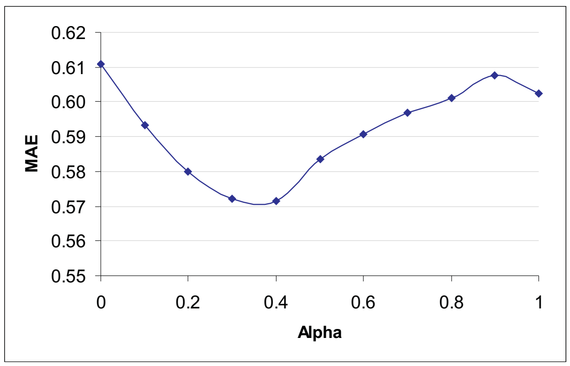

Fig. 16. Effect of varying α on MAE.

reduces the bias injected by segment attack. As previous experiments have shown, segment attack is effective against the item-based algorithm by being able to increase the similarity between a pushed item and a group of items liked by a segment of users. However as Figure 15 shows, the injection of semantic similarity is particularly effective at reducing the ability of an attacker to manipulate the similarity between target movies and the more semantically similar segment movies.

Obviously using only semantic similarity would provide complete protection from ratings attacks, however since the use of ratings data is known to improve accuracy it is advantageous to select $\alpha > 0$. To select the mix of semantic and rating data, we would like to be able to select the combination that provided the highest accuracy. To determine this, we performed an analysis of MAE (Mean Absolute Error) for the semantically enhanced algorithm to select the value of α that provided the highest accuracy. As depicted in Figure 16, $\alpha = .4$ or a blend of 40% item rating similarity and 60% semantic similarity yielded the lowest MAE, and thus the highest prediction accuracy.

In the rest of our experiments, we fix α at 0.4 and compare the resulting hybrid system with our unhybridized item-based recommender. Figure 17 shows prediction shift results for the Average and Segmented attacks (Horror segment) against the unhybridized system

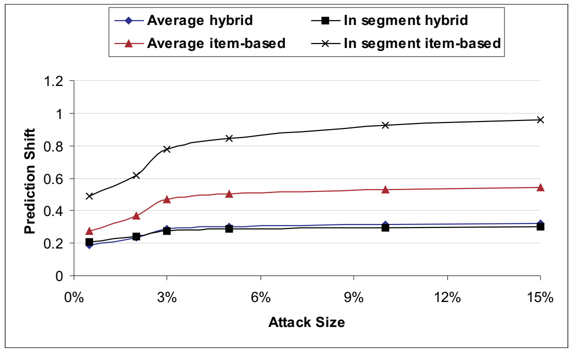

Fig. 17. Comparison of prediction shift with semantic enhancement.

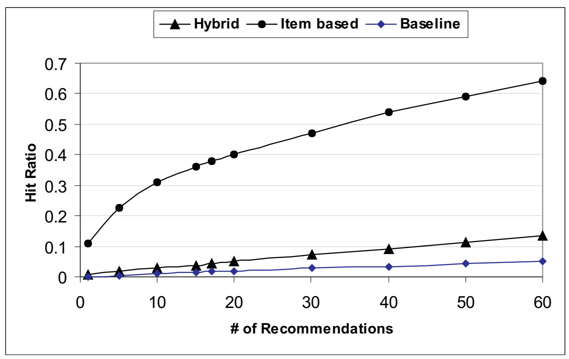

Fig. 18. Hit ratio comparison of in-segment users.

and the hybrid for different attack sizes. The prediction shift is much lower across the entire range of attack sizes.

Even more dramatic are the hit ratio results shown in Figure 18. One of aspects which made segment attack particularly effective against the item-based algorithm was its ability to increase the similarity of the target item with the segment, but it also decreased the similarity of other possible recommendations resulting in drastic hit ratio changes. The hybrid algorithm almost entirely negates this effect, with the hit ratio for the hybrid remaining very close to the pre-attack baseline especially for smaller result set sizes. (The pre-attack baseline depicted is for the semantically enhanced algorithm, the item-based baseline was not significantly different.)

7. DEFENSE AGAINST PROFILE INJECTION ATTACKS

The vulnerabilities of collaborative recommendation were well-established theoretically in prior work. The results shown above demonstrate that these vulnerabilities are of more than just theoretical interest. A collaborative recommender using any of the common algorithms can be exploited by attackers without a great degree of knowledge of the system. We have established that hybrid recommendation offers a strong defense against profile injection attacks, and indeed, the weighted hybrid shown here reduces the impact of attacks from serious body blows to the system's integrity to mere annoyances for the most part. Such hybrids should be seriously considered by implementers interested in robustness and capable of deploying them.

However, even the hybrid system is not unaffected by profile injection attacks, and no collaborative system could be. As long as we accept new profiles into the system and allow them to affect its output, it is possible for an attacker to perform these types of manipulations. Furthermore, in some domains it may not be possible to obtain the necessary semantic domain knowledge for constructing hybrid systems. Therefore, in addition to algorithm robustness, attention must be paid to effective methods for detecting and neutralizing attacks.

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[7] 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.

7.1 Detection Techniques

There have been some recent research efforts aimed at detecting and preventing 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 [Su et al. 2005]. O'Mahony, Hurley and Silvestre (2004) developed several techniques to defend against the attacks described in [Lam and Riedl 2004] and [O'Mahony et al. 2004], including new strategies for neighborhood selection and similarity weight transformations. 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 entails 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 section, we examine approaches that detect attacks conforming to known attack models, such as those we have discussed above. Of course, nothing compels an attacker to produce profiles that have these exact 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. In this way, we hope to minimize the potential harm profile injection can cause.

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

7.2 Detection Attributes for Profile Classification

Prior work in detecting attacks in collaborative recommender systems have employed ad hoc algorithms for identifying basic attack models such as the random attack [Chirita et al. 2005]. Our approach uses supervised learning and is 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. We compute statistics over the profile data and use attribute reduction techniques to create a lower dimensional training set. This training set is a combination of user data from the MovieLens dataset and attack profiles generated using our attack models. 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.

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.

7.2.1 *Generic Attributes for Detection.* 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 Riedl 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:

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

where n_u is the number of items user u rated, $r_{u,i}$ is the rating given by user u to item i, l_i is the number of ratings provided for item i by all users, and $\overline{r_i}$ is the average of these ratings.

—*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. It differs from RDMA only in that the number of ratings for an item is squared in the denominator inside the sum, thus reducing the weight associated with items rated by many users. The WDMA attribute is given by:

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

—*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 discounts the average similarity 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 is 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:

$$\mathit{LengthVar}_u = \frac{|n_u - \overline{\overline{n}}|}{\sum_{k \in U} (n_k - \overline{\overline{n}})^2}$$

where $\bar{n}$ is the average number of ratings across all users.

7.2.2 *Model-Specific 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; 2006a; 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.)

The first step is to divide the profile into the three partitions: the target item (having 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_t in the profile P_u of user u where p_t is from the set of items $P_{u,t}$ in P_u that are given the rating r_t (the maximum rating for push attack detection or the minimum rating for nuke attack detection).

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

where P_u is the profile of user u, p_{target} is the hypothesized target item, $r_{u,i}$ 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 $P_{u,t}$, 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*, the partitioning metric described above.

—*Filler Mean Difference*, which is the average of the absolute value of the difference between the user's rating and the mean rating (rather than the squared value as in the variance.)

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

The next set of attributes are used to detect attacks that target a group of items such as the bandwagon and segment 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:

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

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.

7.2.3 *Intra-profile 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 many 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 likely 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:

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

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

7.3 Experiments With Profile Classification

In the experiments below we apply kNN supervised classification and show that the attributes described above can be effective at detecting and reducing the impact of several of the attack models described above. For our detection experiments, we used the same Movie-Lens 100K dataset[8] used in Section 5. To minimize over-training, the dataset was split into two equal-sized partitions. The first partition was made into a training set, while the second was used for testing and was unseen during training. The training data was created by inserting a mix of average, random, bandwagon, and segment push attacks as well as average, random, and love/hate nuke attacks at various filler sizes that ranged from 3% to 100% and attack sizes between .5% and 1%. To minimize over-training, the segment attack training data was created using the Harrison Ford segment while testing was executed on the 6 combinations of Horror segment movies.

Specifically the training data was created by inserting a training attack at a particular filler size and attack size into the training data set, and generating the detection attributes and class labels for the authentic and attack profiles. This process was repeated for each subsequent training attack by inserting the attack profiles into a copy of the original training data set, then generating the detection attributes. For all these subsequent attacks, the detection attributes of only the attack profiles were then added to the original detection attribute dataset. This approach allowed a larger attack training set to be created while minimizing over-training for larger attack sizes.

Our classifiers use a total of 15 detection attributes: 6 generic attributes (WDMA, RDMA, WDA, LengthVar, DegSim $k = 450$, and DegSim' $k = 2$ with co-rating discounting $d = 963$); 6 average attack model attributes (3 for push, 3 for nuke – Filler Mean Variance, Filler Mean Difference, Profile Variance); 2 group attack model attributes (1

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

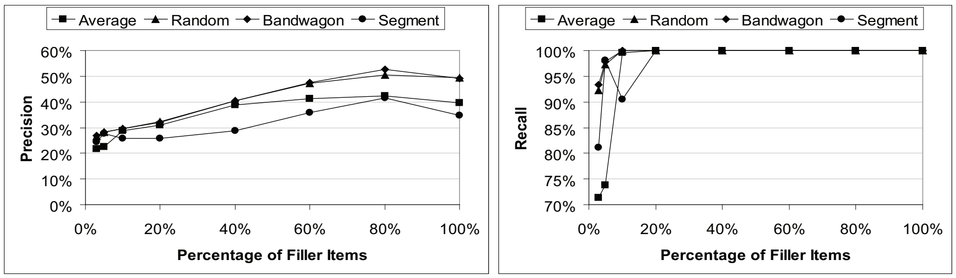

Fig. 19. Detection precision (left) and recall (right) for 1% push attacks.

for push, 1 for nuke – FMTD); 1 target detection model attribute (TMF). Based on this training data, kNN with $k = 9$ was used to make a binary profile classifier. The kNN classifier were implemented using Weka [Witten and Frank 2005]. using one over Pearson correlation distance weighting.

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 of the data. A single training data set was used in all the detection experiments. 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. We used the same 50 users and movies as in the experiments in Section refsec:experiments. The results represent averages over the combinations of test users and test movies. For prediction shift, the "without detection" results are taken from Section 5.

For measuring classification performance, we use the standard measurements of precision and recall. The basic definition of recall and precision can be written as:

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

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

Since we are primarily interested in how well the classification algorithms detect attack, 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. In addition to these classification metrics, we also use the measures of MAE and prediction shift as described in Section 4.

In our first experiment, we examine the effectiveness of profile classification detection across filler sizes for 1% attacks. As Figure 19 depicts, for all push attack models the larger the filler size the easier it is to differentiate attack profiles from authentic profiles.[9] This is intuitive as the more rating examples provided by a profile, the more apparent patterns within a profile would become. Also, because few users rate large numbers of items, the *LengthVar* attribute becomes an increasingly useful discriminator at these large profile

[9]For our detection experiments, we use a variant of the average attack. Rather that picking a fixed subset of the data as the filler set, the filler items are chosen randomly. This version of the attack is not knowledge-reduced since knowledge of the whole ratings distribution is still needed. However, it is much more difficult to detect since the set of rated items changes from profile to profile.

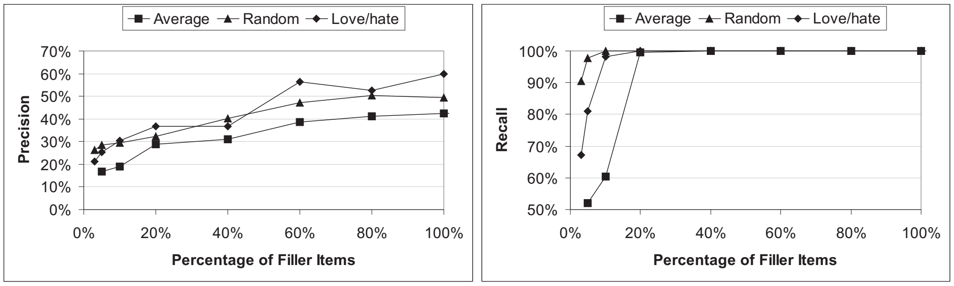

Fig. 20. Detection precision (left) and recall (right) for 1% nuke attacks.

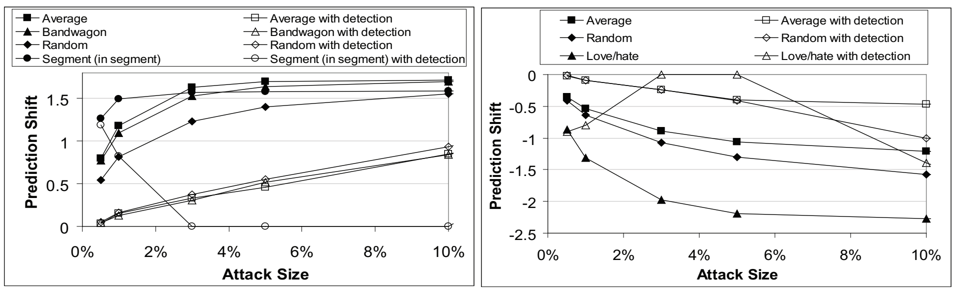

Fig. 21. Prediction shift comparison of user-based algorithm with and without detection, push attacks (left) and nuke attacks (right).

sizes. As the recall shows, while some attack profiles may go undetected at low filler sizes, for any filler size over 20% all attack profiles are detected successfully.

Similar patterns emerge from the nuke attack classification results shown in Figure 20. At higher filler sizes nuke attacks become easier to detect while several attack profiles go undetected at lower attack sizes. A closer examination comparing the push results with the nuke results shows that the average and random nuke attacks are slightly more difficult to detect in terms of recall, than the equivalent push attacks.

It should be noted while precision may seem low, this is due to the much higher number of authentic profiles compared to attack profiles for a 1% attack. This is exhibited by the lowest overall classification accuracy for any filler size and any attack for either push or nuke still being above 97%. In the context of detection, the impact of precision being less than 100% means some authentic users are not being included in collaborative 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. To test this possibility, the predictive accuracy of the 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, which is not statistically significant based on a 95% confidence interval. Thus detection can be added without a significant impact to predictive accuracy.

Figure 21 shows the prediction shift results for the push and nuke attacks with and without detection against the user-based algorithm. For all attacks against the detection enhanced recommender, a filler size of 3% was used as this was shown in the above classification results to be the most likely to avoid detection (i.e. lowest recall).

Under most conditions, the detection works well enough to exclude many of the attack profiles, cutting the prediction shift by a factor of two or more in some cases. The segment attack results exhibit an interesting pattern in that at low attack sizes enough attack profiles go undetected to produce nearly as much bias as without detection. As the attack size increases, however, the unusually high focus on the segment movies and target movie allows the attack profiles to be more easily detected via the TMF intra-profile attribute.

The right half of the figure shows a similar pattern. Most of the attacks are greatly reduced in impact. The love/hate attack is hard to detect at low attack sizes, but at larger sizes its impact is reduced, rising again at larger ones.

8. CONCLUSIONS

This paper has shown several key findings in the area of attacks against recommender systems. We have shown that it is possible to mount successful attacks against collaborative recommender systems without substantial knowledge of the system or users. The examination of the segment attack, a very effective reduced-knowledge attack, also demonstrated the vulnerability of the item-based algorithm, which was previously thought to be relatively robust.

We discovered that mounting a nuke attack is more complex than simply inverting the rating for the target item in the push version of the attack. Some attacks, such as the bandwagon attack, which are effective for push attacks are notably less unsuccessful for nuke attacks and vice-versa. As part of this investigation we introduced two new limited-knowledge nuke attacks, the love/hate attack and the reverse bandwagon attack. These attack models were both successful at limiting the knowledge required to leverage an attack. In fact, the love/hate attack proved to not only require the least amount of knowledge of the system, it also was the most effective attack of this type.

Hybrid recommender systems, which combine collaborative recommendation with other types of recommendation components, seem likely to provide defensive advantages for recommender systems. We have been able to empirically demonstrate the advantages of hybrid recommendation for robustness, using a weighted hybrid with a knowledge-based component. The semantically enhanced item-based algorithm described here improves over the standard item-based algorithm in both prediction accuracy and robustness. It may be possible in some cases for an attacker to find ways to bias the inputs of several recommendation components at the same time (as the experience of search engine spam shows) but this would certainly increase the cost and difficulty for the attacker.

Finally, we have presented a supervised classification approach for attack detection. Using a mix of statistical and model-derived features, we were able to demonstrate greatly increased stability in the face of common attacks. Some attacks, particularly the segment attack (push) and love/hate attack (nuke), still present problems as they can impact the system's recommendations even at low and hard-to-detect attack sizes. We are investigating techniques borrowed from statistical process control to help detect these problematic attacks.

Users' trust in a 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 multifacted 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 evenhandedly 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

ALBERT, M. AND AHA, D. 1991. Analyses of instance-based learning algorithms. In *Proc. 9th Nat. Conf. Artificial Intelligence*. AAAI, Morgan Kaufmann.

BERRY, M., DUMAIS, S., AND OBRIEN, G. 1995. Using linear algebra for intelligent information retrieval. *SIAM Review 37*, 573–595.

BILLSUS, D. AND PAZZANI, M. 2000. User modeling for adaptive news access. *User-Modeling and User-Adapted Interaction 10,* 2-3, 147–180.

BREESE, J., HECKERMAN, D., AND KADIE, C. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *Uncertainty in Artificial Intelligence. Proceedings of the Fourteenth Conference*. Morgan Kaufman, New Orleans, LA, 43–53.

BRIN, S. AND PAGE, L. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems 30,* 1-7, 107–117.

BURKE, R. 2000. Knowledge-based recommender systems. In *Encyclopedia of Library and Information Systems*, A. Kent, Ed. Vol. 69. Marcel Dekker, New York.

BURKE, R. 2002. Hybrid recommender systems: Survey and experiments. *User Modeling and User Adapted Interaction 12,* 4, 331–370.

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. 2006a. Classification features for attack detection in collaborative recommender systems. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'06)*.

BURKE, R., MOBASHER, B., WILLIAMS, C., AND BHAUMIK, R. 2006b. Detecting profile injection attacks in collaborative recommender systems. In *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.

HAUSSLER, D. 1990. Probably approximately correct learning. In *Proceedings of the 8th National Conf. on Artificial Intelligence*. AAAI, Morgan Kaufmann, 1101–1108.

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. L., FRANKOWSKI, D., SCHAFER, J. B., AND SEN, S. 2006. Collaborative filtering. In *The Adaptive Web: Methods and Strategies of Web Personalization*, in press ed., P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds. Springer Verlag.

J.HERLOCKER, KONSTAN, J., TERVIN, L. G., AND RIEDL, J. 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems 22,* 1, 5–53.

JIN, X. AND MOBASHER, B. 2003. Using semantic similarity to enhance item-based collaborative filtering. In *Proceedings of The 2nd IASTED International Conference on Information and Knowledge Sharing*. Scottsdale, AZ.

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

LANG, K. 1995. Newsweeder: Learning to filter news. In *Proceedings of the 12th International Conference on Machine Learning*. 331–339.

MASSA, P. AND AVESANI, P. 2006. Trust-aware collaborative filtering for recommender systems. In *Proceedings of the 11th International Conference on Intelligent User Interfaces*. Agia Napa, Cyprus.

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, to appear.

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.

MOBASHER, B., DAI, H., LUO, T., AND NAKAGAWA, M. 2001. Effective personalization based on association rule discovery from web usage data. In *Proceedings of the 3rd ACM Workshop on Web Information and Data Management (WIDM01)*. Atlanta, Georgia.

MOBASHER, B., JIN, X., AND ZHOU, Y. 2004. Semantically enhanced collaborative filtering on the web. In *Web Mining: From Web to Semantic Web*, B. B. et al., Ed. LNAI Volume 3209. Springer.

MOONEY, R. J. AND ROY, L. 1999. Content-based book recommending using learning for text categorization. In *SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation*. ACM SIGIR, Berkeley, CA.

O'DONOVAN, J. AND SMYTH, B. 2006. Is trust robust?: An analysis of trust-based recommendation. In *Proceedings of the 5th ACM Conference on Electronic Commerce (EC04).* ACM Press, 101–108.

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.

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.

SU, X.-F., ZENG, H.-J., AND CHEN., Z. 2005. Finding group shilling in recommendation system. In *WWW 05 Proceedings of the 14th international conference on World Wide Web*.

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