COMPGA99
Dissertation

Defence against Profile Injection Attacks on Collaborative Filtering

By [redacted] (MSc Information Security)
Supervised by DR J WANG (Adastral Park – University College London)
August 2008

Abstract

Collaborative filtering is a method to make personalised recommendations on information items, such as books and music, for a particular individual. It does this by looking at the items that like-minded people prefer. Collaborative filtering is used by many popular websites. For instance, the online retailer Amazon uses it to identify products that are likely to be of interest to its customers. The video-sharing website YouTube also uses collaborative filtering, to recommend videos to its users.

It has been previously shown that collaborative filtering is vulnerable to malicious manipulation. Attackers, who might want to make their products frequently and highly recommended, can try to introduce biased opinions into a recommender system. This could lead to unfair and inaccurate item recommendations being produced.

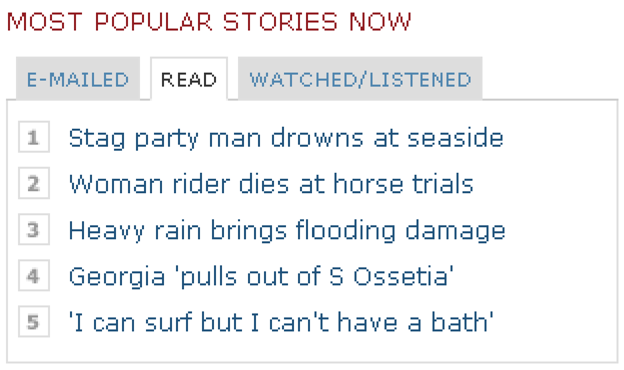
This report describes work that we have undertaken to identify collaborative filtering attacks. In particular, we present some novel classification features that can be used to very accurately detect and neutralise all these attacks. These unique features are derived from observations on the behaviour of users. Our most successful feature uses the observation that real users' opinions are not missing at random, i.e. such a user is not likely to give their opinion on an item they do not like.

Contents

1 Introduction

The Internet is not as safe as it used to be. Fraud, phishing and spam and the like are now very prevalent on the Internet. Unscrupulous people have forever been trying to deceive other people, and this is no different on the Internet. Weaknesses in security are almost always exploited by adversaries, especially if they can gain something from doing so.

Nowadays, there are a seemingly infinite number of items vying for our attention. This can be readily seen on the Web, such as in online stores, news sites and—of course—search engine results pages. Consequently, we expect ways to whittle available items down so that only relevant ones are left. Probably the most simplest and obvious approach would be to present a sorted list of the, say, five most popular items on offer. The BBC News website actually does this, showing each visitor its five currently most read stories (see Figure 1 below). While this has some value, what if the user is not interested in any of the stories? Or, what if a story the user would find interesting is not shown? And, what if they have already read one of the stories? Well the solution here would be to personalise the list of news stories for each individual user.


MOST POPULAR STORIES NOW

E-MAILED READ WATCHED/LISTENED

1 Stag party man drowns at seaside
2 Woman rider dies at horse trials
3 Heavy rain brings flooding damage
4 Georgia 'pulls out of S Ossetia'
5 'I can surf but I can't have a bath'

Figure 1: Screenshot of the five currently most read stories on the BBC News website (news.bbc.co.uk) as at 10:08 on 10 August 2008.

Such personalised recommendation lists can be produced using recommender systems, which try to identify items that are likely to be of interest to a particular user. To do this, a recommender system might compare the user’s preferences with those of the other users of the system, e.g. if Boris and Dave have similar tastes, Boris is likely to be interested in the items that Dave likes. This is a common recommendation technique, and is called *collaborative filtering* [1]. Moreover, it is the focus of this report. Collaborative filtering can be viewed as a scaled up, on-demand version of traditional word of mouth recommendation.

Users give their opinion on items to a collaborative recommender system, so that it can gauge who shares similar preferences. However, an unscrupulous person could be masquerading as one or more users. Suppose their objective is to make a certain item highly recommended. They could try to achieve this by introducing biased opinions on the item into the recommender system. This could lead to the item, which the attacker may have a special interest in, being recommended to more people than usual. The motivation is clear: highly recommended items tend to attract more interest. Continuing the previous example, the situation is similar to Gordon befriending Boris solely to make Boris buy a book that he wrote himself.

Collaborative filtering is seen as one of the most effective ways to alleviate information overload. The fact that a large number of high-profile websites, including Amazon, YouTube, and Last.fm, implement collaborative filtering is testimony to this. Therefore, the integrity of collaborative recommender systems is worth protecting.

1.1 Scope

This report concentrates on what are called *profile injection attacks* in the collaborative filtering literature. In particular, the emphasis is on understanding and characterising the Random, Average and Bandwagon profile injection attacks. The overriding aim of our project was to develop a new way to detect these attacks, because current methods for this purpose are far from perfect. This is especially true for the Average attack. This is the strongest attack, but because of this it turns out to be the hardest to detect too. So detection of the Average attack was given priority during the project.

1.2 Outline

Chapter 2 first introduces the collaborative filtering problem, some definitions that are used throughout this report, and two commonly used collaborative filtering algorithms (including how they work). Next, attacks on these algorithms from the literature are introduced and discussed. All these attacks use a technique called “profile injection”. Finally, the chapter looks at how these profile injection attacks are currently being detected.

Chapter 3 proposes three measurable properties (features) of the attacks that were defined in Chapter 2, for the purpose of attack detection. These features are called Ratings Missing At Random (RMAR), Rated Items Consistency (RIC) and Maximum Ratings (MaxRatings).

Chapter 4 evaluates the new features proposed in Chapter 3, through some experiments on a widely used film rating data set.

Chapter 5 concludes this report and discusses possible directions for future work.

1.3 Main Contributions

The main contributions of our project are:

- New classification features that facilitate more successful detection of current profile injection attacks on collaborative filtering. The best feature is called RMAR and is shown to accomplish perfect or near-perfect detection of current profile injection attacks, including the traditionally difficult to detect Average attack. (Chapters 3 and 4)

- A new class of profile injection attack that is more difficult to detect than existing attacks. (Chapter 3)

- A formal treatment of the complexity of actually mounting a profile injection attack. (Chapter 2)

- The collaborative filtering source code written for experiments will imminently be integrated into the UCL PANDA open source information retrieval platform[1]. (Appendix)

[1] More information about this software is available at http://www.adastral.ucl.ac.uk/~junwang/

2 Related Work

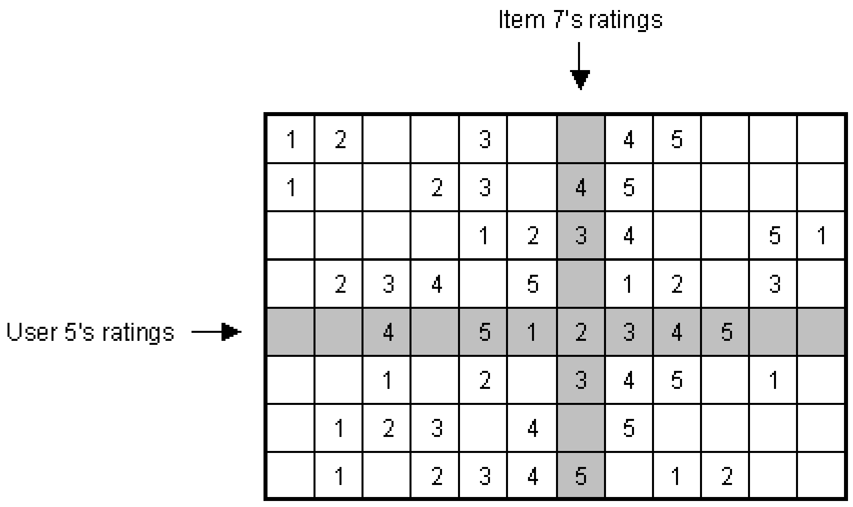
This chapter first gives a detailed introduction to collaborative filtering and profile injection attacks. After this, the current defences against profile injection attacks are presented. Substantially more emphasis is given to the defence presented last, because this report builds on it.

2.1 Collaborative Filtering

Collaborative filtering uses the assumption that similar people like—as well as dislike—the same items. Conceptually, to create personalised recommendations for a given user, a collaborative recommender system will first try to identify a group of other users who have the most similar preferences to the user. Items that are popular amongst this group are then recommended to the user.

Collaborative filtering is widely used and actively researched. Collaborative filtering is implemented by various websites, including but not limited to Amazon, Apple's iTunes Store and—the eBay owned—StumbleUpon.

Two popular collaborative filtering algorithms are now going to be introduced. Both algorithms output a number indicating how likely a given user will be interested in a given item. These will be called the *test user* and *test item* respectively from now on. Both algorithms also make use of a rating matrix **R**: Let $U = \{1, 2, \ldots, m\}$ be the set of users, $I = \{1, 2, \ldots, n\}$ the set of items, and R the set of possible numerical ratings that a user can give to an item. R is depended on the application, so for example it could be [1, 10]. Then $\mathbf{R} = (r_{u,i})_{m \times n}$ where $r_{u,i} \in R$ represents user u has rated item i with $r_{u,i}$ while $r_{u,i} = \varnothing$ means u has not yet rated i (or has chosen not to rate it). Moreover, user u's (user) profile $UP_u = \{(i, r_{u,i}) : r_{u,i} \neq \varnothing\}$ and similarly item i's (item) profile $IP_i = \{(u, r_{u,i}) : r_{u,i} \neq \varnothing\}$. In other words, u's user profile is the set of items that u has rated along with the associated ratings. And i's item profile is the set of users that have rated i along with their respective ratings. So row *u* of **R** roughly corresponds to user *u*’s profile, whereas column *i* of **R** approximately corresponds to item *i*’s profile. Figure 2 below illustrates this.


Item 7's ratings
User 5's ratings

1	2			3			4	5			
1			2	3		4	5				
				1	2	3	4			5	1
	2	3	4		5		1	2		3	
		4		5	1	2	3	4	5		
		1		2		3	4	5		1	
	1	2	3		4		5				
	1		2	3	4	5		1	2		

Figure 2: A fictitious rating matrix with 8 users and 12 items. User 5’s and item 7’s ratings are highlighted. Null ratings are not explicitly shown, and are represented here by blank cells.

In Figure 2, user 5’s profile is {(3, 4), (5, 5), (6, 1), (7, 2), (8, 3), (9, 4), (10, 5)} and item 7’s profile is {(2, 4), (3, 3), (5, 2), (6, 3), (8, 5)}. The size of a user or item profile is defined as the number of ratings it contains (the cardinality), so the sizes of user 5’s and item 7’s profile are 7 and 5 respectively. By definition, the size of *u*’s user profile is equal to the number of items that *u* has rated.

New users initially start with an empty user profile (row). Each user can change the contents of their user profile (by adding item ratings), but obviously not that of another user. In other words, a user has control of only his or her assigned row in **R**. A collaborative filtering algorithm uses a rating matrix to associate a rating prediction to each pair of users and items. This is illustrated below in Figure 3. A rating prediction on item *i* for user *u* is a guess at what rating *u* would have given to *i*. So the job of a collaborative filtering algorithm is essentially to accurately fill in the blanks of a given rating matrix.

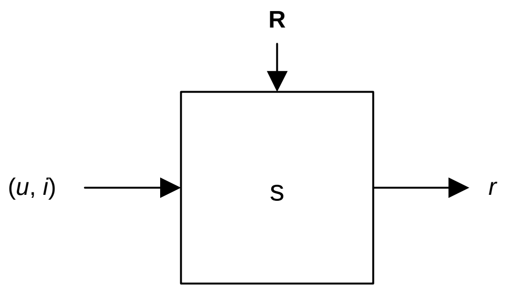


Figure 3: A collaborative filtering algorithm, *s*, which uses rating matrix R. *r* is the rating prediction on item *i* for user *u*. So *r* represents how likely user *u* will be interested in item *i*.

The first collaborative filtering algorithm that will be introduced concentrates on the similarities between users, while the second focuses on item similarity. Both are reasonably accurate and widely used.

2.1.1 User-Based Collaborative Filtering

In user-based collaborative filtering [2], a test item's rating is predicted for a test user using the similarities between their user profile and those of the other users. For example, if Boris and Dave have similar tastes then Boris is likely to be interested in the items that Dave likes, and vice versa. A high-level description of the user-based collaborative filtering procedure follows.

1. Start with all the other users' profiles (i.e. all but the test user's profile).

2. Disregard any user that has not rated the test item, leaving only the user profiles that contain a rating for that item.

3. Ignore any remaining user profile that is considered to be too dissimilar to that of the test user's profile.

4. Combine the ratings for the test item from the left over user profiles to form the required rating prediction, giving extra preference to ratings coming from the more similar users.

Filtering of users occurs in Step 2 and 3, and then a rating is *collaboratively* predicted in Step 4. Similarity between two user profiles in Step 3 can be quantified using Pearson's correlation or cosine similarity [3]. The former is defined as:

$$w_{pearson}(u,v) = \frac{\sum_{j \in J}(r_{u,j} - \bar{r}_u)(r_{v,j} - \bar{r}_v)}{\sqrt{\sum_{j \in J}(r_{u,j} - \bar{r}_u)^2 \sum_{j \in J}(r_{v,j} - \bar{r}_v)^2}}$$

where u, $v \in U$ are two users, $J \subseteq I$ is the set of items that u and v have both rated (i.e. $J = \{i \in I : r_{u,i} \neq \emptyset \text{ and } r_{v,i} \neq \emptyset\}$, and $\bar{r}_w$ is the average of the ratings in w's user profile UP_w. While this is a slightly modified version of the standard Pearson's correlation, because of J, this function's range is still [-1, 1]. Here 1 represents maximum similarity between u and v, while -1 signifies maximum dissimilarity. The similarities between the test user and every other user are computed to determine which need to be filtered out. Two commonly used strategies for this are to either keep: the k users that are most similar to the test user (k nearest neighbour approach); or, all users that have a similarity above a certain threshold (such as 0.1 for Pearson's correlation). For the nearest neighbour approach, $k = 20$ is normally satisfactory.

For Step 4, ratings from the remaining users (from Step 3) can be combined together by taking a weighted average. More precisely:

$$p(u,i) = \bar{r}_u + \frac{\sum_{v \in V} w(u,v)(r_{v,i} - \bar{r_v})}{\sum_{v \in V} |w(u,v)|}$$

where $u \in U$ and $i \in I$ are the test user and test item respectively, and V is the set of remaining users.

2.1.2 Item-based Collaborative Filtering

In item-based collaborative filtering [4], a rating for a test item is predicted for a test user via item profile similarities—instead of user profile similarities, as in the user-based approach. The item-based method is somewhat less obvious than the user-centric method, but the outcome is still the same. The assumption here is that the test user likes and dislikes similar items. For example, if they highly rated the James Bond film Casino Royale then they are likely to enjoy its sequel, Quantum of Solace, as well. The similarity between two items comes from the ratings given to them (i.e. similarity of the two respective item profiles). A high-level description of item-based collaborative filtering follows.

1. Start with all the other items' profiles (i.e. all but the test item's profile).

2. Disregard any item that has not been rated by the test user, leaving only the item profiles that contain a rating from that user.

3. Ignore any remaining item profile that is considered to be too dissimilar to that of the test item's profile.

4. Combine the ratings given by the test user to the left over items to form the required rating prediction, giving extra preference to ratings coming from the more similar items.

Filtering of items occurs in Step 2 and 3, and then a rating is *collaboratively* predicted in Step 4. Similarity between two item profiles in Step 3 can be measured using Pearson's correlation or cosine similarity. The latter has been adapted for the purpose of collaborative filtering, and is defined as:

$$w_{cosine}(i,j) = \frac{\sum\limits_{v \in V} (r_{v,i} - \bar{r}_v)(r_{v,j} - \bar{r}_v)}{\sqrt{\sum\limits_{v \in V} (r_{v,i} - \bar{r}_v)^2} \sqrt{\sum\limits_{v \in V} (r_{v,j} - \bar{r}_v)^2}}$$

where $i, j \in I$ are two items, $V \subseteq U$ is the set of users that have rated both i and j (i.e. $V = \{u \in U : r_{u,i} \neq \emptyset \text{ and } r_{u,j} \neq \emptyset\}$, and $\bar{r}_v$ is the average of the ratings in w's item profile IP_w. This differs from the standard cosine similarity formula in two ways, because of V and the subtraction of the user rating means. The similarities between the test item and every other item are calculated, and dissimilar items can be filtered out using one of the methods described above in the user-based approach (nearest neighbour or threshold).

For Step 4, ratings from the remaining items (from Step 3) can be combined together by taking a weighted average. More precisely:

$$p(u,i) = \frac{\sum\limits_{j \in J} w(i,j) \cdot r_{u,j}}{\sum\limits_{j \in J} w(i,j)}$$

where $u \in U$ and $i \in I$ are the test user and test item respectively, and J is the set of remaining items.

Item-based collaborative filtering is more scalable than the user-based method, because item similarities usually stabilise, rarely fluctuating significantly after a short period of time—so the output of w can be cached for quicker performance.

2.1.3 Why Recommender Systems are Targeted

With the increasingly competitive marketplace that most manufacturers experience, it is in their interest to have their goods frequently and highly recommended. Normally this is achieved by producing quality products with a unique selling point. Thus, companies that make inferior items do not benefit from the increased sales that the rest enjoy. This could lead to dishonest companies attempting to force their goods to be unjustifiably highly recommended. If they are successful in doing this then various parties are affected. Honest manufacturers are affected, because their possibly more suitable products are obscured—so they may see fewer sales. Consumers are also affected, because they would receive biased recommendations. If customers experience this a lot then their trust in the recommender system is likely to diminish. This is also undesirable for the business operating the recommender system, because of the investment made in it. Moreover, dishonest companies would effectively be gaining free advertising from the operator.

2.2 Profile Injection Attacks on Collaborative Filtering

Most collaborative filtering algorithms—including the aforementioned user and item-based ones—assume that the ratings they use to calculate recommendations are unbiased, and an entirely true representation of what all the users think of the items they rated. However, due to the open nature of collaborative filtering, a malicious user can easily give misleading item ratings. This could lead to honest users receiving inaccurate predictions. Moreover, these users would find it difficult to immediately tell if a rating prediction is accurate, because all they know is that some users with similar profiles to them gave a like rating.

It is assumed that the malicious user here wants to affect the rating prediction on a particular *target* item for a subset of *target* users (which could be the whole set of users). The attacker will attempt this by inserting a fake user profile that contains a biased rating for the target item. It is also, of course, assumed that the attacker cannot see the rating matrix used by the collaborative filtering algorithm. Thus, he can only inject new user profiles into the recommender system.

The reader might have already worked out how an attacker can manipulate the user-based collaborative filtering algorithm. But if not, the answer lies in Step 2 and 3 of its high-level description. It should be obvious that the attacker has to make his profile be considered in Step 4, implying he must not be filtered out in Step 2 and 3. Getting passed the first filter is easy: simply provide a rating for the target item. More precisely, max R is given to increase the rating prediction of the target item, while min R is given to decrease it. Negotiating the second filter (Step 3) is much harder though. For this, the attacker has to somehow make his profile look similar to those of the target users. There are various strategies to achieve this, and they will be discussed in a later section. This is all well and good, but the attacker's profile on its own is very unlikely to have a significant effect on the prediction calculation (ratings from honest users should overwhelm the rating from the attacker). To circumvent this problem, the attacker can again exploit the openness of collaborative filtering. It is normally very easy to register as a new user, so the attacker can build multiple profiles associated with fictitious identities. Now it is possible for the attacker to completely influence the rating prediction for the target item—the worst-case scenario. Note that as the number of attack profiles employed by the attacker increases, the probability of at least one of them reaching Step 4 also increases.

Manipulating the item-based collaborative filtering algorithm is not as straightforward. This is because an attacker has to know at least some of the items that each target user has rated (for Step 2), in addition to making the target item's profile similar to that of these items (for Step 3). Recall that an item's profile contains every user that rated it (and the rating each gave). Therefore, a single attack profile can only add—at most—one rating to an item profile. In other words, an attacker does not have much influence on what an item profile looks like. This means that manipulating the similarities between items is relatively hard to do. However certain subsets of users can be targeted in this case, by carefully constructing a number of attack profiles [5].

2.2.1 The Anatomy of a Profile Injection Attack

So as not to distract the reader, a few details were purposefully omitted from the sketch attacks above. These included how to generate artificial user profiles and the number of them required to mount a successful attack. These aspects are the most interesting part of an attack, and receive special treatment in this section.

2.2.1.1 Attack Complexity

The literature does not treat the complexity of mounting a profile injection attack in a formal manner. A rational attacker will want to know the exact cost and feasibility of mounting a successful attack. We adopted the cryptographer Nicolas Courtois' formal notion of security. He says that the security of a system is a triple:

1. Adversarial Goal
2. Resources of the Adversary
3. Access to the System

For example, the security of a certain car could be good with respect to an attacker that wants to (1) steal it, (2) has a toolbox, and (3) has access to the garage it is parked in. However, this may not hold for another triple, e.g. changing (1) to "vandalise it".

It has already been assumed that the adversary's goal is to successfully mount a profile injection attack. With respect to system access, it is also given that the attacker can only insert new user profiles and ratings via legitimate channels, e.g. by clicking a "Sign up for an account" button on a website. Consequently, this attack vector could be guarded with proactive or reactive defences. For example, making users complete a CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) before creating a new account or accepting an item rating may thwart an otherwise successful attack. Also related to system access is how much information about the underlying rating matrix is published, which could be critical to the effectiveness of an attack.

Profile injection attacks require statistics about the rating matrix before they can be executed. Sometimes these statistics can be estimated using an outside source, e.g. the Internet Movie Database website displays the average user vote for each film they have. This is classed as an adversary resource. Another important tool for the attacker is suitable bots (software that automates jobs) to create fictitious users and add item ratings to their profiles quickly. One more equally crucial adversary resource is time. The time to create the necessary number of attack profiles has to be taken into account.

The strongest possible security notion here would be an attacker that wants to (1) alter a target item's rating prediction by an infinitesimal amount, (2) has full knowledge of any security mechanisms employed and a copy of the rating matrix, and (3) has an infinite amount of time to create fictitious user profiles. This triple is, however, hardly practical. A strong, yet realistic triple is an adversary that wants to (1) substantially increase the rating prediction of a target item, (2) can find out only a limited number of statistics about the rating matrix, and (3) has a virtually infinite amount of time to create fictitious user profiles.

Although there are specific attack strategies to decrease the rating prediction of a particular item [6], this report is not going to focus on them. This is because almost no economic advantage is gained by employing such attacks. Although, someone may want to use these attacks, which are said to "nuke" or sink an item, to damage the reputation of a company's product.

2.2.1.2 Attack Terminology

This section introduces some important definitions about profile injection attacks. These definitions are used extensively from here on in.

A *profile injection attack* involves inserting a number of user profiles, corresponding to spoofed identities, into a rating matrix with the intention of increasing the rating prediction of a single item (the target item) for a subset of users (the target users). Here, the number of profiles injected is a percentage of the size of *U*, and is called the *attack ratio*. So an attack with a 10% ratio would increase the size of *U* by 10%. Attacks with ratios ranging from 1% to 15% are the most common in the literature. The attacker uses an attack strategy to build each user profile.

An *attack strategy*[2] is an algorithm for generating a single user profile, called an *attack profile*, for the purpose of profile injection. The overriding objective here is to create profiles that are likely to be considered as similar to the target users. Making profiles that are difficult to detect (i.e. indistinguishable from authentic user profiles) is also important though. This is because if they were not, it would then be very easy to exclude them from the collaborative filtering process—and hence foil the attack.

[2] Attack strategies are also referred to as "attack models" in the literature.

Recall a user profile—and hence an attack profile as well—is a set of item-rating pairs. Attack strategies essentially define how items in an attack profile are chosen and rated. They all do this by first partitioning the set of items I in four: I_T, a singleton set composed of the target item; I_S, a set of *special items* that have certain characteristics; I_F, a set of *filler items* that are randomly chosen from $I \setminus \{I_T \cup I_S\}$; and last but not least $I_\emptyset$, the set of remaining *unrated items*. Next, an appropriate function is applied to each item in I_T, I_S, and I_F to compute respective ratings. These functions are $f_T : I_T \rightarrow \{\max R\}$, $f_S : I_S \rightarrow R$, and $f_F : I_F \rightarrow R$. Two attack profiles built according to the same attack strategy are not necessarily identical, as f_S and f_F are almost always randomised functions. The set of special items, I_S, is seldom non-empty, but if it is then it is normally very small. Filler items are a feature of all attack profiles and represent a predetermined percentage of $|I|-1$. This percentage is called the *filler ratio*, and is normally chosen such that the size of attack profiles is consistent with that of a typical authentic profile (to minimise the chance of detection). Thus, the majority of items in an attack profile are filler items. Attack strategies basically differ only in how they rate filler items (their choice of f_F).

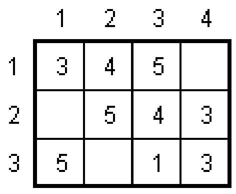
The following pseudo-code outlines a complete generic profile injection attack. The special item set is not included for clarity.

```
$i_T$ := the target item
$I_F$ := {}
filler_size := filler_ratio * (|I| - 1)
while (|$I_F$| < filler_size)
{
    randomly choose $i_F$ from I \ {$I_T$ v $I_F$}
    add $i_F$ to $I_F$
}
attack_size := attack_ratio * |U|
while (attack_size-- > 0)
{
    attack_profile := {}
    add ($i_T$, $f_T$($i_T$)) to attack_profile
    for each $i_F$ in $I_F$
    {
        add ($i_F$, $f_F$($i_F$)) to attack_profile
    }
    insert attack_profile into rating matrix
}
```

Some well-studied attack strategies from the literature will now be introduced. Each attack strategy is formally defined, with definitions originating from [6] but adapted where necessary to integrate with the notation introduced in this section.

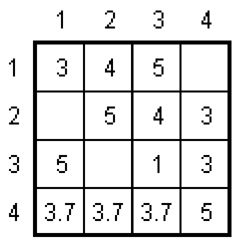
2.2.1.3 Random Attack

The original Random attack strategy [7] was one of the first published attacks against collaborative filtering. The *Random attack strategy* simply involves rating each filler item around the average rating across all users/items. More formally, $f_F(i)$ outputs a random Gaussian distributed value with mean $\bar{r}$ and standard deviation s, where $\bar{r}$ is the arithmetic mean of the non-null elements of the rating matrix **R**, and s is the standard deviation of the same set of elements. For example, suppose the rating matrix has three users and four items and looks like



	1	2	3	4
1	3	4	5	
2		5	4	3
3	5		1	3

then $\bar{r} = (3+4+5+5+4+3+5+1+3)/9 \approx 3.7$. If item 4 is the target and 5 is the maximum rating, a Random attack profile with a 100% filler ratio could be {(1, 3.7), (2, 3.7), (3, 3.7), (4, 5)}. Injecting this into the rating matrix would result in:



	1	2	3	4
1	3	4	5	
2		5	4	3
3	5		1	3
4	3.7	3.7	3.7	5

The Random attack has a relatively low attack complexity (with respect to rating matrix access at least). If the average rating and spread are not publicly available, they can normally be accurately guessed.

2.2.1.4 Average Attack

The original Average attack strategy was introduced at the same time as the Random attack, in [7]. The *Average attack strategy* says that each filler item is rated around the average rating given by all users for *that* item. More precisely, $f_F(i)$ outputs a random Gaussian distributed value with mean $\bar{r}_i$ and standard deviation s_i, where $\bar{r}_i$ is the arithmetic mean of the ratings in i's item profile IP_i, and s_i is the standard deviation of the same set of ratings. Using the original rating matrix from the previous example, $\bar{r}_1 = (3+5)/2 = 4$, $\bar{r}_2 = (4+5)/2 = 4.5$, and

$\bar{r}_3 = (5+4+1)/3 \approx 3.3$. So an Average attack profile targeting item 4 could be {(1, 4), (2, 4.5), (3, 3.3), (4, 5)}.

The Average attack clearly has a considerably higher attack complexity than the Random one (the average rating of each individual filler item has to be known). However, some websites, including Amazon, display the average rating given to items that they offer. But if not, average ratings can be estimated by, again, guessing or using an independent source. It can also be imagined that the adversary is cooperating with an insider who can view the rating matrix (insider attack).

The Average attack has been shown to be the strongest and most effective attack strategy [8], offering the greatest chance of effecting a large rating prediction increase. It is easy to see why this is true; Average attack profiles are statistically likely to be considered similar to those of most genuine users. However, due to this fact alone, the Average attack has so far been the hardest to detect. This has resulted in the attack receiving a lot of attention from researchers.

2.2.1.5 Bandwagon Attack

The Average attack clearly has a high attack complexity. The Bandwagon attack strategy was proposed in [9] as a response to this, and is almost as effective as the Average attack. The *Bandwagon attack strategy* is defined in exactly the same way as the Random attack, except that the special items set is non-empty. I_S contains a predetermined number of the most frequently rated items. And $f_S(i) = \max R$.

The Bandwagon attack is an extension of the Random one. It has an attack complexity somewhere between the Random and Average attacks. Popular items can usually be identified using publicly available information. For instance, any book on the Richard & Judy show’s incredibly successful Book Club reading list enjoys bestseller status.

2.3 Current Defences against Profile Injection Attacks

At present, there are essentially two main methods of countering profile injection attacks. One is implementing robust collaborative filtering algorithms, which are not supposed to be unduly affected by the presence of random noise or injected attack profiles. And the other method is deploying an attack profile detection mechanism in front of an existing recommender system, so that biased profiles from an attacker (hopefully) do not enter, and its collaborative filtering algorithm only sees authentic user profiles. Neither defence technique is completely impervious to attack profiles, with each currently having varying degrees of success. An overview of both follows.

2.3.1 Robust Collaborative Filtering

Some collaborative filtering algorithms have recently been designed from the ground up to withstand profile injection attacks. Dimension reducing techniques such as singular value decomposition (SVD) and principal component analysis (PCA) have been used to achieve this. The idea behind using dimension reduction is that attack profiles in a rating matrix tend to add very little information.

The state of the art robust collaborative filtering algorithms [10] offer satisfactory resistance to Random attacks. However, Average attacks are still very effective against them. Further information about robust collaborative filtering algorithms can be found in [10].

2.3.2 Attack Profile Detection

Genuine banknotes possess several observable features to help us distinguish them from counterfeit ones. These include unique feeling paper, raised print, a metallic thread, a watermark, high quality printing, a hologram, a ultra-violet feature, and microlettering. All of these features can be found on any genuine £10 note and are shown, as ordered, below in Figure 4.

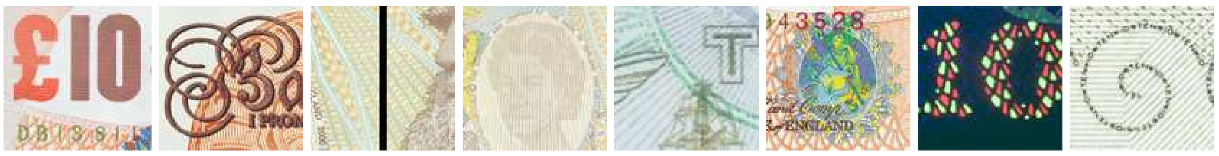


Figure 4: Images of the (disclosed) security features of a £10 note, courtesy of the Bank of England.

A decent counterfeit note will have, to some degree, most of these features (but certainly the most obvious ones). Thus, a perfect counterfeit note will replicate each and every feature flawlessly, so that even an expert cannot tell it apart from a genuine banknote. Conversely, a poor counterfeit note would be recognised by any member of the general public. This could be because of the absence of one or more features or a badly imitated feature. A trained expert will not rely on the presence of just one feature to recognise a genuine note, because a counterfeiter may have managed to perfectly replicate it. Therefore the expert will examine each and every feature he is aware of.

Like counterfeit banknotes, the vast majority of attack profiles lack certain features that genuine user profiles normally possess—or have peculiar features that are not common amongst genuine user profiles. However, the situation here is slightly more complicated, because a user with unusual tastes, which dramatically depart from those of the majority, may inadvertently have the characteristics of an attack profile.

The problem of defending against profile injection attacks can be reduced to designing a classifier to detect attack profiles (similar to the banknote expert above). In particular, features that readily distinguish attack profiles from genuine profiles have to be identified. Given a user profile, a classifier will say whether it belongs to an attacker or not. Detected attack profiles can then be discarded.

2.3.2.1 The Advantages of Attack Profile Detection

Attack profile detection is in a sense analogous to detecting email spam. Email messages that are classed as spam can be filtered into a designated folder, for closer inspection, or simply trashed. Ideally, spam detection should be done before messages are delivered to email clients, so that they receive messages labelled as spam or no spam at all. The separation of spam detector and email client yields numerous benefits. Firstly, each is able to concentrate on their primary objective, which is accurately classifying messages and managing messages respectively, and do it to the best of its ability. This is consistent with the UNIX philosophy of doing one thing well. Another advantage is that any email client can benefit from the specialised spam detection, so they do not have to be overly concerned with spam.

Likewise, using attack profile detection is beneficial, because it allows the use of any collaborative filtering algorithm (robust or otherwise). Suspected attack profiles can be discarded, instead of being inserted into the rating matrix, so that the collaborative filtering algorithm does not have to assume the presence of attack profiles. As a result, collaborative filtering researchers can concentrate on the problem of producing highly accurate rating predictions. On the attack profile detection side, there are also numerous benefits. If the detector is extensible, new features can be plugged in to improve detection of existing attacks or to facilitate the detection of new attacks—akin to antivirus signature updates. Due to the fact that user profiles are inspected individually, one at a time, this approach is indifferent to the number of profiles an attacker injects (attack size), and even the number of adversaries attempting to manipulate the recommender system. The latter is particularly useful, because in reality it is highly likely that at any one time there will be multiple attackers—each using different attack strategies, and targeting different items. Such simultaneous attacks are rarely considered in robust collaborative filtering literature (their experiments only include lone attacks). So simultaneous attacks could ultimately be the Achilles' heel of robust collaborative filtering algorithms.

An overview of a selection of current attack profile features from the literature follows. It should be noted that none are perfect at detecting all attacks.

2.3.2.2 RDMA: Rating Deviation from Mean Agreement

In 2005, Chirita et al. introduced a feature called RDMA in [11]. RDMA is defined as:

$$RDMA(u) = \frac{1}{|UP_u|} \sum_{(i,r) \in UP_u} \frac{|r - \bar{r_i}|}{|IP_i|}$$

where $u \in U$, UP_u is u's user profile, IP_i is item i's profile, and $\bar{r}_i$ is the arithmetic mean of the ratings in i's item profile.

RDMA measures how much a given user's ratings depart from those of the other users. It does this by inspecting each of the items that the user has rated, taking into account the difference between the rating given to them and their average rating. The formula also considers how many other users have given a rating to each of these items. This is because items with very few ratings are more susceptible to profile injection attacks (an attacker's ratings for such items can quickly become authoritative). Thus an attack profile is expected to have a relatively large RDMA value.

2.3.2.3 WDA: Weighted Degree of Agreement

In 2006, Williams et al. proposed a feature called WDA, which is derived from RDMA, in [12]. WDA is defined as:

$$WDA(u) = |UP_u| \cdot RDMA(u) = \sum_{(i,r) \in UP_u} \frac{|r - \bar{r_i}|}{|IP_i|}$$

where $u \in U$, UP_u is u's user profile, IP_i is item i's profile, and $\bar{r}_i$ is the arithmetic mean of the ratings in i's item profile. WDA is precisely the summation component of RDMA.

2.3.2.4 WDMA: Weighted Deviation from Mean Agreement

In [12] Williams et al. also introduced another RDMA derivative, called WDMA and is defined as:

$$WDMA(u) = \frac{1}{|UP_u|} \sum_{(i,r) \in UP_u} \frac{|r - \bar{r}_i|}{|IP_i|^2}$$

where $u \in U$, UP_u is u's user profile, IP_i is item i's profile, and $\bar{r}_i$ is the arithmetic mean of the ratings in i's item profile. WDMA is identical to RDMA except that the denominator inside the summation is squared. As a result, WDMA places more emphasis on rated items with fewer ratings from other users.

2.3.2.5 DegSim: Degree of Similarity with Top Neighbours

In [11] Chirita et al. also proposed a feature called DegSim, which is defined as:

$$DegSim(u) = \frac{1}{|V|} \sum_{v \in V} w_{pearson}(u, v)$$

where $u \in U$, and V is a set of a pre-specified number of the most similar users to u according to $w_{pearson}$, the user similarity measure defined in Chapter 2. DegSim is the average similarity between a given user and its $|V|$ nearest neighbours. The reasoning behind this feature is that attack profiles—by design—exhibit an unusually high amount of similarity between genuine users. So attack profiles are expected to have a high DegSim value.

2.3.2.6 LengthVar: Length Variance

In 2006, Burke et al. introduced the LengthVar feature in [14], which is defined as:

$$LengthVar(u) = \frac{|UP_u| - \bar{l}}{\sum_{v \in U} (|UP_v| - \bar{l})^2}$$

where $u \in U$, UP_w is w's user profile, and $\bar{l}$ is the average user profile size for the users in U. The LengthVar feature measures how much a given user's profile size deviates from that of the other users. Assuming there are a large number of items, it is likely that a genuine user would (and could) only rate a small proportion of these. This is in contrast to an attacker equipped with an automated means of adding item ratings, which can rate a large number of items in a short period of time. Thus, attack profiles with a significantly large filler ratio will have a noticeably high LengthVar value.

The LengthVar feature was designed to target attack profiles containing an exceptionally large number of filler items. It does a good job of this, but LengthVar is of limited use however. This is the case because large attack profiles are not common (higher attack complexity), and not as effective as attack profiles that are around the same size of genuine user profiles anyway. Moreover, user profiles belonging to new users are almost always disproportionately small initially. So until a new user has rated enough items, they are likely to be considered as an attacker by LengthVar, which may lead to the possibility of a false alarm.

3 Novel Features for Detecting Attack Profiles

This chapter proposes three original features for the purpose of user profile classification. The new features are the crème de la crème of the ones that were conceived by us. Each looks at unique aspects of attack profiles that have yet to be studied in the literature. Moreover, they are applicable to all current attack strategies.

In our opinion, existing features work at the wrong abstraction levels to reliably differentiate attack and genuine user profiles. For instance, Average attack profiles are mostly composed of ratings such that when looked at individually, it is impossible to accurately say whether they originated from a genuine user or an adversary. This is probably the main reason why the Average attack has been so good at evading detection. To tackle this problem, we did what clever people do when they want to solve a traditionally difficult problem: avoid it. To be more precise, we approached the problem from a different angle.

Instead of looking one at a time at each rating that a particular user has given, we looked at them as a whole and concentrated on *what* items they rated, not *how* they rated them. This dramatically departs from the approach taken by most other people. Two of the features that are proposed (RMAR and RIC) take advantage of this novel approach and yield statistically perfect profile classification, accurately detecting all current attacks in experiments. In the light of these results, we have modified the current attack strategies to create a new class of attack that is more resistant to detection.

3.1 RMAR: Ratings Missing At Random

For [15] some users of Yahoo’s Internet radio service, LAUNCHcast, were asked about how they rate the songs it plays. In one question they were asked how often they would rate a song given their preference for it. The results from this user study (summarised below in Table 1) show that these two aspects are related, such that LAUNCHcast users are much more likely to rate songs that they love than ones they hate.

	Rating Frequency		
Preference Level	**Never**	**Infrequently**	**Often**
Hate	6.76 %	3.22 %	90.02 %
Do Not Like	4.69 %	8.61 %	86.70 %
Neutral	2.33 %	34.33 %	63.33 %
Like	0.11 %	2.02 %	97.87 %
Love	0.07 %	0.55 %	99.37 %

Table 1: <u>Reproduced</u> results of a LAUNCHcast user survey [15], where participants were asked how often they would rate a song given their preference for it.

The frequency distribution of ratings in a well-known film and book data set are consistent with this user study. These are illustrated below in Figure 5. Note the bias towards higher ratings in both cases. The film data set is actually used in our experiments.

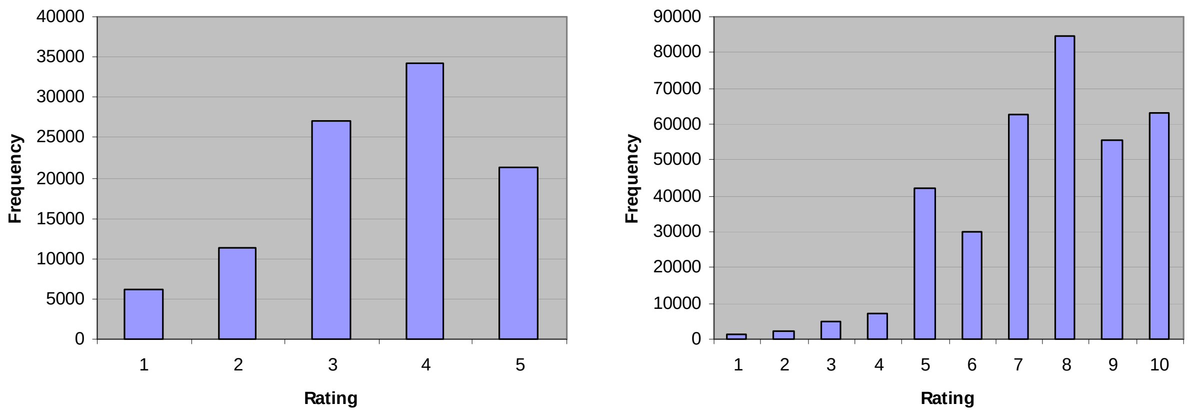


Figure 5: Graphs of the frequency distribution of ratings in a particular MovieLens (left) and the Book-Crossing (right) data set. Greater ratings indicate more liking for a film/book.

Assuming these results hold for all rating data sets used for collaborative filtering, this implies that genuine users are more likely to provide a rating for items that they like than ones that they do not. More precisely, for the cognoscenti, most absent ratings for genuine users should *not* be missing at random (MAR). Conversely, attack profiles—since they are always mainly composed of randomly chosen (filler) items—will by definition have a lot of absent ratings that are MAR. Thus the presence of a large proportion of MAR ratings is a feature of attack profiles, but not of genuine user profiles. While many researchers have proposed features based on the randomness of item ratings to detect attack profiles, none have exploited the randomness of chosen filler items.

We needed a way to accurately measure this feature, which is now called Ratings Missing At Random (RMAR). Notice that a pair of two distinct items chosen at random is unlikely to have similar item profiles —especially when the population of items is large. Consequently, each pair of rated items in an attack profile is not likely to be similar, whereas the opposite of this is more probable for genuine user profiles. With a set of user profiles known to belong to genuine users, item similarity can be quantified using the adjusted cosine similarity measure, w_{cosine} (from the item-based collaborative filtering algorithm, which is defined in Chapter 2). Recall that for two items, w_{cosine} gives the strength of similarity between them, with 1 and –1 represent absolute similarity and dissimilarly respectively. As a result the function *RMAR*: $I^n \rightarrow [-1, 1]$ is defined as:

$$RMAR(i_1, i_2, \ldots, i_n) = \frac{-1}{\binom{n}{2}} \sum_{j=1}^{n} \sum_{k=j+1}^{n} w_{cosine}(i_j, i_k)$$

where i_1, i_2, …, i_n are the items in a user profile, and w_{cosine} is a function defined in Chapter 2. So, here the similarity of each unique pair of items rated by a user is calculated, summed together, and then normalised. This result is negated so that a positive value signifies a high proportion of ratings missing at random present, while a negative value indicates a low amount. Therefore we would expect attack profiles to have a positive RMAR, whereas genuine user profiles have a negative RMAR. Moreover, the magnitude of the result represents the amount of confidence we have in either result. Ideally, the outputs of w_{cosine} should be cached. This makes real-time attack detection much more feasible. The outputs can be cached, because they do not usually fluctuate after items have received a sufficient number of ratings from users.

The key assumption with RMAR is that item similarities (the outputs of w_{cosine}) are reasonably accurate. This in turn means that the item profiles that w_{cosine} uses are assumed to be mature and stable. This is actually a fair assumption, and is precisely the one that the item-based collaborative filtering algorithm makes. It is also assumed that genuine users rate items in a non-random, reasoned fashion.

In theory, as the number of randomly chosen filler items in an attack profile increases, the more likely it is that it will have a high RMAR, and therefore be more noticeable. This is very advantageous, because the attack complexity now increases as the number of filler items in an attack profile grows. This is very pertinent, because increasing the size of an attack profile is traditionally not seen to be expensive to an attacker (they could simply rate another randomly chosen item). Because of RMAR, filler items now have to be chosen more intelligently, in a non-random fashion. Another major advantage of the RMAR feature is that it is applicable to all current attack strategies. This is because each attack makes use of randomly chosen filler items that make up the bulk of an attack profile.

There is however three possible scenarios where RMAR may come unstuck, unable to clearly differentiate between genuine users and attackers. One is when any user profile, genuine or otherwise, contains very few rated items. This is because such profiles may have a disproportionate RMAR value, since there is little evidence to go on. Likewise, genuine users that are unconventional in the way that they rate items may have a RMAR value similar to that of attack profiles. This could be the case because such users deviate from the norm too much, seemingly choosing items to rate randomly—what RMAR uses to separate attack profiles from genuine ones. Finally, RMAR may be ineffective against Bandwagon attacks with a very large number of special items, because these frequently rated items could be similar to each other.

3.1.1 RMAR-Resistant Attacks

For an attack profile to receive a favourable RMAR value, the adversary would need to gather knowledge of potential filler items that are likely to be similar to each other—on top of whatever other information their strategy requires. The new RMAR feature uses the fact that all current attack strategies select filler items randomly. More precisely, each item distinct from the target one and not already a special or filler item has an equal probability of being chosen. Therefore, it would be interesting to see if attack strategies that choose filler items based on popularity can go undetected. This is exactly what is done in our experiments.

Existing attack strategies (including the Random, Average, and Bandwagon ones) can be modified to choose filler items proportionately to their popularity. Popular items are used in the hope that RMAR will think these items are similar. More formally, an item *i* is selected to be a filler item with a probability of $\left|IP_i\right| \big/ \sum_{j=1}^{|I|} \left|IP_j\right|$. Note that unpopular items still have a chance of being selected. In practice a roulette wheel, where probabilities are proportional to item popularity, is used to choose filler items. An attack strategy that chooses filler items like this is going to be said to use “popular filler items”, e.g. Average attack with popular filler items.

All current attacks use filler items that are simply chosen at random. Because this new attack class requires the additional knowledge of which items are popular, it has the highest attack complexity of all (on paper at least). However, this information may be available publicly. It may even be available from the operator of the target recommender system. For example, the Amazon website conveniently displays its 100 bestsellers in any category (e.g. books)—and updates them hourly. Other information sources include the likes of the UK singles and album charts.

3.2 RIC: Rated Items Consistency

The RMAR feature does not consider the item ratings in a user profile—just the items themselves. Consider the case when RMAR encounters a pair of items that are totally dissimilar. If the user happened to rate one of these items with the smallest possible rating and the other with the largest allowable rating, he would still be penalised. RMAR has been extended to take into account the consistency of item ratings (rated items consistency). The function *RIC*: $I^n \times R^n \rightarrow [-1, 1]$ is defined as:

$$RIC(i_1, i_2, \ldots, i_n, r_1, r_2, \ldots, r_n) = \frac{1}{\binom{n}{2}} \sum_{j=1}^{n} \sum_{k=j+1}^{n} w_{cosine}(i_j, i_k) \cdot \frac{(\max R - |r_j - r_k|)}{\max R}$$

where i_1, i_2, …, i_n are the items in a user profile, and r_1, r_2, …, r_n are the corresponding ratings. RIC is identical to RMAR except that it scales w_{cosine}, and does not negate the double summation so that higher RIC values indicate more item consistency. Thus, only genuine users are expected to receive a positive RIC value.

3.3 MaxRatings: Maximum Ratings

It has been noticed in experiments that attack profiles tend to have a very low proportion of items rated with the maximum possible rating (or a rating very close to the maximum). This could be because the majority of items in an attack profile are either rated: around their respective average ratings (in the case of the Average attack); or, the overall average rating (for the Random and Bandwagon attacks). So an attacker’s ratings rarely stray too far from the norm, which is unlikely to be close to the maximum allowable rating. Only the target item is given an extreme rating by an attacker. Contrast this with genuine users, who reserve the maximum possible rating for items they love (and the minimum rating for items they hate).

Because people normally have several favourite items, a genuine user's profile is likely to contain a number of maximum ratings. The maximum allowable rating also has a psychological meaning to people, given to items that completely meet their needs. Thus, it is expected that genuine user profiles contain more maximum ratings than those of attackers. Consequently the function *MaxRatings*: $U \rightarrow [0, 1]$ is defined as:

$$MaxRatings(u) = \frac{1}{|UP_u|} \sum_{(i,r) \in UP_u} \begin{cases} 1 & \text{if } \max R - \delta \leq r \leq \max R \\ 0 & \text{otherwise} \end{cases}$$

where UP_u is *u*'s user profile, and δ is a small positive number.

The dual to MaxRatings is Minimum Ratings (MinRatings), which looks at the proportion of user ratings close to min *R*. In preliminary tests MinRatings performed very poorly, which was quite surprising as the above reasoning for MaxRatings should logically be valid for minimum ratings too. However, on closer inspection of the profiles of some genuine users, it was found that they usually only contain a small proportion of items rated with the minimum allowable rating. This is actually consistent with the results from the LAUNCHcast user survey above.

4 Experiments

This chapter presents the results from experiments that investigated the properties of the new classification features proposed in Chapter 3. In particular, we determined the new features are better than existing ones at detecting attacks, and established when the new features work well (and not so well). We experimented on a commonly used collection of ratings.

4.1 The Data Set

The MovieLens data set consisting of 100,000 ratings[3] has been more or less adopted as the standard for collaborative filtering experiments. A group within the University of Minnesota compiled the data set. It contains 100,000 ratings from 943 users on 1682 films (items). Each user has rated 20 or more films, and each rating is an integer ranging from 1 to 5, inclusive. Greater ratings indicate more liking for a film. The mean rating is 3.5 (1 dp) and the ratings have a standard deviation of 1.1 (1 dp). The average user profile size is 106.0 (1 dp) with a standard deviation of 100.9 (1 dp). Additionally, the median profile size is 65. Therefore, to avoid being too overt, attack profiles should be around these sizes, which correspond approximately to filler ratios of 6% and 3% respectively.

We assumed that the MovieLens data set does not contain any ratings from an adversary. Hence all the 943 users were assumed to have provided completely honest item ratings.

4.2 Results

What follows are the results from a selection of experiments.

[3] At the time of writing, this MovieLens data set is available at http://www.grouplens.org/node/73

4.2.1 Detection of Average Attacks using New Features

All the new features proposed in Chapter 3 were individually evaluated, and contrasted with the state-of-the-art features defined in Chapter 2. To do this, the user profiles from the MovieLens data set were randomly split into two halves. The first half was designated as the training set, and the other was the test set (of genuine user profiles). The training set was used by the classifiers, and as a basis for statistics to create test attack profiles. For a particular attack and filler ratio, an attack profile was generated and then added to the test set. More attack profiles were added until there was exactly the same number of attack profiles as there were genuine ones. So the size of the final test set were twice that of the original test set. Each attack profile was created with a fresh set of filler items and target item. As discussed in Chapter 2, note that the attack ratio is irrelevant here. The test set was used only to assess the accuracy of a classifier, and was unseen by the classifier whilst being trained.

To evaluate the individual features, a simple binary classifier that uses a threshold to discriminate between attack profiles and genuine ones was used. The performance of this classifier—and hence the features—was measured via receiver operating characteristic (ROC) analysis [16]. ROC analysis originates from signal detection theory, and is used in many fields including machine learning. The principal tool is the ROC curve, which is a graph of the fraction of hits (the hit rate) and fraction of false alarms (the false alarm rate) made by a binary classifier as its threshold is varied. The false alarm rate is on the *x*-axis of the graph, while the hit rate is on the *y*-axis. The terms hit, correct rejection, false alarm, and miss in this context are defined below:

- A *hit* is said to occur here if the classifier says an actual attack profile belongs to an adversary. This is also called a true positive (TP).
- A *correct rejection* happens if the classifier says an actual genuine user profile belongs to a real user. This is also called a true negative (TN).
- A *false alarm* happens if the classifier says an actual genuine user profile belongs to an attacker. This is also called a false positive (FP).
- A *miss* occurs if the classifier says an actual attack profile belongs to a genuine user. This is also called a false negative (FN).

We used ROC curves, because they intuitively visualise the trade-off between different hit and false alarm rates. While ROC curves are very useful for visualising and tuning the performance of classifiers, this is subjective as it is difficult to accurately compare curves. Fortunately, several statistics can be derived from a single ROC curve. One is the area under the ROC curve (AUC) statistic, which is what the machine learning community tends to prefer. The AUC statistic will thus always assume a value in [0, 1]. Higher values indicate better accuracy here, with 1 signifying perfect detection. An AUC less than or equal to 0.5 means that using the classifier in question is worse than just randomly guessing the classification of a user profile. Suppose that attack profiles normally have a high value for a certain feature. The AUC has a nice property in that it is equal to the probability that a random attack profile is given a higher feature value than a random genuine user profile. The AUC was computed using this established formula:

$$AUC = \frac{G+1}{2}$$

where *G* is the Gini coefficient:

$$G = 1 - \sum_{k=1}^{n} (X_k + X_{k-1})(Y_k - Y_{k-1})$$

where each (X_k, Y_{k-1}) is a point on a ROC curve such that the sequence (X_k) is monotonically increasing. So X_k is a false alarm rate and Y_k is its corresponding hit rate value. In effect, the AUC is approximated via the trapezium rule (a method to estimate a definite integral) and the penultimate equation.

The opening experiment looks at how the features fare against the Average attack, which has so far been the hardest to detect accurately. A test set was generated as per the stated method above using a 3% filler ratio. This filler ratio was found to yield the most effective attacks, which is consistent with other work, e.g. [6]. Three is also the median percentage of items that users in the training and test set rated, so the LengthVar feature was not be called upon. Figure 6 below shows the resultant ROC curves for each feature.

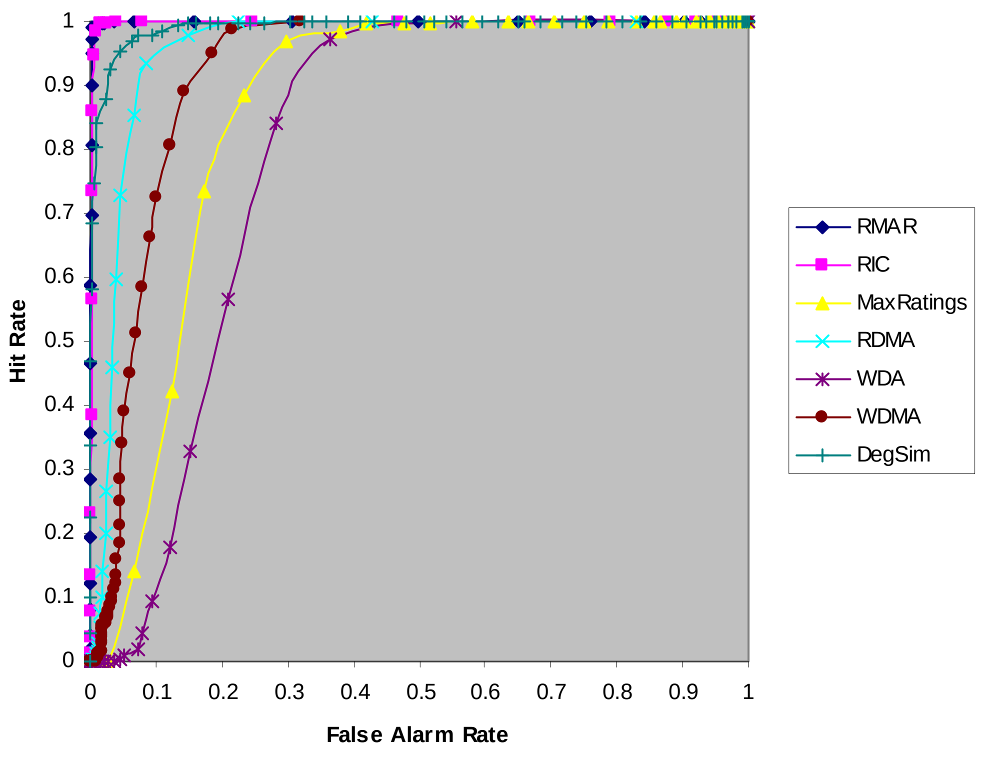


Figure 6: ROC curves indicating how well each feature detects some Average attack profiles with a 3% filler ratio. Corresponding AUC statistics (3 dp): RMAR, 0.999; RIC, 0.998; MaxRatings (δ = 0.25), 0.855; RDMA, 0.957; WDA, 0.797; WDMA, 0.919; DegSim (100 nearest neighbours), 0.991.

From Figure 6 it is clear to see that RMAR and RIC, while not perfect, are excellent features and outperform the ones proposed by others. So for this particular test set, RMAR would correctly rank a random attack profile and genuine profile with a 99.9 percent chance. DegSim is a reasonable feature here. This is to be expected, as Average attack profiles are constructed such that they are considered similar to as many users as possible. The MaxRatings feature is satisfactory, but certainly not outstanding.

To see if RMAR and RIC are actually better than the current features at detecting the Average attack, a significance test was carried out. A paired *t*-test was used, with a significance level of 0.01 (1%). Ten fresh, randomly generated test sets were used in this significance test. Table 2 summaries the results from this experiment.

	Filler Ratio			
Feature	**1%**	**3%**	**6%**	**10%**
RMAR	0.996	1.000	1.000	1.000
RIC	0.994	0.999	0.999	0.999
MaxRatings	0.795	0.887	0.910	0.932
RDMA	0.925	0.955	0.959	0.966
WDA	0.455	0.771	0.910	0.970
WDMA	0.871	0.915	0.921	0.934
DegSim	0.943	0.986	0.930	0.746

(a) Mean AUC statistics (3 dp)

	Filler Ratio			
Feature v Feature	**1%**	**3%**	**6%**	**10%**
RMAR v RDMA	**0.000**	**0.000**	**0.000**	**0.000**
RMAR v WDA	**0.000**	**0.000**	**0.000**	**0.000**
RMAR v WDMA	**0.000**	**0.000**	**0.000**	**0.000**
RMAR v DegSim	**0.000**	**0.000**	**0.000**	**0.000**
RIC v RDMA	**0.000**	**0.000**	**0.000**	**0.000**
RIC v WDA	**0.000**	**0.000**	**0.000**	**0.000**
RIC v WDMA	**0.000**	**0.000**	**0.000**	**0.000**
RIC v DegSim	**0.000**	**0.000**	**0.000**	**0.000**
RMAR v RIC	**0.001**	**0.002**	**0.010**	**0.004**

(b) Associated *p*-values (3 dp). *p*-values less than or equal to 0.01 are in bold (all in this case).

Table 2: Results of the features for 10 different Average attack test sets for each filler ratio. MaxRatings had δ = 0.25. And DegSim used 100 nearest neighbours.

Table 2 shows that RMAR achieves statistically perfect detection of Average attack profiles with filler ratios greater than or equal to 3%. Additionally, RMAR and RIC are indeed significantly better at distinguishing Average attackers from genuine users than the current features. This is true for all the listed filler ratios, which are the most popular ones used in the literature. Moreover, the performance of RMAR and RIC is relatively consistent over the filler ratios, especially when compared to the other features. MaxRatings, RDMA, WDA and WDMA appear to get progressively better as the filler ratio increases. For higher filler ratios, even up to 100%, RMAR exhibited comparable results. This was predicted to happen in Chapter 3. For the sake of brevity, these results are omitted.

Table 2 also reveals a rather surprising result: RMAR is marginally better than RIC. Recall that RIC is an extension of RMAR, and uses some rating data while the latter does not at all. This could suggest that, at least from the point of attack detection, rating values are not as important as the actual items that have been rated by a particular user.

The DegSim results in Table 2 indicate when the Average attack is most effective. The performance of DegSim peaked at the 3% filler ratio. This could mean that a filler ratio around the median proportion of items rated by genuine users (3% for the data set under consideration) is optimal.

4.2.1.1 Detection of Small Attack Profiles using RMAR

The RMAR feature measures the degree of similarity between items within a single user profile. Thus, it would be very interesting to see how the feature is affected by particularly small user profiles. It was hypothesised in Chapter 3 that RMAR might suffer in this situation, leading to genuine users who have not rated many items being treated as an attacker. This was because the rated items in such user profiles may inadvertently look like random selections. This is a very important aspect, because all genuine users' profiles start empty and will likely still be considered as small during their infancy, until enough ratings have been provided.

Now, as previously mentioned, the users in the MovieLens data set have all rated at least 20 items (out of the possible 1682). This is reflected by 1% being the lowest filler ratio in the previous experiment. A 1% filler ratio in this context corresponds to attack profiles of at least size 17 (16 filler items + 1 target item). So, it would not be fair to immediately use a test set that has attack profiles with a filler ratio smaller than 1%, for obvious reasons. Fortunately, the MovieLens data set includes timestamps, to the second, denoting when a user submitted a rating for an item. Thus user profiles from this data set can be shortened, by removing the most recently added ratings. The resultant user profiles can and should still be considered as genuine, because they are precisely what the (assumed genuine) original user profiles once looked like at a certain time. Moreover, the timestamps provide sufficient granularity for accuracy.

To the best of our knowledge, filler ratios less than 1% have not been investigated so far in the literature. Here, filler ratios of 0.8%, 0.6%, 0.4% and 0.2% will be examined. To put these into perspective, the aforementioned filler ratios correspond to profile sizes of 13, 10, 7 and 4 respectively for the MovieLens data set, which has ratings on 1682 items. Figure 7 below shows the ROC curves from some typical Average attack test sets.

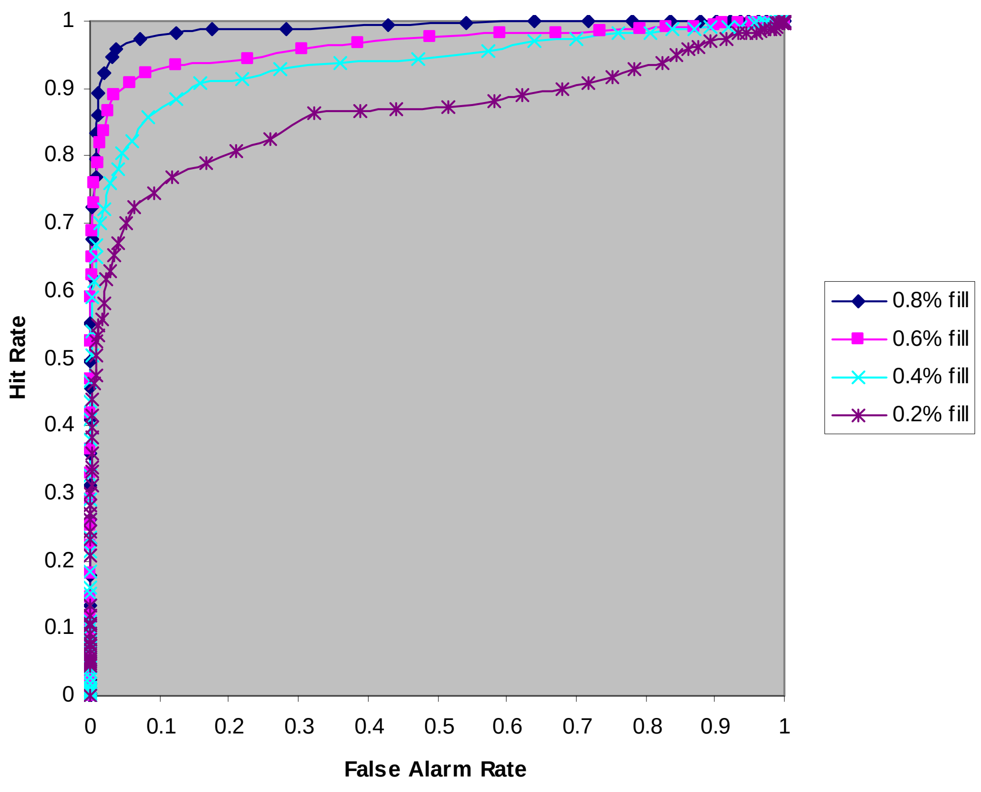


Figure 7: ROC curves indicating how well the RMAR feature detects some typical Average attack profiles with small filler ratios. Corresponding AUC statistics (3 dp): 0.8% fill, 0.989; 0.6% fill, 0.963; 0.4% fill, 0.936; 0.2% fill, 0.861.

The ROC curves in Figure 7 show that filler ratios less than 1% do affect RMAR. However, the affect is certainly not adverse. Even with genuine users and Average attackers who have only rated four items, the performance of RMAR is arguably still respectable. Very similar results were encountered with small Random attacks, which is not surprising as RMAR does not take into account item ratings.

The results suggest that, for this data set at least, most genuine users choose which items to rate in a non-random fashion from the outset. We can also conclude that RMAR handles very small attack and genuine user profiles reasonably well. Nevertheless, RMAR can be modified to say that all profiles containing less than a certain number of ratings are genuine. This would eliminate the possibility of false alarms for new users.

4.2.2 Detection of Random and Bandwagon attacks using New Features

We will now look at the Random and Bandwagon attacks. Given the results from the previous Average attack experiment, in theory RMAR should exhibit almost perfect detection performance for the Random attack. This is because the feature does not measure anything specific to the Average or Random attacks (recall these attacks are identical apart from how they rate filler items). A 3% filler ratio was used again. Figure 8 shows the ROC curves from a Random and a Bandwagon attack test set.

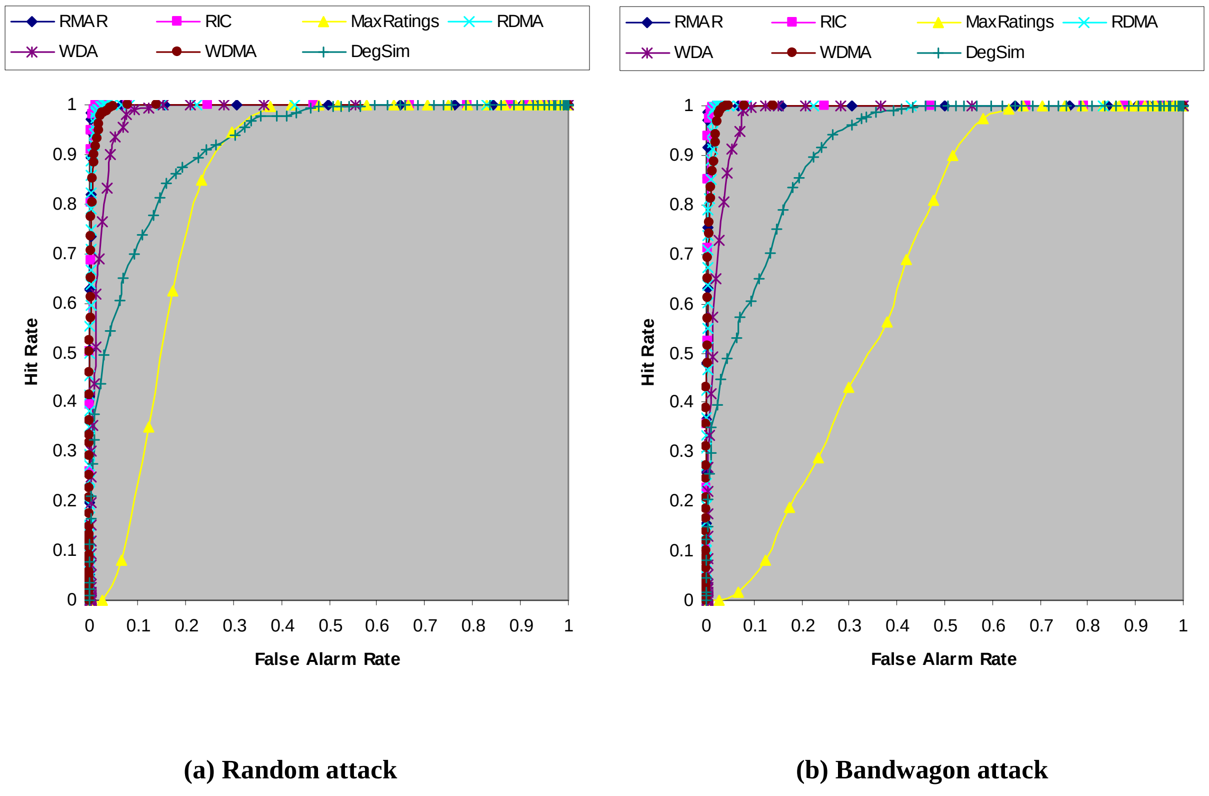


Figure 8: ROC curves indicating how well each feature detects some Random and Bandwagon attack profiles with a 3% filler ratio. Corresponding AUC statistics (3 dp): RMAR, (a) 0.999, (b) 0.999; RIC, (a) 0.999, (b) 0.999; MaxRatings (δ = 0.25), (a) 0.839, (b), 0.668; RDMA, (a) 0.998, (b) 0.997; WDA, (a) 0.980, (b) 0.979; WDMA, (a) 0.996, (b) 0.995; DegSim (100 nearest neighbours), (a) 0.921, (b) 0.913.

Figure 8 indicates that RDMA, WDA and WDMA are much better at detecting Random and Bandwagon attack profiles than Average ones. This is not surprising, because these attack profiles' filler items are always rated around the overall average (which is unlikely to be equal to the filler items' average ratings). As expected, RMAR does not appear to be affected by the Random attack. This feature was not unduly affected by the Bandwagon attack either. For both the Random and Bandwagon attack, the performance of DegSim took a visible hit. This demonstrates how effective the Average attack is, and the marked difference in effectiveness between it and the two attacks in question. The MaxRatings feature did not perform very well with the Bandwagon attack. This is almost certainly because the Bandwagon attack rates a small selection of items with the maximum rating.

Table 3 below summaries the results from a significance test with Random attacks. As with the previous experiment, a paired *t*-test was used with a significance level of 0.01 (1%).

	Filler Ratio			
Feature	**1%**	**3%**	**6%**	**10%**
RMAR	0.994	1.000	1.000	1.000
RIC	0.994	1.000	1.000	0.999
MaxRatings	0.790	0.884	0.901	0.921
RDMA	0.993	0.995	0.998	0.997
WDA	0.757	0.954	0.995	0.994
WDMA	0.980	0.994	0.998	0.996
DegSim	0.838	0.921	0.774	0.508

(a) Mean AUC statistics (3 dp)

	Filler Ratio			
Feature v Feature	**1%**	**3%**	**6%**	**10%**
RMAR v RDMA	0.310	**0.001**	**0.009**	**0.008**
RMAR v WDA	**0.000**	**0.000**	**0.001**	**0.000**
RMAR v WDMA	**0.000**	**0.001**	**0.006**	**0.004**
RMAR v DegSim	**0.000**	**0.000**	**0.000**	**0.000**
RIC v RDMA	0.282	**0.002**	0.018	0.024
RIC v WDA	**0.000**	**0.000**	**0.001**	**0.001**
RIC v WDMA	**0.000**	**0.001**	**0.004**	0.011
RIC v DegSim	**0.000**	**0.000**	**0.000**	**0.000**
RMAR v RIC	0.832	**0.002**	0.049	0.054

(b) Associated *p*-values (3 dp). *p*-values less than or equal to 0.01 are in bold.

Table 3: Results of the features for 10 different Random attack test sets for each filler ratio. MaxRatings had $\delta = 0.25$. And DegSim used 100 nearest neighbours.

Table 3 tells a somewhat different story to the Average attack one. While the performance of the new features (RMAR, RIC and MaxRatings) is more or less the same for the Random attack, DegSim is noticeably poorer whereas RDMA and its derivatives appear to be much better. RDMA, WDA and WDMA may display this behaviour, because they are sensitive to sizeable rating deviations from the norm—a feature of the Random attack. DegSim is likely to be worse at detecting Random attack profiles than Average ones, because Random attacks are inherently weaker, unable to form strong positive correlations with genuine users. For higher filler ratios, even up to 100%, RMAR and RIC exhibited comparable results for each attack. This was predicted to happen for RMAR in Chapter 3. For the sake of brevity, these results are omitted.

It should be noted that RMAR, again, accomplishes statistically perfect detection of attack profiles with a filler ratio greater than 1% here. Interestingly, the table also shows that RMAR and RIC are comparable in terms of performance in detecting Random attack profiles. RIC was shown, above, to be worse at detecting Average attacks when compared to RMAR. Given that the only difference between Random and Average attack profiles is how filler items are rated, these results imply that the scaling component of RIC is superfluous (recall RIC is an extension of RMAR, and largely have similar definitions). This is most likely to be because the Random attack strategy says to rate all filler items around the average rating across the whole rating matrix (3.5 for the MovieLens data set). Thus, all but one of the items (the target) in a Random attack profile will be rated with approximately the same value. This means the scaling component in the definition of RIC is almost always close to 1, effectively reducing RIC to RMAR. This result strengthens the evidence that rating values are not as important for attack detection as the actual items that have been rated by someone.

Results from a similar Bandwagon attack experiment are omitted, because they are largely the same as the results for the Random attack. This was expected because the Bandwagon attack is the same as the Random attack apart from one thing, which is the rating of a very small selection of popular items. These special items are given the maximum rating by the attacker. Since the amount of filler items still dominates a Bandwagon attack profile, RMAR and RIC were not overly affected. And neither were the majority of other features. It was only MaxRatings that was significantly affected. This is likely to have happened because of the special items' maximum ratings, which must have increased the proportion of maximally rated items in an attack profile closer to the average.

4.2.3 Measuring the Effectiveness of a Recommender System Augmented with an RMAR-based Attack Detection Algorithm

The previous experiments showed that the new RMAR feature can detect the Average, Random and Bandwagon attacks with very high probability. It was also shown that the performance of

RMAR visibly drops with any attack profiles that have a filler ratio of less than around 3%. In particular, attack profiles with a filler ratio of 1% had a non-negligible chance of being missed by RMAR. Also, on closer inspection of the results of the previous experiments, a few genuine user profiles were misclassified as belonging to an attacker. In the light of this, we investigated how a popular user-based collaborative filtering algorithm is affected by such conditions. An attack detection algorithm based on RMAR using a discrimination threshold[4] was used. It discarded any profile classed as an attack one (so such profiles were not inserted into the rating matrix and used by the recommendation system).

Here we are primarily interested in the accuracy and robustness of the collaborative filtering algorithm. Accuracy is said to be good if the algorithm still produces sound rating predictions after genuine user profiles classed as an attack profile have been discarded. And robustness is said to be good if the algorithm still makes sound rating predictions after attack profiles classed as a genuine user profile have been inserted into the rating matrix. For consistency with other work in the literature, accuracy will be measured by the mean absolute error (MAE) metric, and robustness will be measured by the prediction shift (PredShift) metric. MAE is defined as:

$$MAE = \frac{1}{|T|} \sum_{(u,i,r) \in T} |r - p_{u,i}|$$

where T is a set of (unseen) test ratings $(u, i, r) \in U \times I \times R$ and $p_{u,i}$ is the rating prediction of item i for user u. MAE values closer to zero point to higher accuracy, and are better than values further away from zero. The prediction shift measure was introduced in [17] and is defined as:

$$PredShift(U_T, I_T) = \frac{1}{|I_T|} \sum_{i \in I_T} \frac{1}{|U_T|} \sum_{u \in U_T} (p'_{u,i} - p_{u,i})$$

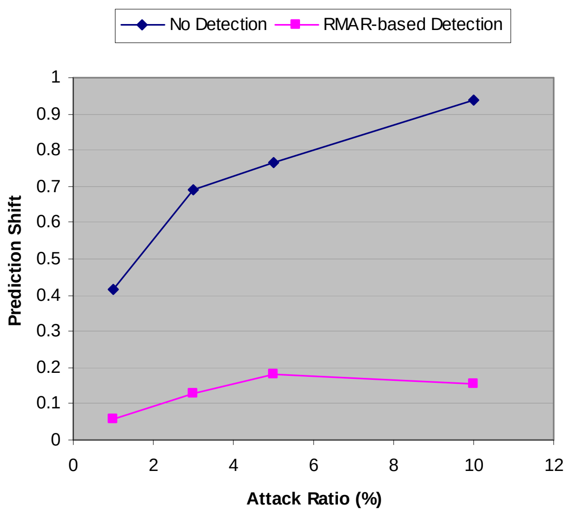
where $U_T \subseteq U$ and $I_T \subseteq I$ are sets of target users and target items respectively, the first term in the inner summation is the rating prediction of item i for user u after an attack, and the second term is the prediction before the same attack. So, prediction shift is the average prediction difference over each target user and target item, and measures the overall affect of a particular attack. A positive prediction shift for an attack implies it was successful, increasing the rating

[4] The threshold was empirically set to 0.2.

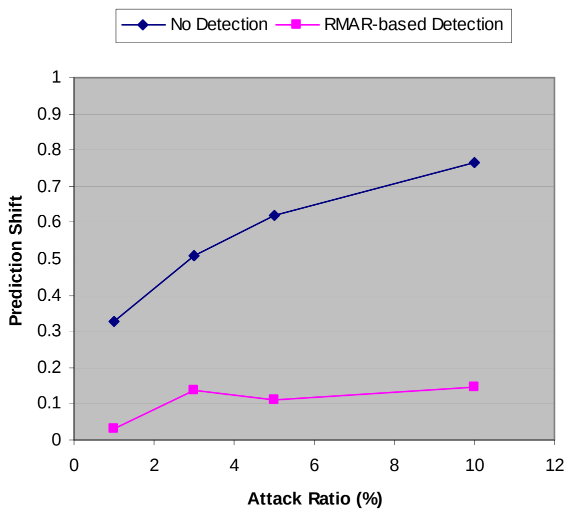
prediction on average for the target users and items. Moreover, the magnitude of the prediction shift says how effective the attack was with respect to those users and items. Any negative prediction shift for an attack clearly means it was largely of no use. The mean rating in the MovieLens data set is 3.5 (1 dp). This implies that a successful attack will result in an average prediction shift of approximately +1.5.

The MAE was calculated with a certain training and test set, before and after the RMAR-based detection algorithm was applied to the training set. The training set initially had 80,000 ratings and the test set had the remaining 20,000 ratings. Before the detector was run the recommender system had a MAE of 0.76081 (5 dp). And after the detector was run it had a MAE of 0.76092 (5 dp). With a *p*-value of 0.293, this result is certainly not statistically significant. Thus, we can conclude the detection algorithm did *not* negatively impact the accuracy of the recommender system.

To calculate prediction shift, fifty random users and items were chosen to be targets. This number of users and items was chosen for computational reasons. The user profiles from the MovieLens data set were separated into two equal halves. One half was used by RMAR to compute item similarities, and to generate attack profiles. The user-based collaborative filtering algorithm used the other half (to compute rating predictions). The rating prediction for each target user and target item pair was then calculated. Next, the attack profiles were injected into the rating matrix used by the recommender system, so that post-attack rating predictions could be produced. This yields the prediction shift for the recommender system without attack detection. After this, the RMAR-based detection algorithm was run on the rating matrix used by the collaborative filtering algorithm, discarding any suspected attack profiles. Using the resultant rating matrix, rating predictions for the same users and items were then computed, finally giving the prediction shift for the recommender system with attack detection. Figure 9 below shows the results of this experiment.



(a) Average attack



(b) Random attack

Figure 9: Graphs of the prediction shifts for some Average and Random attacks using a 1% filler ratio. A user-based collaborative filtering algorithm provided the rating predictions.

Figure 9 shows that, even for attacks with the more covert 1% filler ratio, the RMAR-based detection algorithm neutralised enough attacks to keep prediction shifts below 0.2 (this means rating predictions are still relatively sound, with only a small error). All the prediction shift results were statistically significant at the 99.9% level. Note that the detection algorithm is relatively indifferent to attack size. When the same experiment was done for attacks with higher filler ratios, the prediction shifts for the collaborative filtering algorithm augmented with attack detection were virtually negligible. We can now conclude that the detection algorithm substantially improves robustness and accuracy.

It should now be clear that the new RMAR feature is, beyond reasonable doubt, the best for detecting current profile injection attack methods. Furthermore, it is also actually completely sufficient for this job. The new feature works consistently well across all attacks and filler ratios (even for ratios much less than 1%). Because RMAR can detect any attack that uses random filler items, it is also applicable to the lesser-known Segment, Love/Hate, and Reverse Bandwagon attacks in [6]. The last two attack strategies are specifically designed to sink a target item's rating. RMAR also defeats the even lesser known obfuscated attack class [18].

4.2.4 Evaluation and Detection of Attacks with Non-Random Filler Items

A new class of profile injection attack was defined in Chapter 3. These attack strategies do not choose filler items randomly, but select filler items based on their popularity. Popularity is measured in terms of the number of ratings an item has received. This new class of attack was created in response to the formation of the RMAR feature. To evaluate the strength of the new attacks, we conducted a similar prediction shift experiment to the one before. The only difference was the use of a 3% filler ratio, because this yielded the largest prediction shifts. Here a 3% filler ratio corresponds to just 50 items. Figure 11 below shows the results of this experiment.

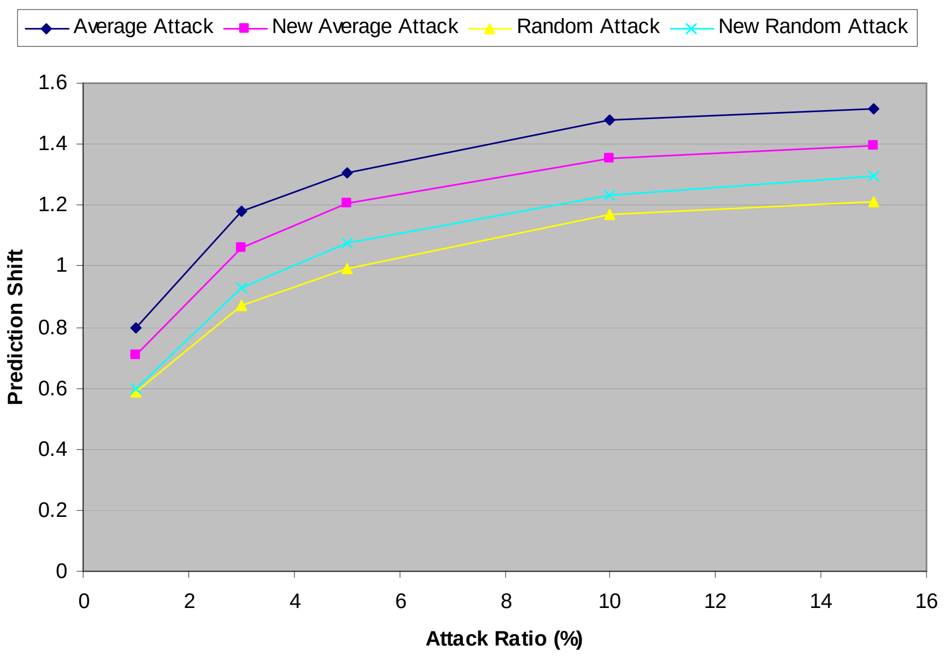
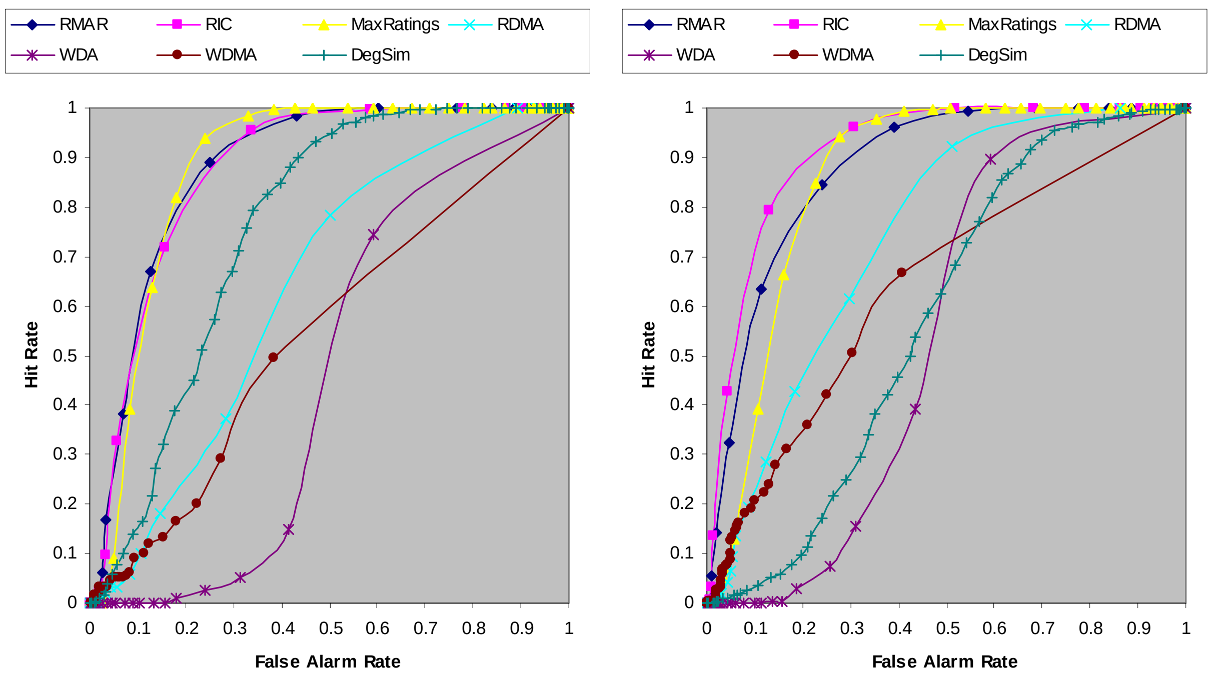


Figure 11: Graph of the prediction shifts for some new and old Average and Random attacks using a 3% filler ratio. A user-based collaborative filtering algorithm provided the rating predictions.

Figure 11 indicates that the strength of the new and old attacks is comparable. Significance tests at the level 0.05 said that none of the prediction shifts were significant. Thus, we can conclude that using popular filler items in place of random ones is not disadvantageous. We are now in a position to see if the new attack can avoid detection.



(a) Average attack with popular filler items **(b) Random attack with popular filler items**

Figure 11: ROC curves indicating how well each feature detects some Average and Random attack profiles with a 3% filler ratio and popular filler items. Corresponding AUC statistics (3 dp): RMAR, (a) 0.874, (b) 0.875; RIC, (a) 0.865, (b) 0.905; MaxRatings (δ = 0.25), (a) 0.877, (b), 0.857; RDMA, (a) 0.630, (b) 0.735; WDA, (a) 0.450, (b) 0.533; WDMA, (a) 0.539, (b) 0.636; DegSim (100 nearest neighbours), (a) 0.757, (b) 0.574.

When compared to the ROC curves of the same attacks using random filler items (Figures 6 and 8), Figure 11 paints a very different picture. The figure shows the new class of attack affects the performance of all the features apart from MaxRatings. In Chapter 3, it was conjectured that RMAR would not be able to detect the new attacks well. This was because the filler items here might be considered as similar. To some extent this is true. However, Figure 11 shows that the performance of RMAR (and RIC and MaxRatings) is still respectable. This suggests that the most popular items are not necessarily related. This could be because popularity is based on an aggregation of all ratings. The above figure also shows that the new class of attack has a very reasonable chance of evading the current features. This adds yet more evidence to our conjecture that ratings values are not critical for attack detection.

From these results we can conclude that RMAR and RIC are not infallible. Our new attacks give a possible direction for how to devise covert profile injection attacks.

5 Conclusions

This report has presented several novel classification features for detecting profile injection attacks on collaborative filtering. They are novel because they are derived from observations on user behaviour. This approach is dramatically different to the ones taken by those who proposed the features described in Chapter 2. However, the results from experiments (in Chapter 4) show that the new RMAR and RIC features (defined in Chapter 3) are consistently significantly better than the current ones.

RMAR was found to be the most successful feature. It can be used to very accurately detect and neutralise all the current profile injection attacks, including the Average, Random and Bandwagon attacks. As mentioned before, the Average attack is the most difficult to detect. More generally, we have shown that *any* profile injection attack using randomly chosen filler items can be detected with very high probability. While this alone foils all current attacks, it also forces any would-be attacker to devise a strategy to intelligently select filler items. This substantially increases the complexity of any new attack. We have proposed and tested one such attack, which chooses filler items based on popularity. This attack was shown to be as successful in effecting prediction shifts as current attacks, but without having an exceedingly high chance of being detected. The new features (RMAR, RIC and MaxRatings) were found to be the best at detecting the new attack. However, perfectly accurate detection could not be attained in this case.

There are now a number of directions for future work. On the defence side, one obvious route would be to improve the detection of our new attack. The success of RMAR suggests that the relationship between a user’s rated items yields useful information. RMAR uses the similarity between item profiles, but other meaningful relationships (such as item genre) might further facilitate the detection of the new attack. Because of RMAR, we have made the choice of filler items as important as the choice of ratings, i.e. nontrivial. So, future work on the attack side includes finding new ways to select filler items such that detection is harder. An optimal set of filler items would actually look like a whole genuine user profile. Preferably, the new method to select filler items would only need a minimal amount of knowledge about users.

Bibliography

[1] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information Tapestry. ACM 1992.

[2] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An Algorithmic Framework for Performing Collaborative Filtering. ACM 1999.

[3] J. S. Breese, D. Heckerman, and C. Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. UAI 1998.

[4] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based Collaborative Filtering Recommendation Algorithms. WWW 2001.

[5] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams. Effective Attack Models for Shilling Item-Based Collaborative Filtering Systems. WebKDD 2005.

[6] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams. Toward Trustworthy Recommender Systems: An Analysis of Attack Models and Algorithm Robustness. ACM 2007.

[7] S. K. Lam, and J. Riedl. Shilling Recommender Systems for Fun and Profit. WWW 2004.

[8] B. Mehta. Unsupervised Shilling Detection for Collaborative Filtering. AAAI 2007.

[9] R. Burke, B. Mobasher, R. Zabicki, and R. Bhaumik. Identifying Attack Models for Secure Recommendation. IUI 2005.

[10] B. Mehta, and W. Nejdl. Attack Resistant Collaborative Filtering. ACM 2008.

[11] P.-A. Chirita, W. Nejdl, and C. Zamfif. Preventing shilling attacks in online

recommender systems. ACM 2005.

[12] C. Williams, R. Bhaumik, R. Burke, and B. Mobasher. The Impact of Attack Profile Classification on the Robustness of Collaborative Recommendation. WebKDD 2006.

[14] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik. Detecting Profile Injection Attacks in Collaborative Recommender Systems. CEC 2006.

[15] B. M. Marlin, R. S. Zemel, S. Roweis, and M. Slaney. Collaborative Filtering and the Missing at Random Assumption. UAI 2007.

[16] R. O. Duda, P. E. Hart, and D. G. Stork. Pattern Classification, Second Edition. Wiley. 2001.

[17] M. O'Mahony, N. Hurley, N. Kushmerick, and G. Silvestre. Collaborative recommendation: A robustness analysis. ACM 2004.

[18] C. Williams, B. Mobasher, R. Burke, J. Sandvig, and R. Bhaumik. Detection of Obfuscated Attacks in Collaborative Recommender Systems. ECAI 2006.