Limited Knowledge Shilling Attacks in Collaborative Filtering Systems

Robin Burke, Bamshad Mobasher, and Runa Bhaumik
DePaul University
Center for Web Intelligence
School of Computer Science, Telecommunications and Information Systems
{rburke, mobasher, rbhaumik}@cs.depaul.edu

Abstract

Recent research in recommender systems has shown that collaborative filtering algorithms are highly susceptible to attacks that insert biased profile data. Theoretical analyses and empirical experiments have shown that certain attacks can have a significant impact on the recommendations a system provides. These analyses have generally not taken into account the cost of mounting an attack or the degree of prerequisite knowledge for doing so. For example, effective attacks often require knowledge about the distribution of user ratings: the more such knowledge is required, the more expensive the attack to be mounted. In our research, we are examining a variety of attack models, aiming to establish the likely practical risks to collaborative systems. In this paper, we examine user-based collaborative filtering and some attack models that are successful against it, including a limited knowledge "bandwagon" attack that requires only that the attacker identify a small number of very popular items and a user-focused "favorite item" attack that is also effective against item-based algorithms.

1 Introduction

Recommendation systems are an increasingly important component of electronic commerce and other information access systems. Users have come to trust personalization and recommendation software to reduce the burden of navigating large information spaces and product catalogs. However, it has now been well-established that the most widely-used and well-understood collaborative algorithm, user-based nearest-neighbor collaborative filtering is highly susceptible to attack [8,13]. A hostile agent can insert biased profiles into such a system to cause it to favor certain products.

The overall goal of our research is to look at recommender systems of all kinds, collaborative and others, and identify algorithms and approaches that are robust against attacks. The example of the Google search engine[1] is a real-world case in which a hybrid approach adding collaborative, link-based data serves as a measure of defense against attackers manipulating the basic data on which search engine recommendations are based (the word-level features of web pages.) We are interested in the spectrum of recommendation hybrids and their potential benefits for recommendation security [2].

The issue of the injection of bias is not limited to recommender systems. Any open personalization system is by definition vulnerable to the introduction of biased data generated by attackers interested in biasing the system's results to suit their own ends.

The primary attack models used in prior research are the *sampling attack* [13], where biased profiles are constructed from samples of the actual user data; the *random attack* where [8] user profiles are generated randomly based on the overall distribution of user ratings in the database; and the *average attack* [8], where the rating for each item is computed based on its average rating for all users. The sampling attack has useful theoretical properties but is not one that can be realistically mounted. The average and random attacks, as previously envisioned, require that the attacker create complete profiles: a rating for each item, also probably an unrealistic requirement.

This paper addresses the question of practical attack models. Can an attacker armed with reasonable guesses be successful, or must an attack be based on detailed knowledge of the rating distribution? How large do attack profiles need to be in order to be successful? One type of attack that requires both limited knowledge and modest profile size is the "bandwagon attack", introduced in [3]. The goal of the bandwagon attack is to associate the attacked item with a small number of frequently-rated items. Another type of attack, called the “favorite item” attack, looks at the knowledge about a target user's preferences, rather than knowledge about items ratings of all users on items. Our experiments indicate that attacks can be successful against collaborative filtering systems even when they use limited knowledge. Indeed several of the attack models improve in performance when limited in their scope and the bandwagon attack in particular is not dependent on any knowledge about the data distribution inside the system.

[1] www.google.com

	Item1	Item2	Item3	Item4	Item5	Item6	Correlation with Alice
Alice	5	2	3	3		?	
User1	2		4		4	1	-1.00
User2	2	1	3		1	2	0.33
User3	4	2	3	2		1	0.90
User4	3	3	2		3	1	0.19
User5		3		2	2	2	-1.00
User6	5	3		1	3	2	0.65
User7		5		1	5	1	-1.00
Attack1	2		3		2	5	-1.00
Attack2	3	2	3		2	5	0.76
Attack3	3	2	2	2		5	0.93

Figure 1. A *push* attack favoring Item6.

2 Attack Models and Attacker Knowledge

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 the attack? Different attack models make different assumptions about what the attacker knows and can be differentiated on that basis.

An attack against a collaborative filtering recommender system consists of a set of *attack profiles*, biased profile data associated with fictitious user identities, and a *pushed item*, the item that the attacker wishes the system to recommend more highly.[2] An attack model is an approach to constructing the attack profile, based on knowledge about the recommender system, its rating database, its products, and/or its users. The form of a *push* attack profile is depicted in Figure 2. An attack profile consists of a *m*-dimensional vector of ratings, were *m* is the total number of items in the system. The rating given to the pushed item, **target**, is r_{max} and is the maximum allowable rating value. The ratings r_1 through r_{m-1} are assigned to the corresponding items according to the specific attack model. Indeed, the specific strategy used to assign ratings to items 1 through *m*-1 is what determines the type of attack model used.

2.1 An Illustrative Example

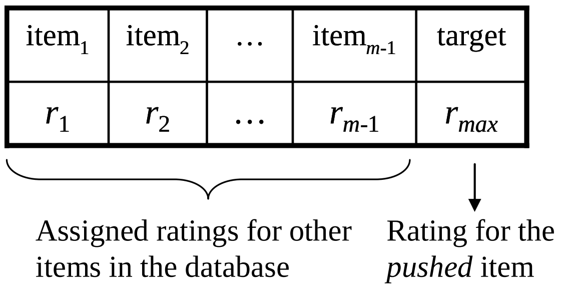
Consider, as an example, a recommender system that identifies books that users might like to read using a user-based collaborative algorithm [5]. A user profile in this hypothetical system might consist of that user's ratings (in the scale of 1-5 with 1 being the lowest) on various books (items). Alice, 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 agent, Eve, has inserted *attack profiles* (Attack1-3) into the system, all of which give high ratings to her book labeled Item6. Without the attack profiles, the most similar user to Alice, using a correlation coefficient, would be User3. The prediction associated with Item6 would be 1, essentially stating that Item6 is likely to be strongly disliked by the user. If the algorithm used the closest 3 users (users 3, 6, and 2), the system would still be unlikely to recommend the item.

Eve's attack profiles may closely match the profiles of 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. In the example of Figure 1, the Attack3 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. Taking the most similar 3 users in this small database would not offer any defense: Attack3, User3, and Attack2 would be selected and Item6 would still receive an above average recommendation score. So, in this example, the attack is successful, and Alice will likely 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.

Prior work on recommender system stability has examined primarily three types of attack models:

- **Sampling attack**: A sampling attack is one in which attack profiles are constructed from entire user

[2] It is also possible to mount an attack aimed at preventing an item from being recommended: what [13] calls the "nuke" attack. We concentrate here on the "push" attack.



item$_1$ item$_2$ … item$_{m-1}$ target
r_1 r_2 … r_{m-1} r_{max}
Assigned ratings for other items in the database
Rating for the *pushed* item

Figure 2. A *push* attack profile.

profiles sampled from the actual profile database, augmented by a positive rating for the pushed item. This attack is used by O'Mahony et al. [8] to provide a proof of the instability of collaborative filtering algorithms, but is the least practical from a knowledge point of view.

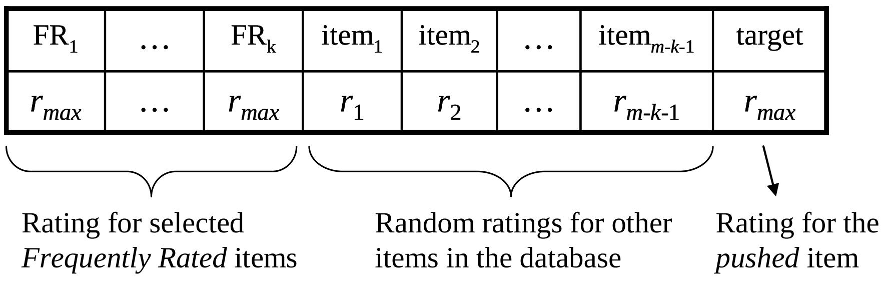
- **Random attack**: Lam et al. [8] show an attack model in which profiles consist of random values (except of course for a positive rating given to the pushed item). Specifically, r_1 through r_{m-1} are assigned to the corresponding items by generating random values within the rating scale with a distribution centered around the mean for all user ratings across all items. 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 effort involved, however, is still substantial, since it involves assigning ratings to every item in each attack profile. Furthermore, as we shall see in the following, the attack is not particularly effective.

- **Average attack**: A more powerful attack described in [8] uses the individual mean for each item rather than the global mean (except again the pushed item.) In the average attack, each assign rating, r_i, in an attack profile corresponds (either exactly or approximately) to the mean rating for item$_i$, across the users in the database who have rated that item. In addition to the effort involved in producing the ratings, the average attack also has considerable knowledge requirements, namely, of order m where m is the number of products in the profile database. Our research, however, suggests that the average attack can be just as successful by assigning the average ratings to a small subset of items in the database, thus substantially reducing the knowledge requirement.

In addition to these of attack models, we are investigating a number of others, some of which were introduced in [3]. In this paper, we specifically focus on two new attack models:

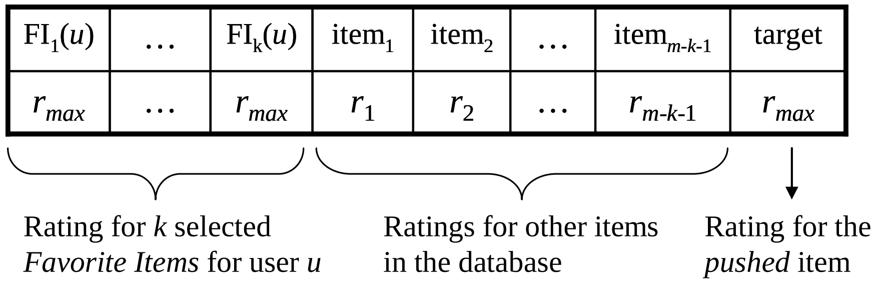
- **Bandwagon attack**: This attack takes advantage of the Zipf's law distribution of popularity in consumer markets – a small number of items, best-seller books for example, will receive the lion's share of attention and also ratings. The attacker using this model will build attack profiles containing 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, 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 has the benefit of not requiring any system-specific data – it is usually not difficult to independently determine what the "blockbuster" products are in any product space.

  Figure 3 depicts a typical attack profile for the bandwagon attack. Items FR$_1$ through FR$_k$ 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. The ratings r_1 through r_{m-k-1} for the other items are determined randomly in a similar manner as in the random attack. It is, therefore, important to note that the bandwagon attack can be viewed as an extension of the random attack. However, as we shall



FR$_1$ … FR$_k$ item$_1$ item$_2$ … item$_{m-k-1}$ target
r_{max} … r_{max} r_1 r_2 … r_{m-k-1} r_{max}
Rating for selected *Frequently Rated* items
Random ratings for other items in the database
Rating for the *pushed* item

Figure 3. A *Bandwagon* attack profile.



$FI_1(u)$ … $FI_k(u)$ $item_1$ $item_2$ … $item_{m-k-1}$ target
r_{max} … r_{max} r_1 r_2 … r_{m-k-1} r_{max}
Rating for *k* selected *Favorite Items* for user *u*
Ratings for other items in the database
Rating for the *pushed* item

Figure 4. A *Favorite Item* attack profile.

see in the following, the bandwagon attack can still be successful even when only a small subset of the “random items”, $item_1$ through $item_{m-k-1}$ are assigned ratings.

- **Favorite item attack**: (called the "consistency attack" in [3]) Rather than knowledge about items, the favorite item attack looks at knowledge of user's preferences. Such an attack is mounted not against the system as a whole, but by targeting a given user (or a group of users). We assume that the attacker knows which items a given user, *u*, really likes, and builds profiles containing only those items. Like the sampling attack, this attack is not particularly practical from a knowledge standpoint, but provides an upper bound on the effectiveness of other attacks focused on user characteristics. It could, however, be extremely effective if the attacker has substantial knowledge about an individual. Figure 4 depicts a typical attack profile for the favorite item attack. $FI_i(u)$ represent the favorite items by user u selected in the attack profile. These items are assigned maximum rating value together with the target item. On the other hand, the other items in the database, $item_1$ through $item_{m-k-1}$ are assigned ratings at random or based on other criteria. In our experiments, best results were obtained when the non-favored items are assigned the lowest possible rating. The favorite item attack has the benefit of being effective against both user-based and item-based algorithms as our experiments below confirm.

3 Background on Recommendation Algorithms

We have concentrated in this work on the most commonly-used algorithms for user-based and item-based collaborative filtering. Each algorithm assumes that there is a user / item pair for whom a prediction is sought, the *target user* and the *target item*. The task for the algorithm is to predict the target user's rating for the target item.

3.1 User-based Collaborative Filtering

The standard collaborative filtering algorithm is based on user-to-user similarity [5]. The algorithm operates by selecting the *k* most similar users to the target user, and computes a prediction by combining the preferences of these users. The similarity between the target user, *u*, and a neighbor, *v*, is usually calculated using the standard Pearson’s *r* correlation coefficient, which we denote by $sim_{u,v}$. 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*.

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

where V is the set of k similar users and $r_{v,i}$ is the rating of those users who have rated item *i* and $\bar{r}_u$ is the average rating for the target user over all rated items, $sim_{u.v}$ 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.

3.2 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 *k*NN 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 [14]. The adjusted cosine similarity formula is given by

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

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

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

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.

3.3 Evaluation Metrics

There has been considerable research in the area of recommender systems evaluation [6]. 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 [13] 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. One way to measure this change in likelihood is to measure *prediction shift*, the average change in the predicted rating for the attacked item before and after the attack [8].

Our average prediction shift is defined as follows. Let *U* and *I* be the set of target users and items. 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 succeeding in making the pushed item more positively rated. The average prediction shift for an item *i* over all users can be computed as

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

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

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

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.

We plan to explore other metrics based on recommendation behavior, such as the bin-based techniques suggested in [8] and others, in our future work.

4 Experiments with Attack Models

In our experiments we have used the publicly-available Movie- Lens dataset [12]. 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. To perform our attack experiments, we must average over a number of different attack items, so we selected 50 movies taking care that the distribution of ratings for these movies matched the overall ratings distribution of all movies. We also selected a sample of 50 users as our test data, again

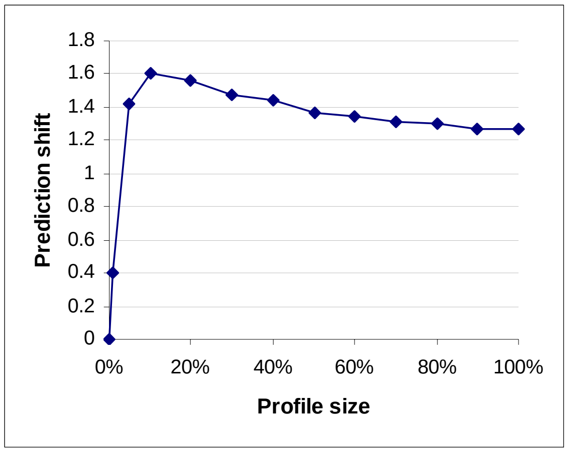


Figure 5. Prediction shift with varying profile sizes in the average attack

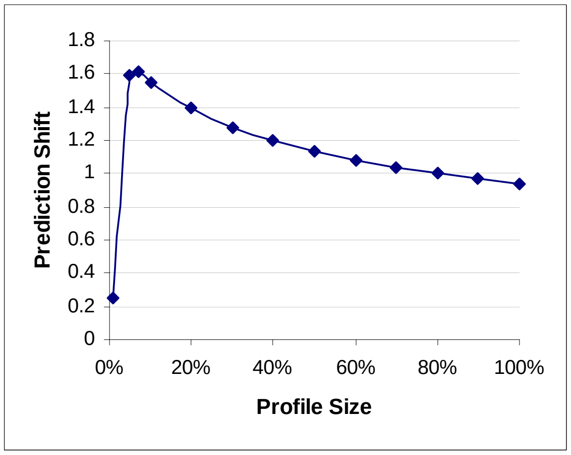


Figure 6. Prediction shift with varying profile sizes in the bandwagon attack.

mirroring the overall distribution of users in terms of number of movies seen and ratings provided.

4.1 Attacks Based on Knowledge about Items

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 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 experimented with variants of the average attack in which random subsets of the items of different sizes were chosen for inclusion in attack profiles. This experiment uses a (rather large) attack size of 15%, meaning that the number of attack profiles inserted is equal to 15% of the original pre-attack user database. Figure 5 shows the rather surprising results of this experiment. The “profile size” in the Figure corresponds to the percentage of items 1 through *m*-1 (see Figure 2), that were assigned average ratings in the attack profile. The remaining items in the database (with the exception of the target item) were left with no ratings.

As the amount of the knowledge increases to around 10%, the prediction shift rises sharply to around 1.6 but after this point it drops off gradually ending at 100% around 1.3. 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. The attack has to achieve a balance between coverage (including enough movies so that it can be used for comparison with any given user) 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.

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 that certain items are the popular ones that users are likely to have rated. The attacker selects *k* such items that will be rated highly along with the target item. The profile size, in this case, is the proportion of the remaining *m*-*k*-1 items that are assigned random ratings based on the overall data distribution across the whole database (see Figure 3). In the case of MovieLens, 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 6 shows the effect of profile size on the effectiveness of this attack. In this particular experiment we set *k* (in Figure 3) to 1 Thus., only selected the most frequently rated items in the database to be rated in conjunction with the target item. We also used an attack size of 10% (i.e., the number of attack profiles were about 10% of the size of the original database). For the bandwagon attack, the best results were obtained by using a 7% profile size (i.e., in each attack profile, only 7% of items, beyond the selected frequent items, were assigned ratings). This number seems to correspond closely to the average number of movies rated by a random user in the database.

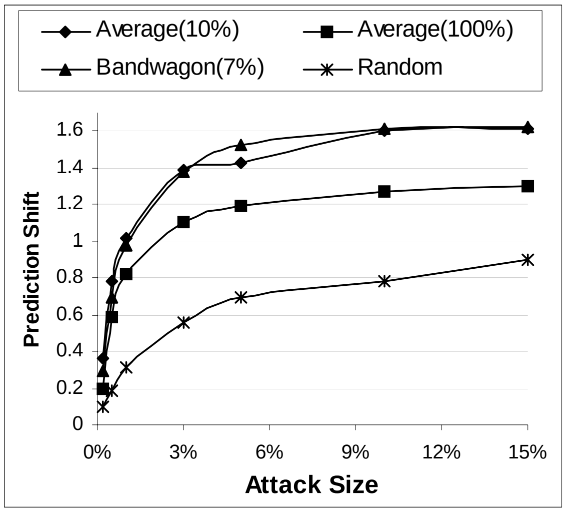


Figure 7. Comparing attack models at different attack sizes

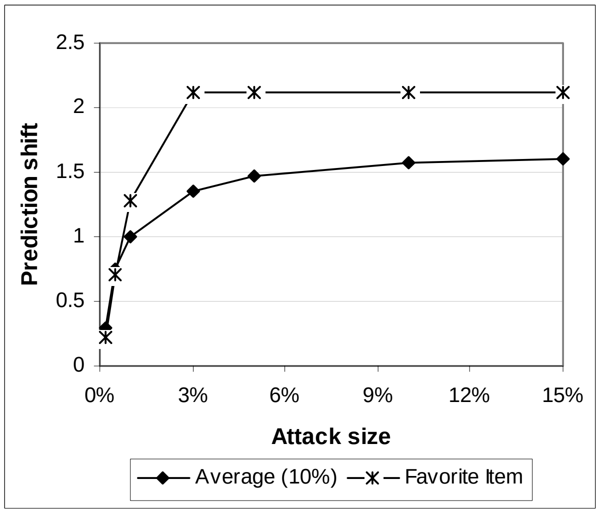


Figure 8. The favorite item attack

Figure 7 shows the results of a comparative experiment examining four algorithms at different attack sizes. The algorithms include the average attack (10% and 100% profile sizes), the bandwagon attack (using 1 frequently rated item and 7% profile size), and the random attack.

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, but the best performance is achieved with relatively small profiles. The summative 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 able to surpass the average attack in performance, even with 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.

4.2 Attacks Based on Knowledge about Users

We might imagine that an attacker could have more direct specific knowledge about individual users of a system. The attacker might have demographic and marketing data that sorts the users into market segments whose preferences might be highly predictable. The *segmented* attack introduced in [3] is one that we intend to pursue in our continuing research. In this paper, we look at a kind of upper bound for the utility of this kind of knowledge in the form of the *favorite item* attack (see Figure 4).

The favorite item attack assumes that we can identify a handful of items that each user likes. Liked items are most likely to be rated – users can often predict that they will not like a particular movie and therefore avoid seeing it. Attack profiles can then be assembled that consist of these liked items and the pushed movie. Other movies are assigned low ratings. Note that a new attack must be formulated for each target user. This is not practical, of course, but if we generalize from the single user to a market niche of users with similar tastes, it becomes plausible that an attacker might construct an attack targeted only to that niche. Figure 8 shows that this attack, with its user-specific knowledge, is even more successful than the best average attack results previously seen.

The favorite item attack was introduced in [3] as an approach that appeared to have a theoretical advantage over previously-developed attack types when applied to the item-based collaborative filtering. Lam et al. [8] showed that the average attack was significantly less successful against item-based algorithms and suggested that item-based algorithms were generally more robust. More work remains to be done on this question. However, Figure 9 demonstrates that the favorite item attack is much more successful against the item-based attack than the average attack, and the effect is quite a bit stronger than its advantage against the user-based algorithms. This is because the favorite item attack directly manipulates the columns of the rating matrix. We plan to explore more practical variants of the favorite item attack in our future work.

5 Conclusions and Future Work

Our experiments have shown that attacks can be successful against collaborative filtering systems even when they use limited knowledge. Indeed several of the

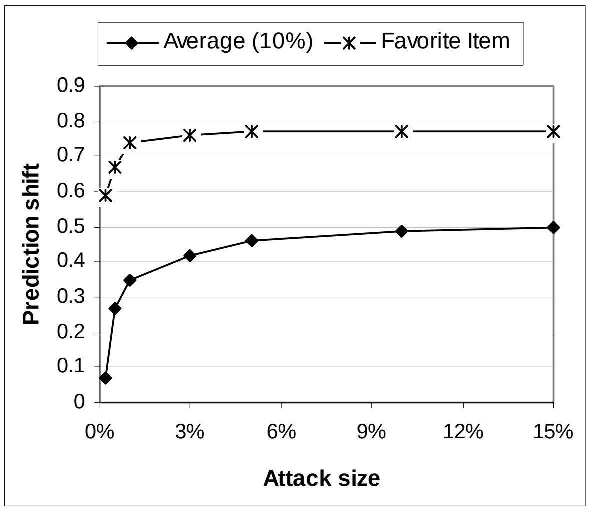


Figure 9. The favorite item attack against item-based collaborative filtering

attack models improve in performance when limited in their scope and the bandwagon attack in particular is not dependent on any knowledge about the data distribution inside the system.

Our research here makes use of the explicit numeric ratings available in the MovieLens data set. Many recommender systems make use of implicit ratings, ratings that are inferred from user behavior, rather than explicitly provided by the user. (See the research reviewed in [7].) Such data sources may have different characteristics than the classic explicit rating scenario. In Web usage mining [4] [15], Web server logs are examined for link traversal and dwell time and continuously-valued ratings derived from this analysis, and as a result, negative ratings are not available from Web usage data. Web usage mining techniques, such as clustering and association rule discovery [9] [10] [11], are alternate recommendation mechanisms whose susceptibility to bias have yet to be explored.

References

[1] Breese, J., Heckerman, D. and Kadie, C. Empirical analysis of predictive algorithms for collaborative filtering. In *Conf. Uncertainty in Artificial Intelligence*, 1998.

[2] Burke, R.: 2002, Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User Adapted Interaction*, *12* (4), 331-370.

[3] Burke, R., Mobasher, B., Zabicki, R. & Bhaumik, R. Attack Models for Secure Recommendation. In *Beyond Personalization: A Workshop on the Next Generation of Recommender Systems*. San Diego, CA. January 2005.

[4] Cooley, R., Mobasher, B. and Srivastava, J. Data preparation for mining world wide web browsing patterns. *Journal of Knowledge and Information Systems*, *1*(1), 1999.

[5] Herlocker, J., Konstan, J., Borchers, A. and Riedl, J. An algorithmic framework for performing collaborative filtering. In *Proceedings of ACM SIGIR*, 1999.

[6] Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004) "Evaluating Collaborative Filtering Recommender Systems". *ACM Transactions on Information Systems*. 22 (1).

[7] Kelly, D and Teevan, J. Implicit feedback for inferring user preference: A bibliography. *ACM SIGIR Forum*, *37*(2), Fall 2003.

[8] Lam, S. and Reidl, J. Shilling Recommender systems for fun and profit. In *Proceedings of the13th International WWW Conference*. New York, May 2004.

[9] Mobasher, M., Cooley, R., and Srivastava, J. Automatic personalization based on web usage mining. *Communications of the ACM*, *43*(8):142-151, 2000.

[10] Mobasher, B., Dai, H., Luo, T., and Nakagawa, M. 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, November 2001.

[11] Mobasher, B., Dai, H., Luo, T., and Nakagawa, M. Discovery and evaluation of aggregate usage profiles for web personalization. *Data Mining and Knowledge Discovery*, 6:61-82, 2002.

[12] MovieLens group. MovieLens 100K Dataset. http://www.cs.umn.edu/research/GroupLens/data/.

[13] O'Mahony, M., Silvestre, G. and Hurley, N. Collaborative Recommendation: A Robustness Analysis. *ACM Transactions on Internet Technology*.

[14] Sarwar, B., Karypis, G., Konstan, J. and Riedl, J. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International WWW Conference*, Hong Kong, May 2001.

[15] Srivastava, J., Cooley, R.,, Deshpande, M., and Tan, P. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explorations*, *1*(2):12-23, 2000.