Evaluation of Profile Injection Attacks In Collaborative Recommender Systems

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

Abstract

Significant vulnerabilities have been identified in collaborative recommender systems. The open nature of collaborative filtering allows attackers to inject biased profile data and force the system to "adapt" in a manner advantageous to them. Previous work has shown both user-based and item-based recommender systems are vulnerable to the segment attack model. In this paper we focus on two techniques that may be used to reduce the impact of the segment attack and also examine their robustness against traditional attack models. One technique is a model-based algorithm based on probabilistic latent semantic analysis that offers significant improvements in stability and robustness against all identified attack profiles. Second, we analyze the effectiveness of a supervised classification approach to detection we have introduced for protecting systems against traditional attacks.

1 Introduction

Recent research has begun to examine the vulnerability of collaborative filtering recommender systems to "shilling" attacks [1, 2, 3, 4]. The more descriptive phrase "profile injection attacks" is also used, and better conveys the tactics of an attacker. In a profile injection attack, an attacker interacts with a collaborative recommender system to build a number of profiles associated with fictitious identities. The aim is to bias the system's output in the attacker's favor. User-adaptive systems, such as collaborative filtering, are vulnerable to such attacks precisely because they rely on user interaction to generate recommendations.

In user-based collaborative filtering, the algorithm collects profiles representing each user's preferences. A recommendation is then produced for an active user by combining the preferences of peers with similar profiles. Biased data in the profile database may be mistaken for genuine users and compromise the accuracy of recommendations [3, 4]. Thus, It is easy to see why collaborative filtering is susceptible to profile injection attacks.

Lam et al. [3] show that item-based collaborative filtering offers an advantage over the user-based approach. In item-based collaborative filtering, a recommendation consists of items that have similar rating profiles to items the active user has already rated highly. By adding biased user profiles, an attacker only alters a portion of the rating profile for any given item. In [1], we introduced the segment attack to specifically target item-based recommendation. The segment attack targets individual segments of users that are predisposed to specific,

*This research is supported, in part, by the National Science Foundation Cyber Trust Grant IIS-0430303.

predictable items.

To overcome this, we have examined model-based approaches to collaborative filtering to provide improved robustness against all profile injection attacks, including the segment attack. Since models create an abstraction of the detailed user profiles, the influence of an attack is reduced because attack profiles are not directly used in recommendation. We have focused on *Probabilistic Latent Semantic Analysis* (PLSA) to infer hidden relationships among groups of users. Each cluster of similar users represents an aggregate profile that is used for recommendation, rather than the original user data. PLSA is a "fuzzy" approach, in that each user has a degree of association with every user cluster. This allows particularly authoritative users to exercise greater influence on recommendation.

In addition to choosing a robust recommendation algorithm, a system can insulate itself from attack through detecting and discounting known attack profiles. Our approach is based on supervised classification of attack profiles and genuine user profiles. The principle behind this approach is that an attacker must generate profiles that are more realistic and contain less bias to avoid detection. By doing this, the profiles are likely to be less effective at influencing recommendation behavior, and more profiles would be needed for effect. However, large attacks are conspicuous. In this way, we hope to render profile injection attacks relatively harmless.

2 Profile Injection Attacks

A profile injection attack against a collaborative recommender system consists of a number of attack profiles added to the database of real user profiles. The goal of a push attack is to increase the system's predicted rating on a target item for a given user (or group of users). An attack model is an approach to constructing attack profiles, based on knowledge about the recommender system, its rating database, its products, and/or its users.

2.1 An Example

Consider, as an example, a recommender system that identifies books that users might like to read using a user-based collaborative algorithm [5]. In general, an user builds up a profile (in the scale of 1-5 with 1 being the lowest) on various books and returns to the system for new recommendations. Figure 1 shows Alice's profile along with that of seven genuine users. An attacker, Eve, has inserted attack profiles (Attack1-3) into the system, all of which give high ratings to her book labeled Item6. Eve's attack profiles may closely match the profiles of one or more of the existing users based on the ratings given across the attack profiles.

Without the attack profiles, a standard user-based collaborative filtering system, which uses correlation-based similarity, will find User6 as the most similar to Alice and the predicted ratings for Alice on Item6 will be 2, essentially stating that Item6 is likely to be disliked by Alice. After the attack, however, the Attack1 profile is the most similar one to Alice, and would yield a predicted rating of 5 for Item6, the opposite of what would have been predicted without the attack. In this example, the attack is successful, and Alice will get Item6 as a recommendation, regardless of whether this is really the best suggestion for her. She may find the suggestion inappropriate, or worse, she may take the system's advice, buy the book, and then be disappointed by the delivered product.

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

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

2.2 Attack Models

The generic form of an attack profile is depicted in Figure 2. Specific attack models define the method for assigning ratings to the set of *filler items* and the target item. The set of filler items represent a group of randomly selected items in the database that are assigned ratings within the attack profile. In certain attack models, a subset of filler items may be pre-selected for a precise impact. The target item in a push attack is generally given the maximum allowed rating. Prior work on recommender system stability has examined primarily two types of attack models:

- Random Attack [3], where user profiles are generated randomly based on the overall distribution of user ratings in the database;
- Average attack [3], where the rating for each item is computed based on its average rating for all users.

In practice, an average attack is much more effective than a random attack. However, it requires greater knowledge about the system's rating distribution. This knowledge cost is minimized by the fact that an average attack can be quite successful with a small filler item set, whereas a random attack usually must have a rating for every item in the database in order to be effective.

An extension of the random attack, the *bandwagon attack* [1, 2] is nearly as effective as the average attack. The goal of a bandwagon attack is to associate the target item with a small number of frequently rated items. This takes advantage of the Zipf's law distribution: a small number of items will receive the lion's share of ratings. In a bandwagon attack, a small set of frequently rated items are selected along with the set of random filler items. Attack profiles give maximum rating to those items that have high visibility, and therefore have a good probability of being similar to a large number of users.

Random, average, and bandwagon attack models are not particularly effective against item-based collaborative filtering. In response, the *segment attack* was introduced in [1] and further examined in [2, 6, 7]. It turns out that a segment attack is also quite effective against user-based algorithms. A segment attack attempts to target a specific group of users who may already be predisposed toward the target item. For example, an attacker that wishes to push a fantasy book might want the product recommended to users expressing interest in *Harry Potter* and *Lord of the Rings*.

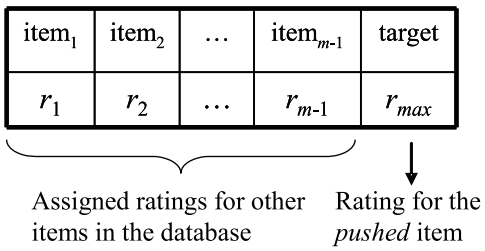


Figure 2: The general form of a push attack profile.

A typical segment attack profile consists of a number of selected items that are likely to be favored by the targeted user segment, in addition to the random filler items. This differs from a bandwagon attack in that the selected items are expected to be highly rated within the targeted user segment, rather than frequently rated. The selected segment items are assigned the maximum rating value along with the target item. To provide the greatest impact on item-based algorithms, all remaining filler items are given the minimum allowed rating.

3 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 kNN collaborative filtering algorithm is based on user-to-user similarity [5]. In selecting neighbors, we have used Pearson's correlation coefficient for user-user similarities and a neighborhood size $k = 20$. 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, predictions are calculated as described in [7].

3.2 Item-Based Collaborative Filtering

Item-based collaborative filtering works by comparing items based on their pattern of ratings across users. For our purpose we have adopted the adjusted cosine similarity measure introduced by [8]. Once again, we consider a neighborhood of size $k = 20$ and ignore items with negative similarity. Once the most similar items are identified, predictions are calculated as described in [7].
In addition to these common collaborative filtering methods, we have also examined some additional model-based approaches as a means of increasing robustness to attack.

3.3 Model-Based Collaborative Filtering

A standard model-based collaborative filtering algorithm uses k-means to cluster similar users. Given a set of user profiles, the space can be partitioned into k groups of users that are close to each other based on a measure of similarity. The discovered user clusters are

then applied to the user-based neighborhood formation task, rather than individual profiles.

A more successful approach based on PLSA models [9] provides a probabilistic method for characterizing latent or hidden semantic associations among co-occurring objects. In [10, 11] PLSA was applied to the creation of user clusters based on web usage data. We have adapted this approach to the context of collaborative filtering.

Given a set of n users, $U = \{u_1, u_2, \cdots, u_n\}$, and a set of m items, $I = \{i_1, i_2, \cdots, i_m\}$ the PLSA model associates an unobserved factor variable $Z = \{z_1, z_2, \cdots, z_l\}$ with observations in the rating data. For a target user u and a target item i, the following joint probability can be defined:

$$P(u, i) = \sum_{k=1}^{l} Pr(z_k) \bullet Pr(u|z_k) \bullet Pr(i|z_k)$$

In order to explain a set of ratings (U, I), we need to estimate the parameters $Pr(z_k)$, $Pr(u|z_k)$, and $Pr(i|z_k)$, while maximizing the following likelihood $L(U, I)$ of the rating data:

$$L(U, I) = \sum_{u \in U} \sum_{i \in I} r_{u,i} \bullet \log Pr(u, i)$$

where $r_{u,i}$ is the rating of user u for item i.

We can now identify segments of users that have similar underlying interests. For each latent variable z_k, we create a user cluster C_k and select all users having probability $Pr(u|z_k)$ exceeding a certain threshold μ. If a user does not exceed the threshold for any latent variable, it is associated with the user segment of highest probability. Thus, every user profile will be associated with at least one user segment, but may be associated with multiple segments. This allows authoritative users to have broader influence over predictions, without adversely affecting coverage in sparse rating data.

4 Evaluation Metrics

There has been considerable research in the area of recommender systems evaluation [12]. Some of these concepts can be applied, but in evaluating security we are interested not in raw performance, rather the change in performance induced by an attack. Our goal is to measure the effectiveness of an attack - the "win" for the attacker. The desired outcome for the attacker in a "push" attack is of course that the pushed item be more likely to be recommended after the attack than before. In the experiments reported below, we follow the lead of [4] in measuring an algorithms stability via prediction shift. The prediction shift metric as computed in [2] measures the change in the predicted rating of an item before and after attack. We have also confirmed our results using other metrics such as hit ratio, the average likelihood that a top N recommender will recommend the pushed item, but for brevity have limited the results to just prediction shift [8, 7].

5 Experimental Results of Attack Models

In our experiments we have used the publicly-available Movie-Lens 100K dataset[1]. This dataset consists of 100,000 ratings on 1682 movies by 943 users. All ratings are integer values

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

between one and five where one is the lowest (disliked) and five is the highest (most liked). Our data includes all the users who have rated at least 20 movies.

In all experiments, we used a neighborhood size of 20 in the k-nearest-neighbor algorithms for user-based and item-based systems. To ensure the generality of the results, 50 movies were selected randomly that represented a wide range of average ratings and number of ratings. Each of these movies was attacked individually and the average is reported for all experiments. We also generally selected a sample of 50 users as our test data, mirroring the overall distribution of users in terms of number of movies seen and ratings provided. The results reported below represent averages over the combinations of test users and test movies. We use the metric prediction shift, as described earlier, to measure the relative performance of various attack models.

Another aspect of attack profiles that impact their effectiveness is filler size or the percentage of items in the profile that are given ratings. For all the attack models we have studied, the effectiveness of the attack varies as a function of the filler size. For brevity we have not included these results, but have selected the filler size for each attack model/recommendation algorithm pair that produces the largest prediction shift. For all the attacks, we generated a number of attack profiles and inserted them into the system database and then generated predictions. We measure "size of attack" as a percentage of the pre-attack user count. There are approximately 1000 users in the database, so an attack size of 1% corresponds to 10 attack profiles added to the system.

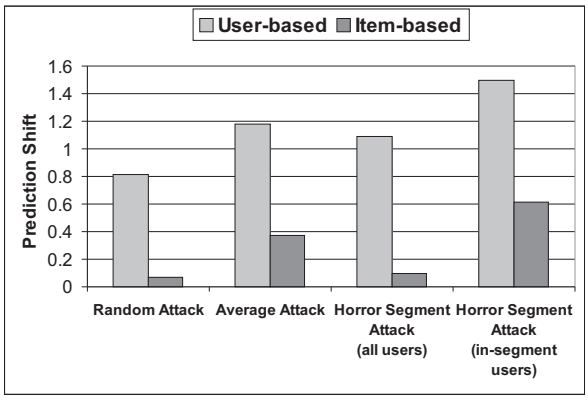
5.1 Vulnerability of Popular Collaborative Filtering Algorithms

Figure 3(a) shows the results of a comparative experiment examining three attacks against the two most popular collaborative recommendation algorithms at a 1% attack. Recall that the rating scale in this domain is 1-5 with an average of 3.6, so a rating shift of 1.4 is enough to lift an average-rated movie to the top of the scale. As the figure shows, the additional system knowledge required for average attack results in a significant increase in prediction shift over random attack. The item-based algorithm on the other hand appears more robust than the user-based algorithm without significant knowledge requirements. This observation led us to develop the segment attack intended specifically to manipulate the item similarity used by the item-based algorithm while still requiring limited knowledge.

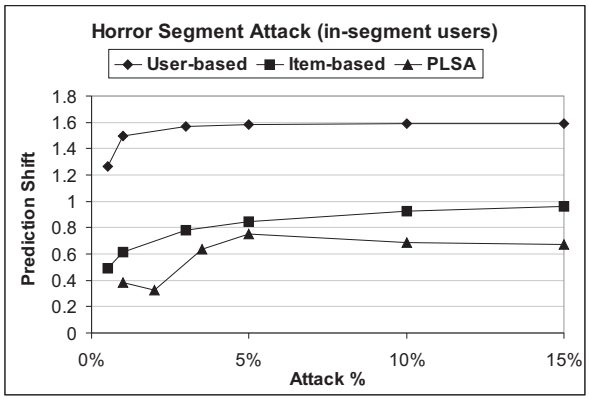
To build our segmented attack profiles, we identified the segment of users as all users who had given above average scores(4 or 5) to any three of the five horror movies, namely, *Alien*, *Psycho*, *The Shining*, *Jaws*, and *The Birds*.[2] For this set of five movies, we then selected all combinations of three movies that had at least 50 users support, and chose 50 of those users randomly and averaged the results.

From Figure 3(a) we see if we evaluate the segmented attack based on its average impact on all users, there is nothing remarkable. However the "in-segment" users, those users who have rated the segment movies highly and presumably are desirable customers for the attacker are impacted significantly for both algorithms. Although item-based is less affected than user-based, a .6 prediction shift for a 1% attack is still a significant bias.

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



(a) 1% attack comparison



(b) PLSA comparison against segment attack

Figure 3: Prediction shift comparisons

5.2 PLSA Comparison

Based on these results, additional model based collaborative recommendation techniques were examined in an effort to identify a prediction scheme that would provide further robustness to these types of attacks. From this exploration, we discovered PLSA recommendation had significant robustness without compromising the quality of the predictions. For the PLSA experiments, we employ a user segment size of 30 which was chosen for its pre-attack prediction accuracy. Although a larger segment size did result in improved MAE, 30 seems to be the point of diminishing returns for our dataset. Larger segment sizes require much greater processing time in order to build a model, with marginal improvement for our purposes. For PLSA, optimal results were obtained using $k = 10$ for the neighborhood size with neighbors with a similarity score less than 0.1 filtered out.

Figure 3(b) depicts the "in-segment" prediction shift for the Horror segment attack across various attack sizes. Clearly, the attack is extremely effective against the k-NN algorithm as noted previously. By contrast, both item-based and PLSA are less affected by the segment attack, but the strength of the PLSA algorithm is shown in its additional robustness at small attack sizes. The PLSA algorithm has the additional benefit of stabilizing the prediction shift for attack sizes beyond 5%, while the item-based algorithm continues to grow logarithmically.

6 Attack Profile Classification

In this section, we present our approach to attack detection and response based on profile classification. Prior work in detecting attacks in collaborative filtering systems have mainly focused on ad hoc algorithms for identifying basic attack models such as the random attack [13]. In contrast, we have proposed an alternate technique based on more traditional supervised learning. We show that a kNN classifier, using classification attributes introduced in our prior work, and built on a training set created through injecting system data with a mix of attack profiles, can be applied to unseen segment attack data with impressive results [14, 15]. For this approach, a user's profile is examined and based on characteristics of the profile, the entry is given a classification as either authentic or attack and would

subsequently be eliminated from consideration in collaborative filtering.

6.1 Attributes for Detection

In prior work we have identified a number of attributes that are informative at detecting profile injection attacks [14, 15]. We use three classes of detection attributes: *Generic Attributes* – attributes designed to identify abnormalities in the overall distribution of a profile; *Model-based Attributes* – attributes designed to identify similarities to known attack models; and *Intra-profile Attributes* – attributes designed to identify suspicious trends across profiles.

The hypothesis behind using generic attributes is based on the expectation that the overall signature of attack profiles differs from authentic profiles. This difference comes from two sources: the rating given the target item, and the distribution of ratings among the filler items. As many researchers in the area have theorized [3, 13, 4, 7], it is unlikely for an attacker to have complete knowledge of the ratings in a real system. As a result, generated profiles often deviate from rating patterns seen for authentic users. An attribute that captures these anomalies is likely to be informative in identifying reduced knowledge attack profiles.

In addition to the generic attributes described in our previous work, we have also included an attribute we have termed *Weighted Deviation from Mean Agreement* (WDMA) [14, 15]. This attribute, designed to help identify anomalies, is an extension of the Rating Deviation from Mean Agreement introduced by Chirita *et al.* [13]. The WDMA attribute places a high weight on rating deviations for sparse items. The WDMA attribute can be computed in the following way:

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

Where U is the universe of all users u; let P_u be a profile for user u, consisting of a set of ratings $r_{u,i}$ for some items i in the universe of items to be rated; let n_u be the size of this profile in terms of the numbers of ratings; and let l_i be the number of ratings provided for item i by all users, and $\overline{r_i}$ be the average of these ratings.

In our experiments, we have found that the generic attributes are insufficient for distinguishing attack profiles from eccentric but authentic profiles. This is especially true when the profiles are small, containing few filler items. Model-specific attributes are those that aim to recognize the distinctive signature of a particular attack model. Our detection model discovers partitions of each profile that maximize its similarity to the attack model and then generates detection attributes based on the rating distribution in these partitions [14, 15].

All of the attributes thus far have concentrated on inter-profile statistics; target focus, however, concentrates on intra-profile statistics. The goal is to use the fact that an attacker often must introduce many profiles in order to achieve their desired bias. We therefore examine the density of target items across profiles. One of the advantages of the partitioning associated with the model-based attributes described above is that a set of suspected targets is identified for each profile. For our *Target Model Focus* attribute (TMF), we calculate the degree to which the partitioning of a given profile focuses on items common to other attack partitions, and therefore measures a consensus of suspicion regarding each profile.

6.2 Experimental Results of Detection

For our detection experiments, we used the same Movie-Lens 100K dataset used in Section 5. To minimize over-training, the dataset was split into 2 partitions. The first partition was made a training set, while the second was used for testing and was unseen during training. The training data was created by inserting a mix of different attacks including average, random, and segment attacks at various filler sizes that ranged from 3% to 100%. Once again to minimize over-training, a completely separate segment was used for training and testing. The kNN algorithm was then applied by comparing a profile to be classified to the profiles in the training data to create a binary profile classifier that output either *authentic* or *attack* using Weka [16]. Figure 4 depicts the average confusion matrix for our detection algorithm averaged over filler sizes of 3%, 5%, 10%, 20%, 40%, 60%, 80%, and 100% for a 1% attack. As is apparent from the confusion matrices, the detection capabilities of the

Random Attack		
authentic	**attack**	
927.14	15.86	**authentic**
0.12	8.88	**attack**

Average Attack		
authentic	**attack**	
925.35	17.65	**authentic**
0.62	8.38	**attack**

Segment Attack		
authentic	**attack**	
922.67	20.34	**authentic**
0.34	8.66	**attack**

Figure 4: Confusion matrices for 1% push attacks averaged across all filler sizes.

combined attributes of our previous work and those discussed in this paper hold significant promise for increasing robustness by eliminating the effects of attack profiles of multiple attacks. It is also important to note, few authentic users were misclassified as attacks which if excluded from collaborative filtering could otherwise reduce the system's accuracy [14].

7 Conclusions

The open and interactive nature of collaborative filtering is both a source of strength and vulnerability for recommender systems. Biased profile data can easily sway the recommendations of a collaborative system toward inaccurate results that serve the attacker's ends. Previous research has shown both user-based and item-based algorithms to be vulnerable to the segment attack. We have introduced a more robust recommendation algorithm based on PLSA. In addition, we have demonstrated that a classifier learning approach can accurately distinguish attack profiles from real users, and can limit the damage caused by attacks.

References

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

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

[3] S. Lam, J. Reidl, Shilling recommender systems for fun and profit, in: Proceedings of the 13th International WWW Conference, New York, 2004.

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

[5] J. Herlocker, J. Konstan, A. Borchers, J. Riedl, An algorithmic framework for performing collaborative filtering, in: Proceedings of the 22nd ACM Conference on Research and Development in Information Retrieval (SIGIR'99), Berkeley, CA, 1999.

[6] R. Burke, B. Mobasher, C. Williams, R. Bhaumik, Segment-based injection attacks against collaborative filtering recommender systems, in: Proceedings of the International Conference on Data Mining (ICDM 2005), Houston, 2005.

[7] B. Mobasher, R. Burke, R. Bhaumik, C. Williams, Effective attack models for shilling item-based collaborative filtering systems, in: Proceedings of the 2005 WebKDD Workshop, held in conjuction with ACM SIGKDD'2005, Chicago, Illinois, 2005.

[8] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International World Wide Web Conference, Hong Kong, 2001.

[9] T. Hofmann, Probabilistic latent semantic analysis, in: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, 1999.

[10] X. Jin, Y. Zhou, B. Mobasher, Web usage mining based on probabilistic latent semantic analysis, in: Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'04), Seattle, Washington, 2004.

[11] X. Jin, Y. Zhou, B. Mobasher, A unified approach to personalization based on probabilistic latent semantic models of web usage and content, in: Proceedings of the AAAI 2004 Workshop on Semantic Web Personalization (SWP'04), San Jose, California, 2004.

[12] J.Herlocker, J. Konstan, L. G. Tervin, J. Riedl, Evaluating collaborative filtering recommender systems, ACM Transactions on Information Systems 22 (1) (2004) 5–53.

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

[14] R. Burke, B. Mobasher, C. Williams, R. Bhaumik, Detecting profile injection attacks in collaborative recommender systems, in: To appear in Proceedings of the IEEE Joint Conference on E-Commerce Technology and Enterprise Computing, E-Commerce and E-Services (CEC/EEE 2006), Palo Alto, CA, 2006.

[15] B. Mobasher, R. Burke, C. Williams, R. Bhaumik, Analysis and detection of segment-focused attacks against collaborative recommendation, in: To appear in Lecture Notes in Computer Science: Proceedings of the 2005 WebKDD Workshop, Springer, 2006.

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