

Unsupervised Retrieval of Attack Profiles in Collaborative Recommender Systems*

Kenneth Bryan, Michael O'Mahony, Pádraig Cunningham
University College Dublin

Technical Report UCD-CSI-2008-03
April, 2008

Abstract

Trust, reputation and recommendation are key components of successful ecommerce systems. However, ecommerce systems are also vulnerable in this respect because there are opportunities for sellers to gain advantage through manipulation of reputation and recommendation. One such vulnerability is the use of fraudulent user profiles to boost (or damage) the ratings of items in an online recommender system. In this paper we cast this problem as a problem of detecting anomalous structure in network analysis and propose a novel mechanism for detecting this anomalous structure. We present an evaluation that shows that this approach is effective at uncovering the types of recommender systems attack described in the literature.

1 Introduction

In recent years there has been significant research interest in the problem of detecting attempts to inject bias into recommender systems (Burke et al., 2006; Lam and Riedl, 2004; O'Mahony et al., 2004, 2005). Many e-commerce systems that make use of endorsement or positive ratings to rank products or sellers are vulnerable to the creation of fake profiles that bias the recommendation process. In this paper we cast this as the problem of detecting anomalous structure in network data and consider what can be borrowed from related research in this wider area to address the problem.

A popular theme in data analysis is the study of collections of entities and the links between these entities. In social network analysis the entities may be individuals and links are relationships between them. There is also a lot research on network analysis in bioinformatics where the nodes can represent genes or proteins and the links indicate interactions between them. One of the dominant research themes in biological network analysis is the discovery of network *motifs* that are in some way anomalous or remarkable (Milo et al., 2002; Saul and Filkov, 2007). A motif is significant in a network if the probability of the observed incidence of the motif occurring by chance is less than some cutoff value, typically 0.01. Such motifs are worth studying as they can reveal something about the structural design principles of the network.

In social network analysis Boykin and Roychowdhury (2005) have shown that a similar analysis of the structure of a network created from the headers of messages in a user's email inbox can help identify spam. In this network the vertices are users (email addresses) and the edges are taken from the 'To' and 'Cc' fields in messages. The key insight in this work is that the network structure around spammers is different to that around legitimate users. There will be few links between addresses occurring in the

*The work was part supported by Science Foundation Ireland Grant No. No. 05/IN.1/I24

‘To’ and ‘Cc’ fields of spam email whereas there will be frequent triangular structures around legitimate users. In the language of biological network analysis we can say that there are characteristic network motifs associated with spam email. Boykin and Roychowdhury analyse a clustering coefficient which quantifies the proportion of pairs of edges that actually form triangles from all possible connected pairs of edges. Because co-recipients of spam email will often not know each other, there will be very few links between these co-recipients and thus very few triangles. They show that this clustering coefficient is effective at identifying spam. Zinman and Donath (2007) discuss how link analysis such as this can be applied to assess requests to join a person’s social network in a social networking service such as MySpace.

The starting point for the research reported in this paper is the observation that the task of identifying attack profiles in recommender systems is similar to the task of identifying biclusters in gene microarray expression data. The objective in biclustering gene expression data is to discover subsets of genes that are correlated across a subset of samples (Cheng and Church, 2000). The research on attack profiles in recommender systems suggests that some important types of attack have a similar structure in that a set of attack profiles will be correlated across a subset of items. Our preliminary evaluation found that that *bandwagon attacks* (see section 2) could readily be identified by biclustering. However, biclustering was less effective at uncovering other types of attack. We found though that the H_v -score that is an effective metric for biclustering expression data (Bryan and Cunningham, 2006) is also effective for uncovering a range of attack strategies. The main contribution of this paper is a comprehensive evaluation of the H_v -score for attack detection.

This approach has an advantage over alternative supervised strategies (Burke et al., 2006) in that, by being unsupervised, it simply uncovers structure that is *anomalous* and thus is not restricted to discovering attack types considered at design time.

The paper proceeds with an overview of the characteristics of recommender system attacks in section 2. In section 3 we review strategies for detecting attacks and present our H_v -score strategy. The evaluation methodology is described in section 4 and the results of the evaluation are presented in section 5. The paper concludes with a summary and some plans for future work in section 6.

2 Attacks on Recommender Systems

People are frequently required to make choices about items without having a significant knowledge of the range of choices available. Consequently, we often seek recommendations from others relating to which movies to see, which books to read or which car to buy etc. Collaborative recommendation algorithms operate in a similar fashion and can be used to filter information and recommend personalised content that satisfy the particular needs and tastes of individual users (Resnick et al., 1994). These algorithms have been successfully employed in many on-line settings and work by gathering preference data and opinions from users and using this information to generate recommendations for others.

While people are relatively adept at assessing the reliability of friends and associates and valuing recommendations from such sources accordingly, it is much more difficult to make judgments concerning users of online environments given their anonymous or pseudo-anonymous nature. Since it is practically impossible to determine in advance the motivations and integrity of those who use online systems, there is no guarantee that the preferences expressed for items reflect the true opinions of users.

In addition, it is often feasible to create a number of identities within a single system and thus the potential for *shilling attacks* or *profile injection attacks* to occur exists (Lam and Riedl, 2004; O’Mahony et al., 2004). These attacks involve the creation of multiple attack profiles which are typically designed to reflect the true preferences of genuine users for certain items, while the target

item is assigned a biased rating with the intention of promoting or demoting recommendations made for the item in question. Such attacks are referred to as *product push* and *product nuke* attacks, respectively. Since it has been demonstrated that the presence of even small quantities of attack profiles can significantly bias recommendations (O’Mahony et al., 2005), it is vital that online systems are protected against these kinds of attack.

2.1 Attack Strategies

A number of attack models have been proposed in the literature; in this paper, we will evaluate our detection algorithm in the context of the widely studied random, average and bandwagon attacks (Burke et al., 2006).

2.1.1 Standard Attacks

For each of the attack models, it is assumed the the objective of the attack is to push or nuke the recommendations that are made for one particular target item, i_t . This item is always included in attack profiles and is assigned the maximum (r_{max}) or minimum rating r_{min} for product push or nuke attacks, respectively. The remaining items for attacks profiles are selected for the different attack models as follows.

Random Attack. In this attack, a subset of *filler* items are simply selected uniformly at random from the set of items present in the system. These items are rated randomly according to a normal distribution with mean equal to the average rating of all items in the system and standard deviation equal to the standard deviation across all items. The advantage of this attack is the low degree of domain knowledge required, given that the overall mean and spread of ratings for many domains can be estimated or mined without too much difficulty.

Average Attack. As with the random attack, filler items are selected uniformly at random from the system item set. Ratings for filler items, however, are assigned based on a more specific knowledge of the domain. In this case, filler items are rated randomly on a normal distribution with mean equal to the average rating of the item being rated and with the standard deviation computed as above. This attack has been found to outperform the random attack in the literature (Burke et al., 2006). Note, however, that the domain knowledge required to implement this attack is significant and thus may be unfeasible to implement in practice.

Bandwagon Attack. This attack can be viewed as an extension to the random attack, where an additional set of *selected* items are also included in attack profiles. These items are picked due to their popularity in the domain, both in terms of the number of ratings assigned to them by genuine users and also because they are generally liked. In practice, these items can easily be identified by examining box-office returns and best-seller lists, etc. In the attack profiles, selected items are assigned ratings of r_{max} , thereby ensuring high degrees of similarity between attack and genuine profiles. As a result of including these items, it has been demonstrated that the bandwagon attack significantly outperforms both random and average attacks.

2.1.2 Obfuscated Attacks

Attack detection can be viewed as a 2-player game between attackers attempting to create more effective attack models and system administrators trying to devise new attack detection algorithms. As detection algorithms become adept at identifying profiles created according to particular attack

models, attackers may attempt to obfuscate attack signatures in order to render them less visible to detection algorithms. Many approaches to obfuscation are possible; in this paper, we consider the models proposed in (Williams et al., 2006) which can be applied to the three attacks described above.

Noise Injection. Noise is added to ratings according to a standard normal distribution multiplied by a constant, α , which governs the degree of obfuscation obtained. We set $\alpha = 0.2$ in our evaluations and we apply noise to all filler and selected items.

User Shifting. This approach involves shifting the ratings of a subset of items within a profile by a constant amount. Shifts can take the positive or negative form, where the amount of shift for each profile is governed by a standard normally distributed random number. In the evaluations, all filler and selected items in attack profiles are shifted.

Target Shifting. As stated in Section 2.1.1, target items in attack profiles all receive ratings of either r_{max} or r_{min} depending on attack intent. Thus, all attack profiles exhibit a particular signature in this regard. Target shifting involves setting the rating of the target item in a percentage of attack profiles to $r_{max} - 1$ for push attacks and to $r_{min} + 1$ for nuke attacks. Where applied in our evaluations, we choose to target shift 50% of attack profiles inserted into the system.

3 Attack Detection

Given the vulnerability of collaborative recommender systems to attack as described above, research activity in the field is now well-focused on the robustness issue. We begin this section with a review of some of the recent techniques that have been proposed to defend collaborative recommender systems against attack.

In (O’Mahony et al., 2006), signal processing theory was applied to the problem of attack detection. Attack data, created according to a set of specific attack models, is deliberately inserted into a system in order to model the predictive accuracy distributions of known genuine and attack ratings, thereby enabling the selection of thresholds that optimised hit and false alarm rates for such attacks.

In (Mobasher et al., 2007), a number of statistical measures to detect attack data were proposed to detect attack profiles. These include generic and model-specific measures, where the latter are derived from the characteristics of a set of known attack models. Classifiers based on these attributes were built using supervised learning methods which are trained to discriminate between genuine and attack profiles. As with the previous technique, classification is tuned to the specific attack models under consideration, and consequently systems employing these approaches would be susceptible to new or unknown attack models that may be developed.

An unsupervised classification algorithm was introduced in (Mehta et al., 2007) which considers the combined effect of attack profiles, with the expectation that the correlation between such profiles would be high relative to that between genuine profiles. The proposed algorithm, based on principal component analysis, achieved favourable attack detection performance when compared to (Mobasher et al., 2007; O’Mahony et al., 2006).

3.1 Standard Detection Metrics

In this section we briefly describe some of the statistical measures which have been proposed in the literature to detect attack profiles. The motivation behind such measures is that the statistical signature for attack and genuine profiles will differ and thus offer a basis on which to perform classification. We will compare the performance of the measures set out below to our proposed metric in Section 5.2.

Rating Deviation from Mean Agreement (RDMA). This measure was introduced by Chirita et al. (2005) and measures a user’s rating disagreement with other users in the system, weighted by the inverse number of ratings assigned to the user’s rated items. It is defined as:

$$\text{RDMA}_u = \frac{1}{|N_u|} \sum_{i=0} \frac{|r_{u,i} - \bar{r}_i|}{|R_{U,i}|}, \quad (1)$$

where N_u is the number of items rated by user u , $r_{u,i}$ is the rating assigned by user u to item i and U_i is the set of users who have rated item i .

Weighted Degree of Agreement (WDA). Derived from RDMA, WDA is designed to capture the sum of the differences of the user’s ratings from the item’s average rating, divided by each item’s rating frequency (Mobasher et al., 2007). The measure is defined as:

$$\text{WDA}_u = \sum_{i=0} \frac{|r_{u,i} - \bar{r}_i|}{|R_{U,i}|}, \quad (2)$$

Weighted Deviation from Mean Agreement (WDMA). As before, this measure is derived from RDMA and places a higher weight on rating deviations that are observed for sparse items by squaring the denominator inside the sum (Mobasher et al., 2007). It is given by:

$$\text{WDMA}_u = \frac{1}{|N_u|} \sum_{i=0} \frac{|r_{u,i} - \bar{r}_i|}{|R_{U,i}|^2} \quad (3)$$

Length Variance (LengthVar). This measure is designed to detect profiles consisting of unusually small or large numbers of items. The ability to accurately detect large-sized profiles can be seen to be of particular importance, given that such profiles would likely exert a significant influence on recommendations. This measure is defined as follows.

$$\text{LengthVar}_u = \frac{|l_u - \bar{l}|}{\sum_{k \in U} (l_k - \bar{l})^2}, \quad (4)$$

where U is the set of all users, l_u is the length of profile u and \bar{l} is the average profile length in the system. Note that, of all the measures described in this section, WDMA was found to provide the highest information gain (Mobasher et al., 2007).

3.2 Residue Based Metrics

The metrics introduced in this study have their origins within bioinformatics, namely the gene expression analysis domain. The *mean square residue* or *H-score* was introduced by Cheng and Church to aid location of *biclusters*, i.e. subsets of genes correlated over a subset of experimental conditions, in an attempt to better model the gene functional modules within expression data (Cheng and Church, 2000). The *H-score* is a cumulative metric based on the *residue* score that can be assigned to an entry in a matrix:

$$R(a_{ij}) = a_{ij} - a_{Ij} - a_{iJ} + a_{IJ} \quad (5)$$

where a_{ij} is the entry at position ij in the sub-matrix (I, J) , a_{iJ} is the mean of the i th row, a_{Ij} is the mean of the j th column and a_{IJ} mean of the whole matrix.

The mean squared residue or H -score for the matrix (I, J) is given by:

$$H(I, J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} (R(a_{ij}))^2 \quad (6)$$

Later, biases within this metric were identified (Aguilar-Ruiz, 2005) and subsequently remedied (Bryan and Cunningham, 2006) giving rise to *variance adjusted* mean square residue or H_v -score:

$$H_v(I, J) = \frac{\sum_{i \in I, j \in J} (R(a_{ij}))^2}{\sum_{i \in I, j \in J} (a_{ij} - a_{iJ})^2} \quad (7)$$

where a_{ij} is the entry at position ij in the matrix (I, J) , a_{iJ} is the mean of the i th row. The added row variance as the denominator compensates for the bias inherent in the mean squared residue.

The H_v has been used to detect highly correlated sub-matrices or *biclusters* within gene expression data (Bryan and Cunningham, 2006, 2007). An analogous problem within the area of collaborative recommendation is that of the detection of certain types of fraudulent *profiles* that exhibit a high correlation over a subset of items. These types of attack are termed *bandwagon attacks* and occur when a standard *random attack* is augmented by the addition of k selected popular items, each of which are assigned the mean rating for that item. As the number of random filler items increases however the intra-correlation of the profiles is reduced and these profiles become less detectable by standard bicluster analysis. *Random* and *Average* attack profiles, mentioned in Section 2, also show less intra-correlation and exhibit an anomalous structure in the context of the larger data set, i.e. all genuine users.

Interestingly, further properties of the H_v -score metric may be used to aid detection and retrieval of such attack profiles. Although the H_v -score may also be used to assess how well each row and column fit into a data matrix. This is achieved in practice by analysing the *partial* H_v -scores for the rows/columns in the matrix. In general, outlying or anomalous rows in a matrix incur higher H_v -scores than well fitting rows. As a result this aspect of the H_v -score would seem to suit the profile injection attack detection problem within the collaborative recommendation domain.

In the context of anomalous profile detection, using the notation consistent with Section 3.1, the partial H_v -score for user u is given by:

$$H_v(u) = \frac{\sum_{i \in I} (r_{ui} - r_{U_i} - r_{uI} + r_{UI})^2}{\sum_{j \in J} (r_{uj} - r_{uI})^2} \quad (8)$$

where r_{ui} is the rating that user u has assigned to item i , r_{U_i} is the average rating that item i has received from all users U , r_{uI} is the average rating that user u has given to all items, and r_{UI} is the average rating the data matrix.

4 UnRAP: Unsupervised Retrieval of Attack Profiles

Step 1: User Profile Scoring using the H_v -score: The first step in the UnRAP algorithm involves assigning H_v -scores to all users in the database (see also pseudo-code in Figure 1). Due to the large number of null values (rating zero) in the sparse recommender system data the standard scoring approach utilized in the gene expression data must be adjusted. The row, column and matrix means

The UnRAP Algorithm:

```
Step 1: ScoreUserProfiles(data matrix)
  for (each user in data matrix) do
    calculate  $H_v(u)$ 
  end for
  sort users based on  $H_v$ -scores
  Return sorted list of users

Step 2: RetrieveAttackedItem(sorted user list)
  for (top 10 users in sorted list) do
    find item with max rating deviation,  $i_{max}$ 
  end for
   $i_t \leftarrow i_{max}$ 
  Return  $i_t$  with +/- (push/nuke) verdict

Step 3: RetrieveAttackProfiles(sorted user list,  $i_t$ )
  for (each window of 10 users in sorted list) do
    calculate deviation of  $i_t$ 
    if (push AND deviation of  $i_t \leq 0$ ) then
       $stopping\ point \leftarrow$  current window
    else if (nuke AND deviation of  $i_t \geq 0$ ) then
       $stopping\ point \leftarrow$  current window
    end if
    slide window by one user
  end for

  for (each user in the sorted list <  $stopPoint$ ) do
    if ( $i_t$  is not rated) then
      remove user (genuine user)
    else if (push AND  $i_t$  rating < user's mean rating) then
      remove user (genuine user)
    else if (nuke AND  $i_t$  rating > user's mean rating) then
      remove user (genuine user)
    end if
  end for
  Return reduced user list (attack profiles)
```

Figure 1: The pseudo-code for the UnRAP algorithm.

are calculated as normal, i.e. over all entries (including null values), however only the *residues* for each non-null value are summed to give the *sum squared residue* for the user. In general anomalous profiles, i.e. profiles that do not fit well into the data matrix, receive a higher H_v -score than well fitting profiles. Therefore after sorting based on the partial H_v -score the attack profiles tend to cluster in the upper range.

Retrieval of Target Item: In the second step the top- n highest scoring users from the sorted user list are examined to identify the target item. In all cases, regardless of attack or filler size, we examine the top 10 users ($n=10$). As the upper range tends to be enriched for attack profiles the target item (i_t) can generally be detected by retrieving the item which deviates most from the mean user rating. In practice the profile mean is subtracted from each item rating and summed over the 10 users. The item with the largest consistent deviation from its mean over the top 10 profiles (positive deviation in the case of a pushed item and negative deviation in the case of a nuked item) is selected as target item (i_t).

Retrieval of Attack Profiles: In Step 3 of the UnRAP algorithm a *sliding window* (of 10 users in size) is passed along the sorted user list, see Figure 2. In each iteration the window is shifted by one user and the sum of the rating deviation for the target item over the current 10 users is calculated. At some point the sliding window moves from a region enriched with attack profiles to a region enriched

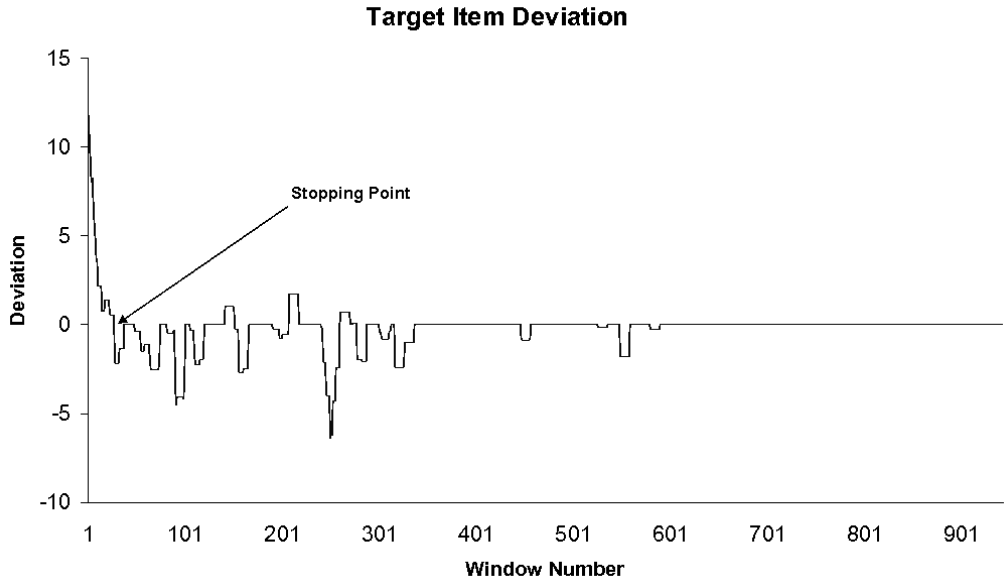


Figure 2: The sliding window assessment in Step 2.

for normal users. This point is signified by a sharp change in the rating deviation for the target item. In practice the *stopping point* is reached when the deviation of the target item reaches or exceeds zero (≤ 0 for a push and ≥ 0 for a nuke attack). A final filtering is then carried out on this reduced set in which individual users that have either no rating for the target item or a rating deviation in the opposite direction from the attack (i.e. if the rating at i_t is less than the user mean for pushed target item than it is removed). After this final filtering the remaining user set is returned as attack profiles for the target item i_t .

5 Evaluation

In the first part of our evaluation we compare the H_v -score against existing generic profile attributes as reported in (Burke et al., 2006) and described in Section 3.1. In Section 5.3 we investigate the ability of UnRAP to detect attack profiles employing various strategies and intents over a range of attack and filler sizes.

5.1 Data

In our evaluations we used the publicly available Movie-Lens 100K dataset¹. The Movie-Lens dataset consists of 100,000 ratings on 1682 movies by 943 users. Ratings are integer values between 1 and 5 where 1 is the lowest (disliked) and 5 is the highest (most liked). Our data includes all the users who have rated at least 20 movies. In this study we generate both the standard and obfuscated attack strategies (*random*, *average* and *bandwagon*) described in Section 2.1.1. Attacks of sizes 1%, 2%, 5% and 10% and filler sizes of 1%, 3%, 5%, 10%, 25%, 40% and 60% are generated. All reported results are based on the mean values over 100 randomly selected target items.

¹<http://www.cs.umn.edu/research/GroupLens/data/>

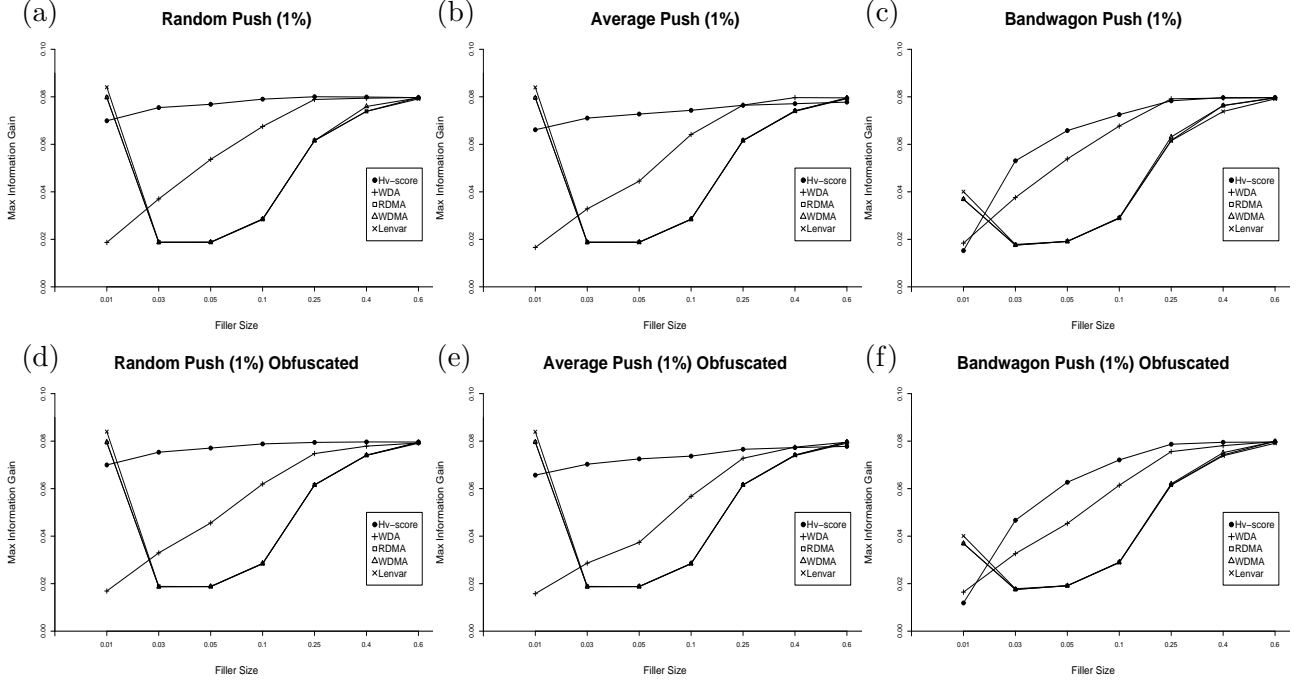


Figure 3: Metric Evaluation - Graphs showing the maximum information gain achievable for each filler size for 1% Standard (a) *Random*, (b) *Average* and (c) *Bandwagon* push attacks after scoring all user profiles with the H_v -score, RDMA, WDMA, WDA and Length variance. Equivalent graphs for Obfuscated attacks are given in (d), (e) and (f). Similar relative performance is observed over increased attack size and nuke attack.

5.2 Evaluation of the H_v -score Metric

Recently Burke *et al.* carried out an extensive evaluation of generic ‘attack detection attributes’ (Burke et al., 2006). They found that RDMA, WDMA, WDA and Length variance, see Section 2, were the most distinguishing generic attributes of attack profiles. We compare the H_v -score metric to these attributes and, like Burke *et al.*, use *information gain* as an evaluation criterion. In this procedure all user profiles are first scored with each metric. Subsequently, the information gain for the ‘best split’ between attack profiles and genuine users is calculated. All result presented in this section are mean results over 100 attacks on randomly selected items.

In Figure 3 we present the comparison of metrics for standard *Random*, *Average* and *Bandwagon* attack strategies of size 1% over several filler sizes. We can see that H_v -score performs well against the RDMA, WDMA, WDA and Length variance profile attributes over most filler sizes. The one exception seems to be the lowest filler size of 1% in which the H_v -score is outperformed by the other metrics. Importantly, the H_v -score metric performs well over the mid-range of filler size, where other metrics experience a dip in performance. In Figure 3 (d), (e) and (f) we present the comparison for obfuscated versions of the above attacks. We can see that, despite obfuscation, similar performance is achieved.

Burke *et al.* incorporated their metrics as a ‘generic attributes’ part of a fully supervised attack detection approach. Given the results achieved in this section it is clear that such an approach would benefit from an additional H_v -score feature. In the next section we evaluate the performance of the UnRAP algorithm which incorporates the H_v -score as its principal metric.

Table 1: Precision (P) and recall (R) for *Random Push* and *Nuke* attacks of varying sizes and filler sizes. Underlined recall values show improvement upon previously reported results [7].

Filler		1%		3%		5%		10%		25%		40%		60%	
Push															
Attack size	P	R	P	R	P	R	P	R	P	R	P	R	P	R	
1%	0.89	<u>0.99</u>	0.90	<u>1.00</u>	0.90	<u>1.00</u>	0.90	<u>1.00</u>	0.90	<u>1.00</u>	0.90	<u>1.00</u>	0.90	<u>1.00</u>	
2%	0.93	<u>0.99</u>	0.94	<u>1.00</u>	0.94	<u>1.00</u>	0.94	<u>1.00</u>	0.94	<u>1.00</u>	0.94	<u>1.00</u>	0.94	<u>1.00</u>	
5%	0.96	<u>0.99</u>	0.97	<u>1.00</u>	0.97	<u>1.00</u>	0.97	<u>1.00</u>	0.97	<u>1.00</u>	0.97	<u>1.00</u>	0.97	<u>1.00</u>	
10%	0.98	<u>0.99</u>	0.98	<u>1.00</u>	0.99	<u>1.00</u>	0.99	<u>1.00</u>	0.99	<u>1.00</u>	0.99	<u>1.00</u>	0.99	<u>1.00</u>	
Nuke															
1%	0.41	0.99	0.46	1.00	0.47	1.00	0.49	1.00	0.49	1.00	0.49	1.00	0.49	1.00	
2%	0.54	0.99	0.60	1.00	0.62	1.00	0.63	1.00	0.64	1.00	0.64	1.00	0.64	1.00	
5%	0.71	0.99	0.77	1.00	0.79	1.00	0.80	1.00	0.81	1.00	0.82	1.00	0.81	1.00	
10%	0.82	0.99	0.86	1.00	0.88	1.00	0.89	1.00	0.89	1.00	0.90	1.00	0.90	1.00	

Table 2: Precision (P) and recall (R) for *Average Push* and *Nuke* attacks of varying sizes and filler sizes. Underlined recall values show improvement upon previously reported results [7].

Filler		1%		3%		5%		10%		25%		40%		60%	
Push															
Attack size	P	R	P	R	P	R	P	R	P	R	P	R	P	R	
1%	0.89	<u>0.98</u>	0.90	1.00	0.90	<u>1.00</u>	0.90	<u>1.00</u>	0.90	<u>1.00</u>	0.90	<u>1.00</u>	0.88	<u>1.00</u>	
2%	0.92	<u>0.97</u>	0.93	<u>1.00</u>	0.94	<u>1.00</u>	0.94	<u>1.00</u>	0.94	<u>1.00</u>	0.94	<u>1.00</u>	0.93	<u>1.00</u>	
5%	0.95	<u>0.97</u>	0.97	<u>1.00</u>	0.97	<u>1.00</u>	0.97	<u>1.00</u>	0.97	<u>1.00</u>	0.98	<u>1.00</u>	0.98	<u>1.00</u>	
10%	0.97	<u>0.98</u>	0.98	<u>1.00</u>	0.98	<u>1.00</u>	0.98	<u>1.00</u>	0.99	<u>1.00</u>	0.99	<u>1.00</u>	0.99	<u>1.00</u>	
Nuke															
1%	0.37	0.96	0.41	1.00	0.43	1.00	0.45	1.00	0.47	1.00	0.47	1.00	0.47	1.00	
2%	0.51	0.97	0.55	1.00	0.57	1.00	0.60	1.00	0.62	1.00	0.62	1.00	0.63	1.00	
5%	0.67	0.97	0.74	1.00	0.76	1.00	0.77	1.00	0.80	1.00	0.80	1.00	0.81	1.00	
10%	0.76	0.98	0.84	1.00	0.85	1.00	0.87	1.00	0.89	1.00	0.89	1.00	0.89	1.00	

Table 3: Precision (P) and recall (R) for *Bandwagon Push* and *Nuke* attacks of varying sizes and filler sizes. Underlined recall values show improvement upon previously reported results [7].

Filler		1%		3%		5%		10%		25%		40%		60%	
Push															
Attack size	P	R	P	R	P	R	P	R	P	R	P	R	P	R	
1%	0.00	0.00	0.42	0.82	0.75	<u>0.99</u>	0.67	<u>1.00</u>	0.70	<u>1.00</u>	0.70	<u>1.00</u>	0.70	<u>1.00</u>	
2%	0.00	0.00	0.57	<u>0.89</u>	0.83	<u>1.00</u>	0.84	<u>1.00</u>	0.82	<u>1.00</u>	0.79	<u>1.00</u>	0.79	<u>1.00</u>	
5%	0.00	0.00	0.75	0.87	0.87	<u>1.00</u>	0.91	<u>1.00</u>	0.93	<u>1.00</u>	0.92	<u>1.00</u>	0.91	<u>1.00</u>	
10%	0.00	0.00	0.74	0.67	0.91	<u>1.00</u>	0.97	<u>1.00</u>	0.97	<u>1.00</u>	0.97	<u>1.00</u>	0.96	<u>1.00</u>	
Nuke															
1%	0.00	0.00	0.26	0.80	0.36	1.00	0.43	1.00	0.48	1.00	0.49	1.00	0.49	1.00	
2%	0.00	0.00	0.35	0.92	0.50	1.00	0.57	1.00	0.63	1.00	0.64	1.00	0.64	1.00	
5%	0.00	0.00	0.50	0.94	0.67	1.00	0.77	1.00	0.80	1.00	0.81	1.00	0.82	1.00	
10%	0.00	0.00	0.56	0.96	0.77	1.00	0.86	1.00	0.89	1.00	0.90	1.00	0.90	1.00	

5.3 Evaluation of the UnRAP Algorithm

In this section we examine the efficacy of the UnRAP algorithm in detection of both standard and obfuscated *Random*, *Average* and *Bandwagon* attacks of varying sizes, intents and filler sizes. All results represent average performance over 100 randomly selected items from the Movie-Lens database. In Tables 1, 2 and 3 we present results in terms of the precision (P) and recall (R) of attack profiles for standard *Random*, *Average* and *Bandwagon* attacks respectively. We examine performance over attack sizes of 1%, 2%, 5% and 10% and filler sizes of 1%, 3%, 5%, 10%, 25%, 40% and 60%. Both Push and Nuke attack intentions are investigated.

Table 4: Precision (P) and recall (R) for *Obfuscated Random Push* and *Nuke* attacks of varying sizes and filler sizes.

Filler		1%		3%		5%		10%		25%		40%		60%	
Push															
Attack size	P	R	P	R	P	R	P	R	P	R	P	R	P	R	R
1%	0.87	0.88	0.90	0.84	0.89	0.84	0.90	0.85	0.90	0.84	0.86	0.87	0.82	0.87	0.87
2%	0.93	0.83	0.93	0.84	0.93	0.84	0.93	0.85	0.94	0.83	0.93	0.84	0.95	0.86	0.86
5%	0.96	0.84	0.97	0.84	0.97	0.85	0.97	0.85	0.97	0.84	0.97	0.84	0.98	0.85	0.85
10%	0.98	0.82	0.98	0.84	0.98	0.84	0.99	0.84	0.99	0.84	0.99	0.83	0.99	0.83	0.83
Nuke															
1%	0.41	0.95	0.45	0.97	0.46	0.95	0.47	0.96	0.48	0.97	0.48	0.97	0.48	0.97	0.97
2%	0.55	0.96	0.58	0.97	0.61	0.97	0.63	0.96	0.63	0.96	0.63	0.96	0.64	0.95	0.95
5%	0.73	0.96	0.77	0.97	0.79	0.97	0.80	0.97	0.81	0.94	0.84	0.84	0.90	0.68	0.68
10%	0.81	0.95	0.86	0.97	0.87	0.97	0.89	0.94	0.93	0.72	0.97	0.41	0.99	0.24	0.24

Table 5: Precision (P) and recall (R) for *Obfuscated Average Push* and *Nuke* attacks of varying sizes and filler sizes.

Filler		1%		3%		5%		10%		25%		40%		60%	
Push															
Attack size	P	R	P	R	P	R	P	R	P	R	P	R	P	R	R
1%	0.83	0.90	0.90	0.93	0.89	0.93	0.90	0.95	0.86	0.97	0.80	0.98	0.65	0.99	0.99
2%	0.92	0.89	0.93	0.92	0.93	0.93	0.93	0.95	0.90	0.96	0.80	0.98	0.71	0.98	0.98
5%	0.95	0.88	0.97	0.93	0.97	0.94	0.97	0.94	0.96	0.93	0.95	0.94	0.86	0.98	0.98
10%	0.97	0.87	0.98	0.93	0.98	0.93	0.98	0.94	0.99	0.89	0.96	0.92	0.92	0.94	0.94
Nuke															
1%	0.37	0.90	0.41	0.95	0.43	0.96	0.45	0.95	0.46	0.96	0.51	0.95	0.52	0.96	0.96
2%	0.50	0.89	0.55	0.95	0.57	0.97	0.59	0.97	0.61	0.96	0.62	0.97	0.62	0.96	0.96
5%	0.68	0.91	0.74	0.95	0.75	0.96	0.77	0.97	0.80	0.96	0.80	0.96	0.80	0.96	0.96
10%	0.78	0.90	0.84	0.95	0.85	0.95	0.87	0.96	0.89	0.94	0.89	0.96	0.89	0.96	0.96

Table 6: Precision (P) and recall (R) for *Obfuscated Bandwagon Push* and *Nuke* attacks of varying sizes and filler sizes.

Filler		1%		3%		5%		10%		25%		40%		60%	
Push															
Attack size	P	R	P	R	P	R	P	R	P	R	P	R	P	R	R
1%	0.01	0.01	0.30	0.73	0.63	0.98	0.66	0.99	0.76	0.98	0.71	0.98	0.73	0.98	0.98
2%	0.00	0.00	0.51	0.59	0.73	0.91	0.77	0.98	0.84	0.94	0.86	0.92	0.90	0.90	0.90
5%	0.00	0.00	0.64	0.20	0.80	0.79	0.91	0.97	0.95	0.90	0.96	0.87	0.97	0.84	0.84
10%	0.00	0.00	0.64	0.10	0.86	0.70	0.96	0.94	0.98	0.86	0.98	0.85	0.99	0.83	0.83
Nuke															
1%	0.00	0.01	0.24	0.74	0.36	0.94	0.42	0.97	0.47	0.97	0.48	0.96	0.49	0.98	0.98
2%	0.00	0.00	0.33	0.81	0.48	0.94	0.57	0.97	0.63	0.97	0.65	0.97	0.70	0.98	0.98
5%	0.00	0.00	0.49	0.82	0.65	0.95	0.76	0.97	0.82	0.98	0.86	0.99	0.90	1.00	1.00
10%	0.00	0.00	0.55	0.82	0.75	0.95	0.85	0.96	0.91	0.98	0.95	1.00	0.95	1.00	1.00

Where possible, figures are compared with previously reported results. The underlined values represent improvements upon results from the *PCAVarSelect* approach developed by Mehta *et al.* to detect push attacks (Mehta et al., 2007). *PCAVarSelect* also requires the exact size of the attack to be input as a parameter and therefore requires more supervision than our UnRAP approach which retrieves attacks of various sizes without the need for a precise attack size constraint.

In Tables 4, 5 and 6 precision and recall of obfuscated attack profiles are reported. This obfuscation involves *Noise Injection*, *Target Shifting* and *User Shifting* parameters (see Section 2.1.2). Comparison with results of Mehta *et al.* is not performed as their obfuscation parameters and values were not reported. It can be seen that UnRAP performs well despite obfuscation.

From the tables it can be seen that in the case of *Bandwagon* attacks of 1% filler size UnRAP performs poorly. This is as a result of a high ratio of *selected item* ratings as compared with randomly imputed item ratings which cause these profiles to become less conspicuous. However, as these attacks are highly intra-correlated, they can be detected by standard biclustering approaches as described in Section 3.2.

In the case of push attacks, precision seems to improve as attack size increases (down the table). This may be because as an item is pushed or nuked with increasing frequency the item mean shifts (up for a push attack and down for a nuke attack) in relation to the user and database mean. This fact has ramifications for the residue of the entry, see Equations 6 and 8 in Section 3.2, making profiles more ill-fitting. This improvement is also seen in the case of nuke attacks, however overall precision is lower. An explanation for this is that in nuke attacks both the mean user rating (r_{uI}) and the mean item rating for the target item (r_{U_i}) are lower than push attacks. This serves to reduce the overall mean squared residue across the user profile, see the nominator in Equation 8, Section 3.2.

6 Conclusion

The UnRAP algorithm employs a sparse matrix variation of the H_v -score metric, developed in the area of gene expression data analysis. In this paper we have shown that this metric performs well in separating fraudulent attack profiles from genuine users. Given its performance when compared with standard metrics, the H_v -score may be used to improve performance of existing attack detection methods, whether they be fully supervised or otherwise. The UnRAP algorithm builds on the strength of the H_v -score, incorporating additional steps to maximize retrieval of attack profiles.

UnRAP shows good performance over a range of standard and obfuscated attack strategies and various attacks sizes and filler sizes. UnRAP also shows good performance against previous algorithms presented in the literature and requires less supervision than previous approaches, in terms of input constraints. Further more the generic unsupervised nature of UnRAP should mean that it is better capable of detecting future novel attack strategies that may confound supervised approaches that fit the current problem models.

Bibliography

- J. Aguilar-Ruiz. Shifting and scaling patterns from gene expression data. *Bioinformatics*, 21(20): 3840–3845, 2005.
- P. O. Boykin and V. P. Roychowdhury. Leveraging social networks to fight spam. *IEEE Computer*, 38(4):61–68, 2005.
- K. Bryan and P. Cunningham. Bottom-Up Biclustering of Expression Data. *Computational Intelligence and Bioinformatics and Computational Biology, 2006. CIBCB'06. 2006 IEEE Symposium on*, pages 1–8, 2006.
- K. Bryan and P. Cunningham. BALBOA: Extending Bicluster Analysis to Classify ORFs using Expression Data. *Bioinformatics and Bioengineering, 2007. BIBE 2007. Proceedings of the 7th IEEE International Conference on*, pages 995–1002, 2007.
- R. Burke, B. Mobasher, C. Williams, and R. Bhaumik. Classification features for attack detection in collaborative recommender systems. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 542–547, 2006.

- Y. Cheng and G. Church. Biclustering of expression data. *Proc Int Conf Intell Syst Mol Biol*, 8: 93–103, 2000.
- P. Chirita, W. Nejdl, and C. Zamfir. Preventing shilling attacks in online recommender systems. *Proceedings of the seventh ACM international workshop on Web information and data management*, pages 67–74, 2005.
- S. Lam and J. Riedl. Shilling recommender systems for fun and profit. *Proceedings of the 13th conference on World Wide Web*, pages 393–402, 2004.
- B. Mehta, T. Hofmann, and P. Fankhauser. Lies and propaganda: detecting spam users in collaborative filtering. *Proceedings of the 12th international conference on Intelligent user interfaces*, pages 14–21, 2007.
- R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network Motifs: Simple Building Blocks of Complex Networks, 2002.
- B. Mobasher, R. Burke, R. Bhaumik, and C. Williams. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. *ACM Transactions on Internet Technology*, 7(4), 2007.
- M. P. O’Mahony, N. J. Hurley, N. Kushmerick, and G. C. M. Silvestre. Collaborative recommendation: A robustness analysis. *ACM Trans. Internet Techn.*, 4(4):344–377, 2004.
- M. P. O’Mahony, N. J. Hurley, and G. C. M. Silvestre. Recommender systems: Attack types and strategies. In M. M. Veloso and S. Kambhampati, editors, *AAAI*, pages 334–339. AAAI Press / The MIT Press, 2005. ISBN 1-57735-236-X.
- M. P. O’Mahony, N. J. Hurley, and G. C. M. Silvestre. Detecting noise in recommender system databases. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI’06)*, pages 109–115, Jan 29–Feb 1 2006.
- P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW’94)*, pages 175–186, October 22–26 1994.
- Z. Saul and V. Filkov. Exploring biological network structure using exponential random graph models. *Bioinformatics*, 23(19):2604, 2007.
- C. Williams, B. Mobasher, R. Burke, J. Sandvig, and R. Bhaumik. Detection of Obfuscated Attacks in Collaborative Recommender Systems. *Proceedings of the ECAI06 workshop on recommender systems, Held at the 17th European Conference on Artificial Intelligence (ECAI06), Riva del Garda, Italy, August, 2006*.
- A. Zinman and J. S. Donath. Is Britney Spears spam? In *In Proceedings of Fourth Conference on Email and Anti-Spam, Mountain View, CA, 2007*.