Detecting noise in recommender system databases

Conference Paper · January 2006

DOI: 10.1145/1111449.1111477 · Source: DBLP

3 authors, including:

Michael P. O'Mahony
University College Dublin
81 PUBLICATIONS **1,435** CITATIONS

SEE PROFILE

Neil Hurley
University College Dublin
163 PUBLICATIONS **2,434** CITATIONS

SEE PROFILE

Detecting Noise in Recommender System Databases

Michael P. O'Mahony
UCD School of Computer Science and Informatics
University College Dublin
Belfield, Dublin 4, Ireland
michael.p.omahony@ucd.ie

Neil J. Hurley
UCD School of Computer Science and Informatics
University College Dublin
Belfield, Dublin 4, Ireland
neil.hurley@ihl.ucd.ie

Guénolé C.M. Silvestre
UCD School of Computer Science and Informatics
University College Dublin
Belfield, Dublin 4, Ireland
guenole.silvestre@ihl.ucd.ie

ABSTRACT

In this paper, we propose a framework that enables the detection of noise in recommender system databases. We consider two classes of noise: natural and malicious noise. The issue of natural noise arises from imperfect user behaviour (e.g. erroneous/careless preference selection) and the various rating collection processes that are employed. Malicious noise concerns the deliberate attempt to bias system output in some particular manner. We argue that both classes of noise are important and can adversely effect recommendation performance. Our objective is to devise techniques that enable system administrators to identify and remove from the recommendation process any such noise that is present in the data. We provide an empirical evaluation of our approach and demonstrate that it is successful with respect to key performance indicators.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering; K.4.4 [**Electronic Commerce**]: Security

General Terms

Security, Performance, Experimentation

Keywords

Collaborative Recommender Systems, Malicious Attacks, Noise Detection, Robustness, Performance Measures

1. INTRODUCTION

In recent years, the need for automated techniques to deal with the information overload problem has become very clear. In this regard, recommender systems [16, 18, 17] are acknowledged as a key tool, enabling users to quickly find information and products that meet their own specific requirements. While the performance of these systems has been extensively evaluated in the literature, it is only in recent times that the issue of the security of these systems has been considered [6, 13].

Security is a major concern for all on–line systems and applications. Even if one assumes, however, that a recommender system's database and algorithms are secure against manipulation, these systems remain vulnerable given the open manner in which they operate. Since it is practically impossible to assess the integrity of those who use a system or to perform adequate quality assurance, there is no guarantee that the data that is inserted into a system reflects the true preferences of users. Thus, in the normal course of events, some degree of noise in the data must be anticipated. In general, noise in recommender system databases can be categorised by two classes. These classes are:

- **Natural Noise**. This class of noise relates to the methods by which recommender systems collect or infer user preferences. In the first instance, many recommender systems operate over explicit rating schemes, wherein users are required to express a preference for items that they have reviewed or purchased. Since all human activity is prone to error, and indeed, given that preference entry is often seen by users as an onerous process, some errors in the data will naturally occur. In systems where implicit rating schemes are utilised (for example, by inferring user preference from web site visits), the same, or perhaps even greater, potential for noise exists.

- **Malicious Noise**. Of more serious concern is the possibility of biased noise being deliberately inserted into a system. It is certainly conceivable that malicious agents may be motivated to attack recommender systems, given the significant advantages that may accrue. Since many recommender systems operate in a commercial environment, strong motives exist for those minded to behave in an unscrupulous manner. Consider, for example, authors wishing to promote their work by forcing a recommender system to output artificially high ratings for their publications, while reducing the recommendations made for other works of the same genre.

The objective of this paper is to develop techniques to identify and exclude from the recommendation process any noise that may be present in a system's database. If we are successful, removing natural noise from a system will result in an improvement in accuracy performance. In addition,

IUI'06, January 29–February 1, 2006, Sydney, Australia.

given the success of the various malicious attack strategies that have been investigated in previous work [6, 14], it is imperative that recommender algorithms are made robust against such activity.

With respect to attacks, the main attack types that have been considered in the literature are *product push* and *product nuke* attacks. The objectives of these attacks is to promote or demote the predictions that are made for targeted items, respectively. In this paper, we consider push attacks only, noting that a similar analysis and results apply with respect to nuke attacks.

In all cases, it is assumed that attackers have no direct access to a system's database. The only interaction an attacker has with the system is, as with genuine users, through the normal user interface. Attacks are implemented in the following manner. The attacker, through multiple registrations with the system, assumes several identities within the application database. It is in this manner that the attacker is able to insert biased noise into the system. Each of the separate identities assumed by the attacker are referred to as an *attack profile*.

In earlier work [13], we introduced the concept of *profile utility* as a means to achieve robust recommendation. In this approach, novel approaches to neighbourhood formation are defined which are based on certain characteristics of the items that are contained in users' profiles. In addition, new similarity weight transformations are proposed that serve to enhance the robustness of the recommendations that are made, particularly those that are made for new items which are likely to be the subject of most attacks. Other researchers have approached the problem of security from a trust perspective. For example, in [10], computational models of trust are developed and are integrated into standard collaborative recommendation algorithms. Trust is expressed as a function of a user's previous recommendation performance – i.e. the more accurate a user's past recommendations have been, the more likely it is that the user is trustworthy and suitable as a neighbour. In addition, it is shown that the inclusion of trust in the algorithm leads to an improvement in predictive accuracy. Some of the other related work in this field has been proposed in [1, 7, 9].

The profile utility approach proved most successful when based on the *popularity* of profile items, i.e. the number of times that items have been rated by users. The approach that we propose here is based on the actual *ratings* that have been assigned to profile items. Using signal detection theory, the aim is to provide system administrators with a tool to determine whether or not particular user profiles represent a security threat to a system through an examination of each user's rating behaviour.

2. NATURAL NOISE

We begin by introducing the following notation. A recommender system database is modelled as a user–item matrix, which contains the preferences that a set of users U have expressed on a set of items I. In this paper, we consider ratings–based recommender systems only. Let $r_{i,j} \in R$ represent the rating assigned by user i to item j, where R is a discrete set of rating values (including the special symbol $\perp$ which is interpreted as "not rated"). We denote by $U_g \subset U$ the set of genuine user profiles that are contained in a system and by $U_a \subset U$ the set of attack profiles that are present.

The problem that we are addressing in this section is to determine whether a particular rating $r_{u,v}$ contains natural noise or is a noise–free instance. For the purpose of this analysis, it is assumed that all of the user profiles that are contained in a system are genuine – i.e. $U_a = \emptyset$. The detection of noise needs to take place with respect to some framework. One approach, for example, would be to measure how closely the rating $r_{u,v}$ compares with the respective mean–user or mean–item rating. For ratings–based recommender systems, however, such an approach is not likely to perform well – it has been noted in [15] that the distributions of users' ratings are not necessarily centered around the midpoint of the rating scale in use.

The approach that is adopted in this paper is to consider how consistent the actual rating for a particular user–item pair is with respect to the predicted rating that is made using some recommendation algorithm, G. To make such a prediction, we assume that a training set $T \subset U_g$ exists, on which all predictions are based. In practice, the responsibility of selecting the members of the training set would rest with the system administrator. Such a set would consist of, for example, known trusted users who have rated the item in question and/or data obtained from critical product reviews, etc. It is important to note that the individual system users would have no role in the the selection of training set members. This approach is advantageous from two perspectives – firstly, users do not have to concern themselves with the identification of such training sets and secondly, a high degree of quality assurance can be achieved when this task is performed exclusively by the system administrator.

We define the *consistency* c of a rating $r_{u,v}$ as the Mean Absolute Error [2] between the actual and predicted rating as follows:

$$c(G,T)_{u,v} = \frac{|r_{u,v} - p_{u,v}|}{r_{max} - r_{min}} \quad (1)$$

where $p_{u,v}$ is the predicted rating for the user–item pair (u, v) and r_{min}/r_{max} are the minimum/maximum permissible ratings, respectively. The normalisation term is introduced so as to facilitate a comparison between the performance of systems that employ different rating scales. A rating $r_{u,v}$ is deemed to represent noise, and is excluded from the recommendation process, if:

$$c(G,T)_{u,v} > th \quad (2)$$

where th is a threshold value. Figure 1 illustrates the approach.

In essence, the underlying assumption behind our approach is that the particular recommendation algorithm is "always" accurate to within the threshold th, and that accuracy for any user–item pair exceeding this threshold implies the presence of noise in that rating. One of the novel aspects of our scheme is that it imposes a two–fold check on neighbourhood selection. In traditional neighbourhood formation schemes, such as k–nearest neighbour and neighbourhood thresholding [5], neighours are selected based on their similarity to the active user. By definition, such similarity computations cannot be based on the target item (i.e. the item for which a prediction is being sought), since it has, as yet, received no rating from the active user. With our approach, a further check on the suitability of candidate neighbours can be performed by examining the ratings that have been assigned by them to the target item.

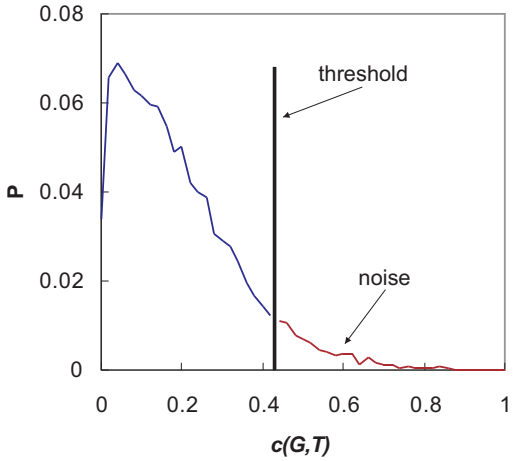

Figure 1: Probability distribution of $c(G,T)$, which is calculated according to normalised mean absolute error. Values that exceed the threshold are deemed to represent noise in the data. (The data shown is the actual distribution obtained from the MovieLens dataset.)

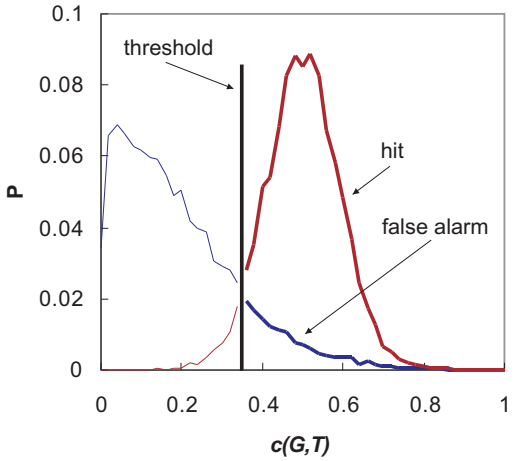

Figure 2: Probability distributions of $c(G,T)$ for genuine (left) and attack profiles (right), which are calculated according to normalised mean absolute error. (The data shown is for a random attack, carried out against the MovieLens dataset.)

3. MALICIOUS NOISE

Malicious noise concerns the deliberate insertion of attack profiles into a system with the objective of influencing the predictions that are made in some manner. A similar approach to that outlined above is adopted to distinguish between genuine and attack profiles that are present in a system. As before, we assume the existence of a training set $T \subset U_g$. Further, assume that the object of the attack is to push a particular item v. Two test sets are constructed which contain users who have all rated item v. The first test set $T_g \subset U_g$, $T_g \cap T = \emptyset$ contains known genuine profiles only; the second set $T_a \subset U_a$ consists exclusively of known attack profiles that have been created according to some malicious attack strategy. Both sets contain an equal number of profiles, i.e. $|T_g| = |T_a|$. Using the training set T, predictions are made for item v across all the profiles that are contained in each test set. The consistency c is computed for the predicted ratings according to (1). The above procedure is repeated for all items $v \in I$. The distributions of the consistency of genuine (c_g) and attack profiles (c_a) can now be obtained. Given that consistency is defined as the absolute error between actual and predicted ratings, we would expect that $\mu_{c_g} < \mu_{c_a}$, where μ_{c_g} and μ_{c_a} are the mean consistencies of the genuine and attack distributions, respectively.

Signal detection theory can now be applied to the problem of detecting malicious noise. If there is no overlap between the distributions, then choosing a threshold value above which malicious noise is deemed to be present is trivial. In many cases, however, it is reasonable to expect some overlap in the distributions, and the selection of a threshold value becomes more problematic. Such a scenario is illustrated in Figure 2, where the hit and false alarm rates that correspond to a particular threshold value are shown. For both hits and false alarms, since the consistency value is greater than the threshold, we decide noise is present. Hits correspond to the correct identification of malicious noise; false alarms correspond to the misidentification of genuine profiles as noise.

When the distributions overlap, it is not possible to achieve a perfect hit rate without incurring at least some false alarms. The findings in [14] indicate that the presence of even small quantities of malicious noise can significantly influence predictions. It is therefore necessary to choose a threshold that ensures a high hit rate at the expense of a correspondingly high rate of false alarms. From the analysis that is presented in the previous section, however, some of these false alarms may correspond to natural noise in the data, and any such false alarms can therefore be safely ignored. In Section 5, the trade–off between different threshold values and accuracy and coverage performance is examined. If it is possible to maintain good performance across these criteria, while maintaining a perfect hit rate, then we have succeeded in making the recommendation process robust against attack.

3.1 Attack Strategies

Four specific attack strategies are considered in this paper. It is assumed that full knowledge concerning both *item popularity* and *item likeability* is available to an attacker. Item popularity refers to the number of ratings received by each item and item likeability refers to the average rating received by each item. Where required, this knowledge facilitates the creation of optimum attack profiles so as to maximise attack success. (See [14] for an analysis on the effect of the above knowledge on the performance of particular attack strategies.) The attacks as outlined below relate to product push attacks – product nuke attacks are not considered but are similarly constructed [11, 14].

3.1.1 Random Attack

The random attack is the most straightforward strategy that we consider. Attack profiles are constructed by simply choosing $l-1$ items at random from the product space under attack. The ratings for these items are chosen accord-

ing to a uniform random distribution over the rating scale that is employed. The item being pushed is also included in the profiles, and is set to a rating of r_{max}. In this and all subsequent attacks, l is set to the average profile size of the genuine users that are present in a system.

3.1.2 Popular Attack

The key motivation behind this approach is that the most popular items of the product space under attack are ideal candidates for attack profiles. Such items are generally easy to identify and also provide the following advantages from an attacker's perspective [14]:

(a) a large number of common items between genuine and attack profiles can be expected, helping to ensure high genuine–attack profile similarities,

(b) such profiles have a high probability of being located in the neighbourhood of many targeted users, thereby reducing the *cost* of attack in terms of the number and size of attack profiles that are required, and

(c) popular items tend to receive consistent and predictable (i.e. high) ratings from users.

Attack profiles are constructed using the $l-1$ most popular items of the targeted product space along with the item being pushed. The following ratings are applied: the pushed item is assigned a rating of r_{max} and ratings of $r_{min}+1$ and r_{min} are assigned to the more– and less–liked remaining items, respectively.

3.1.3 Probe Attack

The probe attack was originally proposed in [14] and uses the output of a recommender system as a means to select attack profile items and ratings. By rating a small number of initial seed items, the attacker can interrogate the system and progressively build up attack profiles which will closely match the distribution of genuine users in the system. Thus, the probability of strong similarities between genuine and attack profiles is likely to be high. In our evaluations, attack profiles are created by choosing 10 seed items at random from the 100 most popular items that are contained in the product space. The same ratings strategy as outlined in the previous section is applied to these seed items – thereafter the recommender system is used to select any further items and corresponding ratings. As before, the item under attack is rated r_{max} and the total number of items in each attack profile is equal to l.

3.1.4 AverageBot Attack

This attack, proposed in [6], involves the creation of attack profiles consisting of all items that are contained in a system. The pushed item receives a rating of r_{max}. All other items are rated randomly on a normal distribution with mean equal to the average rating of the item being rated and standard deviation equal to the standard deviation across all items. In our implementation of this attack, we apply the same ratings strategy but limit the number of items to the $l-1$ most popular items from the product space under attack. Another strategy, referred to as the *RandomBot* attack, was also proposed in [6]. This approach did not, however, perform as well as the *AverageBot* attack and thus is not considered further.

4. RECOMMENDATION ALGORITHM

In this paper, we consider the well established user–based collaborative recommendation algorithm that is described in [15]. This algorithm employs a deviation from mean approach to the calculation of predictions, taking into account any differences in rating patterns across different users. A prediction $p_{a,j}$ is computed for a user a on an item j as a weighted average of n neighbours' ratings as follows:

$$p_{a,j} = \bar{r}_a + \frac{\sum_{i=1}^{n} w(a,i)(r_{i,j} - \bar{r}_i)}{\sum_{i=1}^{n} |w(a,i)|} \quad (3)$$

where $\bar{r}_a$ is the average rating of user a, $r_{i,j}$ is the rating assigned by neighbour i to item j and $w(a,i)$ is the similarity between the active user a and neighbour i.

We use a tuned version of the user–based ACF algorithm that was described above. We employ the k–nearest neighbour neighbourhood scheme, with neighbourhood size chosen by experiment in order to optimise predictive accuracy (k is set to 35 for MovieLens and 45 for EachMovie). In addition, we incorporate the significance weighting algorithm extension proposed in [5], which considers the number of co–rated items between users when computing similarity weights. Weights that are based on low numbers of co–rated items are devalued. In this paper, we consider the Pearson correlation similarity metric [15] only. In earlier work [12], the performance of various other neighbourhood formation schemes and similarity metrics when subjected to attack was examined. None were found to provide robustness against attack.

4.1 Metrics

We use the following metric to evaluate the robustness of systems that are subjected to attack. For product push attacks, we calculate the increase in the number of *good predictions* (GP) that are made for targeted items following an attack. A good prediction for an item j is defined as the number of times the following expression holds true:

$$p_{u,j} : p_{u,j} \geq \delta_g, \forall\ u \in U_j \quad (4)$$

where δ_g is a threshold rating and U_j is the set of all genuine users who have rated item j.In this paper, we set $\delta_g = r_{max} - 1$.

Accuracy and coverage results are also presented. Accuracy is calculated according to Normalised Mean Absolute Error [4]. Coverage is defined as the ratio of the number of predictions that a system is able to deliver to the total number of predictions that are requested.

5. EXPERIMENTAL EVALUATION

5.1 Datasets

We use the following real–world datasets in our evaluations.

The EachMovie recommender system operated between 1995 and 1997. The original dataset has some $72,916$ users who provided $2,811,983$ ratings on $1,628$ movies. From this, a random sample of $1,000$ users is selected, which contains $63,507$ transactions on a rating scale of 1 to 6.

The second dataset that is used is provided by the MovieLens research project. MovieLens is a web–based movie recommender system that began operating in 1997. It consists of 943 users, $1,682$ movies and contains $100,000$ transactions in total. Ratings are based on a scale of 1 to 5.

5.2 Experimental Procedure

In this paper, the following procedure is used to evaluate robustness, accuracy, and coverage performance. The performance of a particular item is evaluated over all the genuine users in the system who have rated the item in question. For each of these users, an *all but one* protocol is adopted wherein the test set is comprised of the user–item pair in question, and a prediction for this pair is made using all remaining data. The average result over all users is then computed. This procedure is repeated for all items that are contained in the system, and the mean result is presented.

5.3 Results – Natural Noise

In the natural noise detection framework described in Section 2, the key issue concerns the selection of the threshold value th. Ratings whose consistency values exceed the threshold are deemed to contain noise. In all of our experiments, training sets were constructed using genuine profiles that were randomly selected from the system database, subject to the condition that each had rated the item for which a prediction is being sought.

In the results that are presented below, training sets consisted of at most 20 users. Multiple experiments, conducted across a range of different training sets, provided similar results. Experiments were also carried out using larger–sized training sets, but no significant improvement in noise detection rates were observed.

Figure 3 shows the effect of various values of the threshold value on the accuracy performance that is achieved for the EachMovie (EM) and MovieLens (ML) datasets. At the lowest threshold value of $th = 0.01$, accuracy performance was poorest, but thereafter improved as the threshold was increased. At higher threshold values, the accuracy that was achieved exceeded that of the standard algorithm (labeled $th = inf$ in the figure). In addition, the accuracy that was provided by a non–personalised algorithm, which calculates a prediction by simply returning the average of all ratings that a particular item has received, was worse than that achieved at even the lowest threshold value. The non–personalised algorithm resulted in normalised mean absolute error values of 0.1986 and 0.2341 for EachMovie and MovieLens, respectively, compared to corresponding values of 0.1941 and 0.2153 at $th = 0.01$.

The coverage that was provided by our approach is shown in Figure 4. In this case, the best coverage was achieved when no noise detection was performed (i.e. at $th = inf$). For all other values of $th < inf$, the coverage provided was diminished. In the case of MovieLens, for example, 75% coverage was still achieved at $th = 0.05$, compared to 91% for the standard algorithm. This result was to be expected, given that, as the threshold was reduced, increasingly fewer neighbours were able to satisfy the threshold criterion.

The above findings are significant, since the removal of natural noise from the prediction process resulted in improved accuracy performance at certain threshold values, thereby justifying our approach. While coverage was seen to decrease as the threshold was reduced, the rate of decline in performance was relatively small – with the exception of at $th = 0.01$, the lowest value that was considered. In addition, recall that these findings were based on training sets that consisted of at most 20 known genuine users – the identification of such few numbers of genuine users by system administrators ought not to be too problematic in practice. These findings are discussed further in the next section, where the relationship between threshold value and robustness against malicious attack is examined.

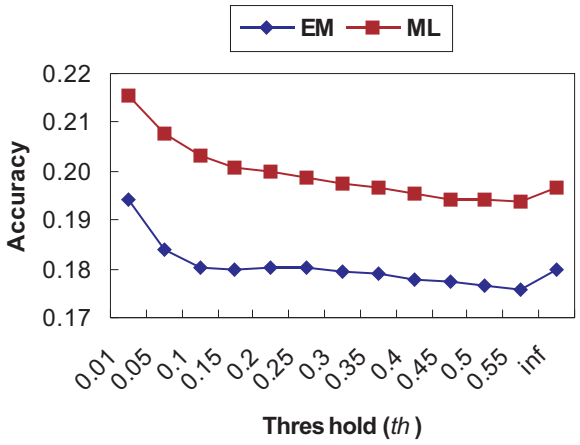

Figure 3: Accuracy calculated according to normalised mean absolute error against noise threshold values th.

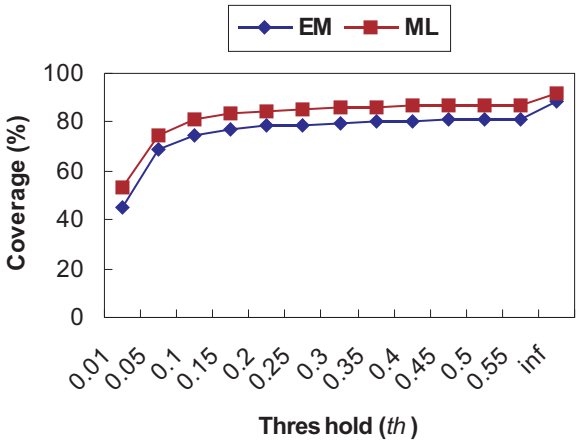

Figure 4: The effect of the noise threshold th on system coverage.

5.4 Results – Malicious Noise

We now evaluate our malicious noise detection framework as outlined in Section 3. As before, at most 20 users were contained in the training set T. The sizes of both the genuine and attack test sets (T_g and T_a, respectively) was fixed at a maximum of 10. The number of items l that were contained in each attack profile was set to the average genuine profile size – 64 and 106 for the EachMovie and MovieLens datasets, respectively. Experiments using larger–sized training and test sets were also carried out – none of these yielded a significant improvement in terms of noise detection.

For both datasets, we applied our framework to each of the attacks that are described in Section 3.1. The results are presented in the form of Receiver Operating Characteristic (ROC) curves, as shown in Figures 5 and 6. ROC curves are produced by plotting false alarm and corresponding hit rates

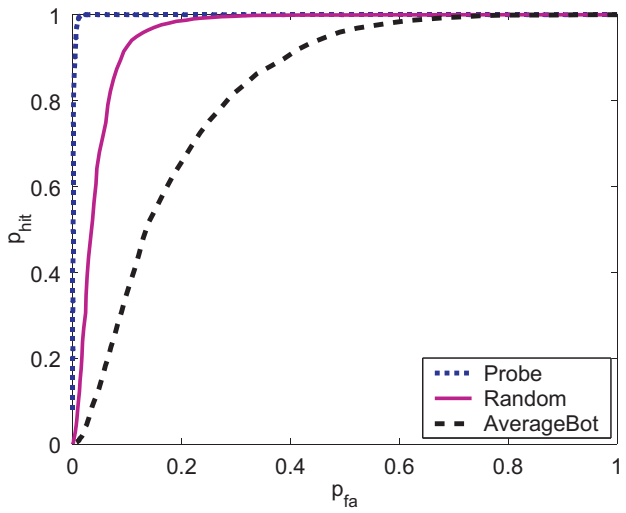

Figure 5: ROC curves for the various attack strategies carried out on the EachMovie dataset. The curve for the Popular attack strategy is omitted – the result closely matched that of the Probe attack.

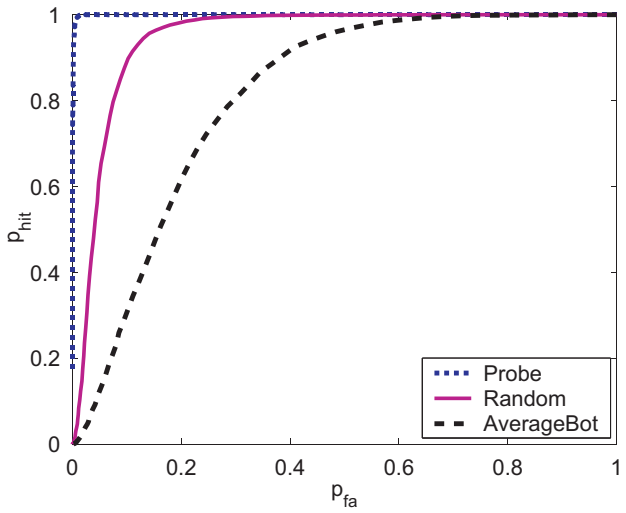

Figure 6: ROC curves for various attack strategies carried out on the MovieLens dataset. As before, the curve for the Popular attack strategy closely matched that of the Probe attack, and is omitted.

for various values of the threshold th. If there is little or no overlap between the genuine and attack consistency distributions, then high hit rates are achieved at little cost in false alarms. Such is the case for the probe attack strategy, as is indicated in the figures. Similar trends were observed for a popular attack carried out against both datasets (for clarity of presentation, the popular attack results are omitted). Thus, for these attacks, our noise detection framework has performed very successfully.

To analyse these results in greater detail, we performed the following experiments using the EachMovie dataset,[1] in which the effect of the four attacks versus noise threshold is investigated. A total of 64 attack profiles were inserted into the dataset in the course of each of the attacks. The percentage increase in the number of good predictions that was achieved by each attack is shown in Figure 7. As can be seen from the figure, the popular and probe attacks achieved very considerable success when no threshold was applied (i.e. at $th = inf$), where the percentage increase in good predictions was 250% and 207%, respectively. However, both attacks were rendered ineffective for all threshold values $th \leq 0.5$.

Regarding the random attack, our approach was also successful. For each of the datasets, a 100% hit rate was realised at a false alarm rate of approximately 35%. This false alarm rate corresponds to a noise threshold of $th = 0.18$. Referring to Figure 7, however, it is clear that this attack was the least successful of the four that were considered. Given that attack profile items and ratings were chosen randomly, this result was to be expected. It can be seen that, for all $th \leq 0.45$, the attack achieved little success – at most a 6% increase in the number of good predictions.

For both datasets, attack profiles constructed using the *AverageBot* strategy proved to be the most difficult to detect, requiring high false alarm rates of approximately 74% in order to achieve perfect noise detection. The results in Figure 7 indicate that the attack remained successful at the higher threshold values. For example, a 100% increase in the number of good predictions was observed at a threshold value of $th = 0.4$. At lower threshold values, however, the attack was much less successful, where a percentage increase of only 5% was achieved at a threshold value of $th = 0.05$.

In previous work, we have argued that, in addition to accuracy and coverage performance, robustness against malicious attack also needs to be considered as a key performance criterion for recommender systems. As is usual in cases where multiple criteria apply, some trade–offs in performance are to be expected. The findings of this section and those of Section 5.3 indicate that satisfactory trade–offs exist with respect to the datasets that were evaluated. For example, choosing a threshold value of $th = 0.05$ in each case assures robustness against all of the attack types that were considered, while maintaining good accuracy and coverage performance. It is also clear that very similar trends were observed when our approach was applied to both datasets. This is an important result, since it indicates that, at least in the case of these particular datasets, appropriate noise detection thresholds can be set which are system independent.

6. CONCLUSIONS

The issue of security is a major concern for all software applications, and in this paper, we have proposed a framework that enables natural and malicious noise to be identified and removed from recommender system databases. We have shown the the removal of natural noise leads to an improvement in predictive accuracy. In addition, using signal detection theory, we have devised a technique for detecting any attack profiles that may be present in a system. We have demonstrated that our approach provides robustness against four product push attack strategies, while also delivering good performance across other key dimensions.

The framework that we have developed is dependent on

[1]Similar trends were observed for the MovieLens dataset.

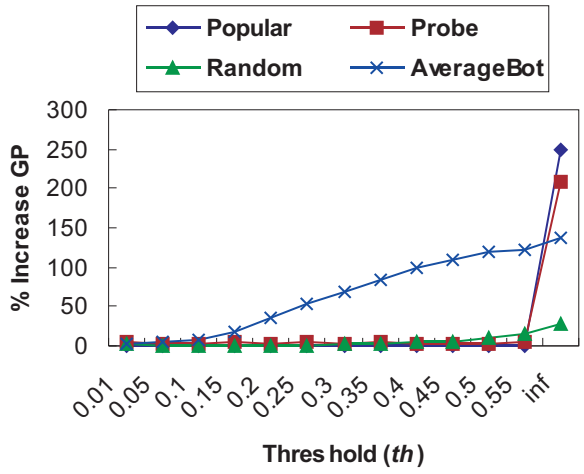

Figure 7: The effect of the noise threshold *th* on the percentage increase in the number of good predictions that were achieved by the various attack strategies (EachMovie dataset).

the particular recommendation algorithm that is employed. We plan on expanding our analysis to various other recommendation algorithms that have been proposed in the literature. Ideally, knowledge of any particular algorithm should not render it susceptible to attack. As a matter of course, new attack strategies will be developed in an attempt to compromise system integrity. Recent work in [3, 8] proposes a number of such strategies. Thus, the issue of secure recommendation is likely to remain for some time to come, and offers wide scope for future research.

7. REFERENCES

[1] P. Avesani, P. Massa, and R. Tiella. Moleskiing: A trust–aware decentralised recommender system. *In Proceedings of the 1st Workshop on Friend of a Friend, Social Networking and the Semantic Web*, September 1–2 2004.

[2] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. *In Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 43–52, July 24–26 1998.

[3] R. Burke, B. Mobasher, and R. Bhaumik. Limited knowledge shilling attacks in collaborative filtering systems. *In Proceedings of Workshop on Intelligent Techniques for Web Personalization (ITWP'05)*, August 2005.

[4] J. Canny. Collaborative filtering with privacy via factor analysis. *In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 238–245, August 11–15 2002.

[5] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. *In Proceedings of the 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 230–237, August 15–19 1999.

[6] S. K. Lam and J. Riedl. Shilling recommender systems for fun and profit. *In Proceedings of the 13th International World Wide Web Conference*, pages 393–402, May 17–20 2004.

[7] P. Massa and P. Avesani. Trust–aware collaborative filtering for recommender systems. *In Proceedings of the International Conference on Cooperative Information Systems (CoopIS'04)*, pages 492–508, October 25–29 2004.

[8] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams. Effective attack models for shilling item-based collaborative filtering system. *In Proceedings of the 2005 WebKDD Workshop (KDD'2005)I*, August 2005.

[9] M. Montaner, B. Lopez, and J. L. de la Rosa. Developing trust in recommender agents. *In Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 304–305, July 15–19 2002.

[10] J. O'Donovan and B. Smyth. Trust in recommender systems. *In Proceedings of the 10th International Conference on Intelligent User Interfaces (IUI'05)*, pages 167–174, January 10–13 2005.

[11] M. P. O'Mahony. *Towards Robust and Efficient Automated Collaborative Filtering*. PhD thesis, University College Dublin, Department of Computer Science, Belfield, Dublin 4, Ireland, Dec 2004.

[12] M. P. O'Mahony, N. J. Hurley, and G. C. M. Silvestre. An evaluation of the performance of collaborative filtering. *In Proceedings of the 14th Irish International Conference on Artificial Intelligence and Cognitive Science (AICS'03)*, pages 164–168, September 17–19 2003.

[13] M. P. O'Mahony, N. J. Hurley, and G. C. M. Silvestre. Efficient and secure collaborative filtering through intelligent neighbour selection. *In Proceedings of the 16th European Conference on Artificial Intelligence (ECAI'04)*, pages 383–387, August 23–27 2004.

[14] M. P. O'Mahony, N. J. Hurley, and G. C. M. Silvestre. Recommender systems: Attack types and strategies. *In Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05)*, pages 334–339, July 9–13 2005.

[15] 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.

[16] P. Resnick and H. R. Varian. Recommender systems – introduction to the special section. *Communications of the ACM*, 40(3):56–58, 1997.

[17] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Recommender systems for large–scale e–commerce: Scalable neighborhood formation using clustering. *In Proceedings of the Fifth International Conference on Computer and Information Technology (ICCIT'02)*, December 27–28 2002.

[18] J. B. Schafer, J. Konstan, and J. Riedl. Recommender systems in e–commerce. *In Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 158–166, November 3–5 1999.