Promoting Recommendations: An Attack on Collaborative Filtering

Conference Paper · September 2002

DOI: 10.1007/3-540-46146-9_49 · Source: DBLP

CITATIONS
82

READS
322

3 authors, including:

Michael P. O'Mahony
University College Dublin
83 PUBLICATIONS **1,929** CITATIONS

SEE PROFILE

Neil Hurley
University College Dublin
185 PUBLICATIONS **3,554** CITATIONS

SEE PROFILE

Promoting Recommendations: An Attack on Collaborative Filtering

Michael P. O'Mahony, Neil J. Hurley, and Guenole C.M. Silvestre

Department of Computer Science, University College Dublin,
Belfield, Dublin 4, Ireland
{michael.p.omahony, neil.hurley, guenole.silvestre}@ucd.ie

Abstract. The growth and popularity of Internet applications has reinforced the need for effective information filtering techniques. The collaborative filtering approach is now a popular choice and has been implemented in many on-line systems. While many researchers have proposed and compared the performance of various collaborative filtering algorithms, one important performance measure has been omitted from the research to date - that is the *robustness* of the algorithm. In essence, robustness measures the power of the algorithm to make good predictions in the presence of noisy data. In this paper, we argue that robustness is an important system characteristic, and that it must be considered from the point-of-view of potential attacks that could be made on a system by malicious users. We propose a definition for system robustness, and identify system characteristics that influence robustness. Several attack strategies are described in detail, and experimental results are presented for the scenarios outlined.

1 Introduction

The information overload problem has prompted much research and various techniques have been proposed to improve matters. Content-based information retrieval algorithms filter information by matching user preference data against item descriptions and thereby eliminate irrelevant content. Collaborative filtering [4], [11] is an alternative class of retrieval algorithms and has gained in popularity in recent years. These so-called recommender systems attempt to filter in a personalised manner, and operate by recommending items to a user based upon the transactions of similar users in the system. Collaborative filtering algorithms have now been implemented in many application areas, ranging from on-line book stores, movie and music finders, job recruitment services, etc.

Many approaches to collaborative filtering have appeared in the literature. These include memory-based algorithms, which utilise the entire user database to make recommendations; and model-based approaches, where a model is first derived from the user database and is then used to make predictions [6], [12], [1], [3]. Many researchers have compared the performance of the various algorithms, with refinements and new ideas continuously emerging.

However one important performance measure has been omitted from the research to date - that is the *robustness* of the algorithm. In essence, robustness

R. Cicchetti et al. (Eds.): DEXA 2002, LNCS 2453, pp. 494–503, 2002.

measures the power of the algorithm to make good predictions in the presence of noisy data. Generally, collaborative filtering applications update their datasets whenever users input new ratings, without any quality assurance that the entered rating is a true reflection of a real user's preference. Indeed, since rating entry can be an onerous process, users may be careless in the values that they enter and inaccuracies (or noise) in the data must be expected. More sinisterly, it is possible to imagine scenarios in which malicious users may be motivated to deliberately attack the recommendation system and cause it to malfunction. (Imagine, for example, publishers wishing to promote their work by forcing a book recommendation system to output artificially high ratings for their publications). Depending upon the application, the cost of significantly modifying the recommendation system's output may be prohibitively high. Consider that the author may have to buy a book each time he wishes to modify the book's rating. Cost for other applications may simply be in terms of the amount of time required to input erroneous data.

Most recommendation applications operate in a web environment in which it is impossible to check the honesty of those who access the system. Hence, it is important to understand how much tolerance the recommendation algorithm has to noise in the dataset. In this paper, we propose a definition for system robustness, and identify system characteristics that influence robustness. Several attack strategies are described in detail, and experimental results are presented for the scenarios outlined.

2 Formal Framework

The performance of a system can only be measured with respect to some utility function which models the usefulness of a recommendation. Clearly, the utility of a recommendation depends upon the perspective of the players involved.

In our model of robustness, we take the end-user's perspective (to whom recommendations are delivered), and seek systems which are consistent in the recommendations that are made. Primarily, we examine how the system can be compromised by external third-parties (such as authors or publishers in the case of a book recommendation system). To such "attackers", the system is a black-box which can only be viewed through the recommendations that it outputs and can only be modified by the insertion of new data through the normal user interface.

We consider memory-based collaborative filtering (CF) in which the task is to predict the votes of a particular user (the *active* user) from a database of user votes, drawn from a population of other users. Let $\mathcal{U}$ be the universe of all users and fix some set of items I for which users vote. The votes of a user a on all items may be expressed as a vector, termed the *user profile* $\mathbf{v}_a \in V^m$ where V is a discrete set of vote values (including the special symbol $\perp$ interpreted as "not voted on") and m is the total number of items. A CF database D_U for $U \subset \mathcal{U}$ is a collection of votes $\{\mathbf{v}_a | a \in U\}$ for a particular set of users U.

Let $v_{i,j}$ be the j^{th} element of $\mathbf{v}_i$ corresponding to the vote for user i on item j. Using the notation of [2], define I_i as the set of items on which user i has voted. The predicted vote of the active user a for item j, $p_{a,j}$ is given by

$$p_{a,j} = \bar{v}_a + \kappa \sum_{i=1}^{n} w(a,i)(v_{i,j} - \bar{v}_i) \; . \tag{1}$$

where $\bar{v}_i$ is the mean vote for user i, n is the number of users in the database with non-zero weights $w(a,i)$ and κ is a normalising factor.

The Pearson correlation coefficient weighting, proposed in [11] is adopted to explore and test our ideas on robustness in CF systems. Hence,

$$w(a,i) = \frac{\sum_j (v_{a,j} - \bar{v}_a)(v_{i,j} - \bar{v}_j)}{\sqrt{\sum_j (v_{a,j} - \bar{v}_a)^2 \sum_j (v_{i,j} - \bar{v}_i)^2}} \; . \tag{2}$$

where the sum is over those items which have been voted for by *both* user a and user i. From now on, we refer to this algorithm as *Resnick's algorithm* [11].

While various enhancements to this standard model have been proposed in the literature, such as default voting and case amplification [2], we restrict ourselves to the standard model as a solid and non-trivial basis upon which to develop our discussion. In our implementation of Resnick's algorithm, we have imposed a limit to the number of users involved in the calculation of the predicted rating, i.e. a nearest-neighbour approach is employed with the top-k nearest users chosen on the basis of highest (absolute) correlation with the active user.

3 Definitions of Robustness

One of the underlying claims of knowledge-based systems (KBS) is that they can deal with incomplete, inaccurate or uncertain data. In this regard, the previous work of Hsu and Knoblock [7], [8] and Groot et al [5], who propose methodologies for examining the robustness of knowledge discovery and knowledge-based systems, is of interest. However, much of the work done to date tends to focus on "normal" variations in the data. In this paper, we argue that robustness from the point-of-view of malicious attacks carried out on the system also needs to be considered.

We take the following approach to robustness of collaborative filtering. An *attack* is a transformation T which maps a database D_U to a new database $D'_{U'}$. Under the transformation, each vote in D_U $v_{i,j}$ is mapped to a new vote $v'_{i,j}$. If either $v_{i,j} = \perp$ or $v'_{i,j} = \perp$,this implies the addition of new votes or the deletion of existing votes, respectively. A transformation may also entail the addition of new users, so that $U \subseteq U'$.

Let $\mathcal{T}$ be the set of all possible attacks. Define a cost function $C : \mathcal{T} \rightarrow \mathcal{R}$. In general the cost of a transformation is application dependent. Various criteria can be used to construct a cost function. If the cost is related to the amount of effort it takes to perform the transformation, then this could be modelled by

a monotonically increasing function of the number of user-item pairs which are modified by the transformation. There may also be a real cost, if ratings can only be entered into the system by purchasing the item in question.

Definition 1. *Fix a set, A, of unrated user-item pairs in the database, which remain unrated in the transformed database i.e. $A \subseteq \{(a,j)|v_{a,j} = v'_{a,j} = \perp\}$. We define robustness for the set A. Assuming that the set of vote values is bounded above by R_{max} and below by R_{min}, for each $(a,j) \in A$, the normalised absolute error, NAE, of prediction pre- and post-attack T is given by*

$$NAE(a,j,T) = \frac{|p_{a,j} - p'_{a,j}|}{\max\{|p_{a,j} - R_{max}|, |p_{a,j} - R_{min}|\}} \ . \tag{3}$$

Definition 2. *The* robustness *of prediction $p_{a,j}$ to attacks of cost c is given by*

$$Robust(a,j,c) = 1 - \max_{\{T \in \mathcal{T}: C(T) \leq c\}} NAE(a,j,T) \ . \tag{4}$$

Furthermore, the robustness of the CF recommendation system on the set A to attacks of cost c is given by

$$Robust(A,c) = \min_{(a,j) \in A} Robust(a,j,c) \ . \tag{5}$$

While the above definition gives a strong measure of robustness on the set A, it may also be useful to examine various other projections of the robustness (i.e. average or maximum robustness) over elements of A. Note that our definition of robustness does not contain any notion of "true" rating for the user-item pair. This reflects the distinction between the accuracy and robustness performance measures. A system which reports the same ratings regardless of the dataset is very robust, though it is unlikely to be very accurate.

The above definition is a useful measure when we are concerned with the general robustness of a system. However, additional metrics may be required depending on the nature of an attack. Consider, for example, a scenario whereby the objective is to force the predicted ratings of a particular item, or group of items, to a certain value. Given the set A of user-item pairs and an item j, let $A_j \subseteq A$ be those members of A who have rated j. Let $\mathcal{T}_c$ denote the set of all attacks of cost c from this class.

Definition 3. *We define* power of attack *(POA) for attack $T \in \mathcal{T}_c$ and item j by*

$$POA(A_j,j,T) = 1 - \frac{1}{|A_j|} \sum_{a \in A_j} \kappa_{a,j} \ . \tag{6}$$

where $\kappa_{a,j} = 1$ iff $p'_{a,j} = R_{target}$ and 0 otherwise. We can now write the average POA for all such attacks of cost c over set A_j as

$$POA(A_j,j,c) = \frac{1}{|\mathcal{T}_c|} \sum_{T \in \mathcal{T}_c} POA(A_j,j,T) \ . \tag{7}$$

Let I be the set of all items in A. Then the average POA of an attack of cost c on set A is:

$$POA(A, c) = \frac{1}{|I|} \sum_{j \in I} POA(A_j, j, c) \; . \tag{8}$$

In our experimental evaluation (Section 5), the cost is simply a function of the number of attack profiles that are added to the database.

Note that while an item-based attack may not force all of the post-attack predicted ratings to target, nevertheless the distribution of these ratings may be shifted significantly towards the target distribution. Therefore it is also worthwhile to compare the pre-and post-attack distributions of predicted ratings with the target distribution.

4 Attacks

Various forms of attack are possible on CF systems, and in this section we discuss some potential scenarios that may occur in practice. In all cases, the strategy used to mount an attack is the same. An attack consists of a set of malicious user profiles, which is inserted (via the normal user-interface) into the database.

Note that with Resnick's algorithm, the attack profiles added need to be sufficiently close or similar to the target users if they are to have an influence on the predicted ratings. For example, consider an attack designed to promote a particular product. The ideal scenario would be that the attack profiles would correlate perfectly with the target users and therefore maximise the predicted rating of the product. Attack parameters, such as size and number of profiles added, along with the items and ratings that comprise each profile, will need to be adjusted to implement the desired attack. In the following sections, we discuss these aspects in detail and identify some system characteristics that may be exploited to reduce system robustness.

4.1 Scenario 1: Random Attack

In a *random attack*, we consider that the goal of an attacker is to reduce the overall performance of a system as a whole. In these attacks, the focus is not on particular users or products, rather it is to target all users and items equally in an attempt to compromise the general integrity of the system. As a potential real-life scenario, consider a rival recommender system owner, who wishes to undermine the opposition and attract additional customers. The attack strategy in this case is relatively straightforward - the number of items in the attack profiles are randomly selected, along with the items and ratings that comprise them.

4.2 Scenario 2: Random Product Push/Nuke Attack

A *product push* or *nuke* attack attempts to force the predicted ratings of a particular item, or group of items, to some target rating. Imagine, for example,

authors wishing to promote books by forcing a recommendation system to output high ratings for their work. In this scenario, an attacker implements the attack by building false profiles with the item to be pushed (or nuked) set to the maximum (or minimum) rating, and with the remainder of the items selected randomly as before.

4.3 Scenario 3: Focussed Product Push/Nuke Attack

Here again the attacker's goal is to target specific items, but in this case an important weakness in the Pearson correlation formula is exploited to implement the attack. The particular weakness in question is that the correlation between users is calculated only over the items that *both* users have rated. Hence, users can correlate strongly even though they have few items in common. While this weakness has been noted from the predictive accuracy point-of-view and some modifications to the standard model have accordingly been proposed, we highlight here its implications for robustness of recommendation.

With Pearson's formula, the correlation between users who have *exactly* two items in common is always $+1$ or -1. If the attacker adopts a strategy of building false user profiles consisting of the item to be pushed (or nuked) together with two other carefully selected items, then the probability of a successful attack is high. Ideally, two items about which there is strong consensus in the user population are selected to generate the attack profiles.

The remaining difficulty in this strategy is in ensuring that the attack profiles correlate in the same way (i.e. either positively or negatively) with all the target users. This implies that the attacker needs to know the ratings that the target users have given to the items contained in the attack profiles. However, some simple heuristics can suffice in practice. It is a fair assumption that items that have many ratings in the database (i.e. popular items) are generally rated higher than average. Such items are therefore appropriate choices when building attack profiles.

In addition, we discuss the scenario where multiple items are targeted in a single attack. In the next section, we demonstrate that by constructing a set of *independent* attack profiles it is possible to implement this type of attack successfully. Finally, we conclude by highlighting the vulnerability to attack that consistently and frequently rated items pose to a system, and the corresponding reduction in robustness that follows.

5 Results

We used two different datasets to evaluate the attack scenarios outlined in the previous section. The `Movielens` dataset [9] consists of 943 users, 1,682 movies and contains 100,000 transactions in total. Movies are rated on a scale of 1 to 5. The second dataset, `PTV` [10], is more sparse and consists of 1,542 users, 8,129 TV programs, and has 58,594 transactions. The rating scale for this dataset is from 1 to 4. From experiment we set the number of nearest neighbours to 50

for all attacks involving both datasets. An *all but one* protocol [2] is adopted in which the test set is obtained by removing a single user-item pair from the database. A prediction is then made for this pair using all the remaining data.

5.1 Scenario 1

A general random attack is performed on the `Movielens` and `PTV` databases. In the attack a number N of random attack profiles are generated. The number of items in each profile is a uniformly random number ranging between the minimum and maximum profile sizes that occur in the database. The results are presented in Fig. 1. Average robustness according to Eqn. 5 is plotted against $X\%$ (a percentage of the total number of users in the original dataset) of attack profiles added. For both databases, this attack is not very successful, with robustness falling to only 0.93 for `Movielens` and 0.85 for `PTV` at $X = 500\%$ (i.e. when 5 times the number of original users are added as attack profiles). This can be explained by the fact that randomly generated profiles are as likely to correlate positively as they are negatively with target users. This means that the attack profiles that contribute to predictions are likely to have no significant net effect.

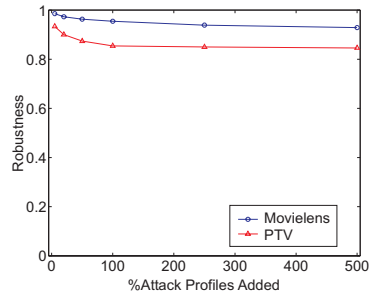

Fig. 1. Scenario 1: Robustness Degradation following Random Attack

5.2 Scenario 2

In this experiment, a random product nuke attack is performed. The attack profiles include the item to be nuked (set to the minimum rating), with the remainder of the items generated randomly as before. The number of attack profiles added is equal to the number of users in the original dataset. POA calculated according to Eqn. 8 are 0.98 for `Movielens` and 0.89 for `PTV`. In addition, the distribution means of the pre- and post-attack predictions are almost identical for both `Movielens` (3.18 and 3.33 respectively) and `PTV` (2.17 and 2.15 respectively), indicating that this type of random attack has little effect on predictions. As before, this is explained by the fact that randomly generated profiles are as

likely to correlate positively as they are negatively with target users, thereby having no significant overall effect.

5.3 Scenario 3 - Experiment 1

In this experiment, a focussed product nuke attack is carried out. The attack proceeds by generating a 3 item attack profile consisting of the item to be nuked together with the two most popular items in the database. The identical attack profile is repeated a number N times. Note that this attack can only be successful for those users that have rated the two most popular items (35% of all users for `Movielens` and 18% for `PTV`). Therefore, we measure POA according to Eqn. 8 only over those users who have rated the items in question.

Fig. 2 presents POA against the size of the attack, N. For `Movielens`, POA falls to 0.5 when $N = 60$, showing that the attack was successful in forcing one half of all predictions to the minimum rating. Also the distribution means of pre- and post-attack predictions (3.24 and 2.34 respectively) further indicate a successful attack. For `PTV`, the attack is not successful, with POA falling only slightly below the baseline level (i.e. the POA at $N = 0$, corresponding to those user-item pairs for which predictions are already at the minimum rating). This is explained by the fact that the average number of attack profiles that contribute to predictions is significantly less for `PTV` than for `Movielens`. For example, consider the `Movielens` database, where, at $N = 60$, 73% of all profiles that contribute to predictions are attack profiles, as opposed to only 45% for `PTV`. Here we see the heuristic of assigning a higher rating to the most popular item in the attack profiles breaking down for `PTV`. This is due to the sparsity of the dataset, which is an order of magnitude more sparse than the `Movielens` dataset.

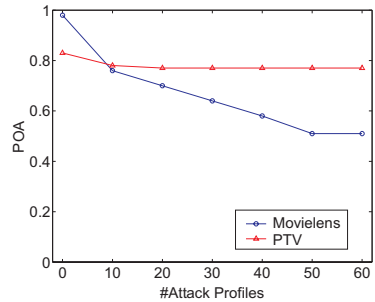

Fig. 2. Scenario 3 - Experiment 1: POA for Product Nuke Attack

5.4 Scenario 3 - Experiment 2

Again a product nuke attack is carried out on the `Movielens` and `PTV` databases, but this time the goal is to attack a *group* of items at once. In this case, N distinct

attack profiles are generated, one for each of the items to be attacked. All attack profiles consist of 3 items, chosen in the same way as in Experiment 1 above. Thus each profile contains one of the items to be attacked, set to the minimum rating, together with two other items. If the attack profiles are independent of one another - and we can achieve this by ensuring that none of the items we wish to attack appear together in any of the attack profiles - then we can obtain similar results over the whole group as we achieved for the single-item attacks described in Experiment 1. This means, for example, that it would be possible for authors or publishers to demote or promote multiple titles in a single attack.

5.5 Scenario 3 - Experiment 3

It is clear that items which are rated identically over the entire database present a grave security risk from the point-of-view of robustness. This point is confirmed in this experiment, in which the database is initially modified so that two items have identical ratings over all users. These two items can now be used to generate very effective attack profiles to nuke (or push) any selected item in the database. We repeated the experiment outlined in Experiment 1 above on our modified database, using 60 attack profiles. The result was that POA over all items fell to 0.21 for `Movielens`, and 0.04 for `PTV`. The change in pre- and post- distribution means is also very significant - for `PTV` the means are 2.19 and 0.52 respectively; for `Movielens` the means are 3.17 and 1.03 respectively. Fig. 3 shows a histogram for `Movielens` of the recommendations made for all attacked user-item pairs, pre- and post-attack, which clearly illustrates the successful nature of the attack.

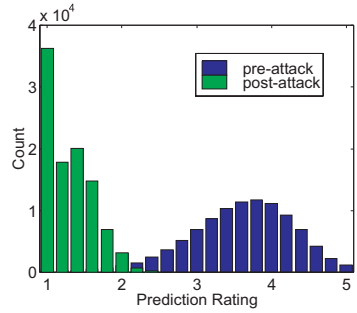

Fig. 3. Scenario 3 - Experiment 3: Histogram of Pre- and Post-Attack Predictions

Note that although it may seem unlikely that a real database will contain such agreement across all users for any item, it is reasonable to expect that there are groups of users who will agree on some common sets of items. This experiment shows that if the goal of an attack is simply to target a particular user group, then a simple and effective strategy is to seek those items on which the group agree and use them to build attack profiles.

6 Conclusion

In this paper we have introduced a new and important measure of performance, namely robustness, which has not previously been considered in the context of recommendation systems. We have demonstrated that a robustness analysis can highlight weaknesses in a system which do not become apparent by simply analysing recommendation accuracy. We have presented several attack scenarios and have shown that effective attack strategies can be devised to degrade the performance of the system. Future work will focus on robustness analysis of more sophisticated collaborative filtering systems and will investigate ways of securing systems against the types of deliberate attack that have been outlined in this paper.

References

1. C. C. Aggarwal, J. L. Wolf, K.-L. Wu, and P. S. Yu. Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In *Knowledge Discovery and Data Mining*, pages 201–212, 1999.
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 1998.
3. M. Condliff, D. Lewis, D. Madigan, and C. Posse. Bayesian mixed-effects models for recommender systems. In *Proceedings of the ACM SIGIR Workshop on Recommender Systems: Algorithms and Evaluation, 22nd Intl. Conf. on Research and Development in Information Retrieval, 1999*, 1999.
4. D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, December 1992.
5. P. Groot, F. van Harmelen, and A. ten Teije. Torture tests: a quantitative analysis for the robustness of knowledge-based systems. In *European Workshop on Knowledge Acquisition, Modelling and Management (EKAW'00)*. LNAI Springer-Verlag, October 2000.
6. J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the SIGIR*. ACM, August 1999.
7. C.-N. Hsu and C. A. Knoblock. Estimating the robustness of discovered knowledge. In *Proceedings of the First International Conference on Knwledge Discovery and Data Mining*, Montreal, Canada, 1995.
8. C.-N. Hsu and C. A. Knoblock. Discovering robust knowledge from dynamic closed-world data. In *Proceedings of AAAI'96*, 1996.
9. http://movielens.umn.edu/.
10. http://www.changingworlds.com/.
11. P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J.Riedl. An open architecture for collaborative filtering of netnews. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*. ACM, 1994.
12. B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *ACM Conference on Electronic Commerce*, pages 158–167, 2000.