Shilling Protection Using Network Topology Methods

John Clow and Ran Gross

Abstract

This paper covers several methods of tackling the problem of shills on a network by trying to identify, filter, or mitigate their influence they have on the sign prediction of a network. We have covered several methods including the use of Collaborative Filtering, general vs personal PageRank, and Random Walks and have found that Random walks was the most resilient to shill attacks.

1. INTRODUCTION

Whether for personal gain or malicious intent, when there is something that works, there will be someone who wants to either take advantage of it, or to break it. This is true for the realm of online store purchasing as well, where users rely on reviews to determine the best product for them. However as retailers have been hiring fake reviewers, and incentivizing other reviewers to be shills (people who rate a product highly to deceive others); these raw reviews and review scores have become less relevant to a user. Another problem we have is that different users care about different features of a product, so a general rating will most likely not be relevant to a particular user. We seek to use new methods for interpreting local network topology to solve both problems. In this field, Collaborative Filtering has long been the standard for improving predictions. However, most focus has been put on item-to-item similarity, and also does not consider the effects of shilling on the accuracy of results. Therefore, we want to test out the collaborative filtering algorithm on networks with shills and also create new algorithms that use local network topology and are more resistant to shills.

Despite being an old technique, Collaborative Filtering is still a major part of rating predictions or various algorithms. For example, the Belkor Recommendation system that won the Netflix Prize in 2009 used Collaborative Filtering as the local component in its system. Additionally, there is the Amazon Item-to-Item collaborative filtering paper described later, written in 2010, which uses Collaborative Filtering in more recent years. Although Collaborative Filtering as a stand alone is not as common, and it is an old technique, it is still used as a major component of many recommendation systems today. However, after seeing how vulnerable it is to shilling attacks, we decided that we needed an alternative method for detecting shilling. Therefore, we used various methods to detect shills directly, including Similarity scores from node random walks, Personal Pagerank of trusted nodes vs the entire graph, as well as neural network models for shill detection.

2. Related Works

2.1. Item-to-Item Collaborative Filtering

Amazon has conducted research on how to find high quality recommendations for their users using a users purchasing and rating history [5]. They focused on maintaining quality while scaling with the customer-base and the catalog size. Their approach involved using item to item similarity to find recommendations rather than the customers similarity to other customers. Their algorithm goes as follows:

```
For each item in product catalog, $I_1$
  For each customer $C$ who purchased $I_1$
    For each item $I_2$ purchased by
       customer $C$
      Record that a customer purchased $I_1$
       and $I_2$
  For each item $I_2$
    Compute the similarity between $I_1$ and $I_2$
```

This paper relates to the class topics relating to signed graphs and edge prediction where we consider whether a user bought a product or rated it positively as positive and negative otherwise. From this they predict what an edge that doesnt exist yet should look like.

To calculate the similarity of two items in traditional methods, they take the cosine of the item vector as follows:

$$Similarity(\overrightarrow{A}, \overrightarrow{B}) = cos(\overrightarrow{A}, \overrightarrow{B}) = \frac{\overrightarrow{A} \cdot \overrightarrow{B}}{||\overrightarrow{A}|| * ||\overrightarrow{B}||}$$

This paper fails however to address possible similarity between neighbors of neighbors. While direct neighbors have a good chance of being similar, it does not explore if, and to what network distance can we cannot deem items to be similar.

Additionally, this papers goal is finding item recommendations rather than guess a person's rating. While the concepts are similar and related, the approach the paper takes we believe takes away from the ability to gain rating. Their concern lies with viewing item similarity and recommending items similar to items previously bought by the user. In addition, their dataset contains purchased items as well as items rated, while the focus of our project is reviews, for which the former might not even be applicable.

The paper also does not show supporting data and comparisons of their algorithm with respect to the old methods they described of collaborative filtering, clustering, and search based methods. It simply provides a brief analysis of the big-O improvement in their method as well as an analysis of the relevancy of the results found. They give some numbers indicating the size of the dataset they tested, however they do not list any numerical evidence to indicate improved speed or relevancy or recommended items. They do elaborate a bit though on the difference on what parts of the algorithms can and cannot be computed online and offline with each one.

This paper provides good insight on different approaches to finding similarity and potential signed edges between nodes by looking at the items rather than the more traditional method of looking at people. It would be interesting to take this approach further by potentially using negative nodes on top of positive ones and use a concept similar to the behavioral theory where the enemy of my enemy is my friend, or in this case what someone with opposing tastes dislikes, I will like.

It would also be compelling to expand this view of item to item comparison to see if it can improve the relevancy of the items recommended rather than just improving the online time required to search for the items.

2.2. Empirical Analysis of Predictive Algorithms for Collaborative Filtering:

The paper analyzes the effectiveness of a variety of recommender systems that rely on local graphs, including collaborative filtering methods using correlation and vector similarity, as well as extensions such as default voting and Inverse User Frequency.[1] Then the paper compared the performance of these algorithms on the datasets MS Web, Neilson network television data, and EachMovie. Using ANOVA, it compared the performance of various algorithms when given a set of cases to choose from. It found that in each dataset, a Collaborative network with inverse user frequency was able to predict the score result with high scores, only the bayesian network had higher predictive scores.

One of the problems with the dataset is that it doesnt give a detailed description of their baseline. The only thing they write is that it is a zero-order collaborative filtering system. In addition, they do not ensemble various predictors into a potentially higher-scoring predictor. By not doing this, they have failed to gleam valuable insights on how correlated the errors of the various prediction methods.

In addition, they ignored that there are relatively few votes per person in the MS Web dataset. There was a mean of 3.95 page visits (aka implicit votes) per user, and a median of 3 page visits per user. With so little data per person, it is questionable whether or not Collaborative Filtering was actually effective because it found general correlations for users, or it was scoring well because it was able to successfully correlate a small subsection of the user base.

They also fail to explain why the different tasks, Given2, Given5 and Given10 were expected to show differences that would be meaningful to analyze. If there was more analysis on why the authors thought that these tests would create different sets of results, then they could justify showing all of the results. In this case, however, the results look similar enough that I would just show the Given5.

We believe that by picking through why the models failed, one could figure out ways to improve the accuracy of the various models. With more analysis, I bet that they would be able to find ways of refining the algorithms to maximize the predictive powers of them. In a similar vein, they could analyze how other predictive methods would fare compared to the presented methods, such as neural networks for analyzing the graph relationships.

Finally, they can run the models on modern datasets to see if the trends still hold. All of the datasets that they used were very small. The largest, EachMovie only had 4119 users with 46 votes per user. The other two datasets were an order of magnitude smaller. As we have large, modern datasets, such as the beer rating dataset that we have with millions of datapoints. In addition, some of the models wouldnt be able to cope with the dataset. Even for the EachMovie dataset, they were only able to train the Bayesian model on the top fifth of the movies, as otherwise the model wouldnt train in a reasonable amount of time.

2.3. Rating Aggregation in Collaborative Filtering Systems

This paper reviews the effect of rating aggregation, one of which is a shilling attack, on a Collaborative Filtering based recommender system. [2] It analyzes the effect of shilling attack on two types of similarity functions as well as the difference between predicting by mean, median, and mode. In the paper, they arrive at the conclusion that the median is more resilient to attacks than the mean, and that mode is the most resilient of the three.

This paper looks at the problem statistically, and uses those methods to try to limit the effect of outliers. However, it assumes that the number of people shilling is less than the number of people who were honest, which would not be the case if the person originating the shilling has many resources. Additionally, the method they proposed is heavily influenced by their tie breaking rule for both the median and mode methods. Since they only showed it for a single data set, and a relatively small one too, it could also be that their solution is further affected by the nature of the items being recommended. This paper proposes an interesting insight on possible shilling robustness.

3. Algorithms Used

3.1. Baseline

Our baseline includes two approaches. The first approach uses a majority rule to predict the vote. That is, which ever vote occurs more often, will be the predicted result. The second approach we attempted is the classic user-user Collaborative Filtering. [1] For this approach, we calculate for every single person a personal bias that is the mean of all their votes so far.

$$\bar{V_p} = \frac{1}{|I_p|} \sum_{i \epsilon I_p} v_{p,i}$$

where $\bar{v_p}$ is the mean rating user p gave, I_p is the set of items user p has reviewed, and v_p, i is the review user p gave item i. This gives us a general idea of how user p tends to review.

We then calculate a similarity weight between the user and any user who has reviewed the item in question and has other items in common with the user. This is done to make sure that every user we consider has either reviewed the item in question and has some similarity with the user in question, and otherwise they would not contribute to the prediction as their votes have don't tell us much since we can't tell if they are similar or different from our user. This will help us find users that are more relevant to our user and give a lower weight to those who are different.

$$w_{p,q} = \frac{\sum_j (v_{p,i} - \bar{v_p})(v_{q,i} - \bar{v_q})}{\sqrt{\sum_j (v_{p,i} - \bar{v_p})^2 \sum_j (v_{q,i} - \bar{v_q})^2}}$$

where $w_{p,q}$ is the similarity weight between user p and user q.

Finally, we can calculate the prediction. To do this we calculate the weighted average of all the users who reviewed the item, add in the user bias, and then take the sign of the result to give us the final prediction. To deal with the case of a neutral result, we break ties, predictions of 0, by saying they are positive.

$$p_{p,i} = \bar{v_p} + \frac{1}{\sum_q w_{p,q}} \sum_q w_{p,q}(v_{q,i} - \bar{v_q})$$

Where $p_{p,i}$ is the prediction made for user p on item i and the summation is over all non-zero weights.

The algorithm used to run this was as follows:

```
// Preprocess bias for all users
For every user in network:
  Bias[user] <- mean(out-edge weights)

//Prediction step (dependant on item)
Predict(user, item):
  SimilarUsers <- {}
  For every item reviewed by the user:
    SimilarUsers <- SimilarUsers ∪ reviewedItem.Pred
  For every user in SimilarUsers:
    weight(user, similar User) <- wq,p
    // as shown in the formula above

  predecessors <- item.Predecessors
  Prediction <- Bias[user]
  For node in predecessors:
    $Prediction + = 1/(sum(w_{p,q})) * w_{p,q}(v_{q,i} - \bar{v_q})$
Return Prediction
```

3.2. PageRank Approack

After seeing how vulnerable the baseline was to shills, we decided that we should investigate multiple methods of detecting shills directly.

The first approach uses the user's general PageRank and compares it to the personalized PageRank they get using a trusted set as the teleport set. We use the ratio of the personalized PageRank in comparison to the general PageRank to determine whether a user is a shill or not. The intuition behind this method is that shills will have some targeted item that they are trying to either raise or lower its value. However, the rest of their reviews to items other than the targeted set, would be relatively random. Therefore although they might have a good general PageRank due to all of them reviewing the same targeted item, the personalized PageRank they get would be much lower because the trusted set reviews the common item much more infrequently than the shills.

In this problem we treat it as if we already have a known trusted set. This is reasonable as in the real world, companies have people that they already know, and know aren't working as part of a set of shills.

3.3. New Approach by Random Walk

The second approach we attempted is through a random walk. In this method, we look for every user at the similarity of the items the reviewed. We then compare, using multiple methods, the item-to-item similarity as an indicator of how likely is the user a shill. Our comparison methods include the mean and variance of the similarity scores.

The intuition for the mean here is that users will generally use and rate items that have some relation or similarity to one another. On the other hand, a shill would only care about the one item it is targeting. Thus all other reviews it

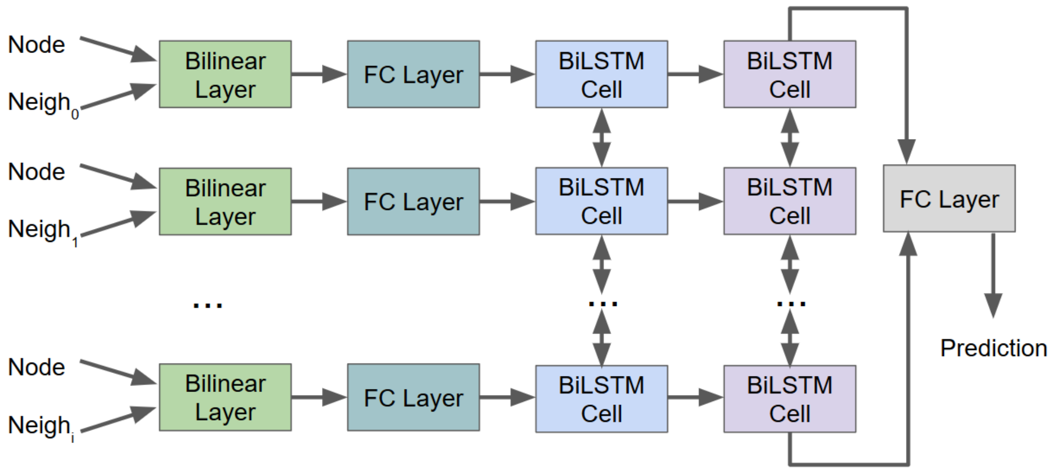

Figure 1. A diagram of our neural network implementation. All cells with the same color share the same weights.

gives will not have the same ”human” pattern or similarity of the items reviewed.

The intuition for using the variance here is that users will either rate items that are very similar to one another, so they will have fairly low variance in the item-to-item similarity. On the other hand, a shill might will have a more random pattern where some items might be similar to one another, while others vary wildly. Especially since a shill will often not rate a competitor item, or rate it opposite to the target item, causing similarity to differ a lot.

To deal with the potential bottleneck of having to calculate the item-to-item similarity of every item pair that every user reviewed, we use a random walk. This lowers the run time per node from O(d^2) to O(d), where d is the user's total reviews, or node degree. To do this we walk through each item the user reviewed twice. For each item, we check it's similarity with another item that has not been calculated yet. We then take that item and calculate the similarity with another item that has not been calculated yet. We repeat this until all items have been gone over twice. The first and last item are reviewed with each other as well at the end.

To calculate the actual value of the item-to-item similarity, several methods can be used. One such method is using the vector similarity between the items that a reviewer reviews, while another is to use node2vec cosine distance between items.

This method has a couple more levels of depth in its search for shills than Collaborative Filtering. This is because in order to find the legitimacy of a user, we use the item-to-item similarity, which can be calculated in multiple ways, and after producing that, we then connect it to users who reviewed these items as a similarity metric. As an extension, it is also possible to use the predicted shills to locate targeted items and use those as an additional metric to adjust the prediction of who is and isn't a shill, increasing the overall accuracy.

Once the random walk is over, we calculate the mean or variance of the similarity between the items that the user reviewed. For the mean, we predicted that the correct metric would be that a user is highly probable to be a shill if it is under a certain threshold. As will be discussed in the results section, we found that the opposite was actually true. Thus, instead of checking if the mean is below a certain threshold to signify a shill, we checked if it is above the threshold. For the variance, if this variance is above a certain threshold, we say that this user is highly probable to be a shill.

The algorithm goes as follows for mean:

```
for each user:
  for each step of random walk:
    calculate item-to-item similarity
  calculate the user's review mean
  if the mean > threshold:
    user is a shill
  else:
    user is not a shill
```

For Variance:

```
for each user:
  for each step of random walk:
    calculate item-to-item similarity
  calculate the user's review variance
  if the variance > threshold:
    user is a shill
  else:
    user is not a shill
```

Once we identify the shills, we go ahead to remove them from the graph and do the same collaborative filtering as before to predict the sign of the edge of interest.

3.4. Detecting Shills through Graph Neural Networks

After building the random walk method, we thought that using a neural network to predict which users are shills. In many other graph models such as Node2Vec, the makers assume that they have known labels for a subset of the graph. In our case, we do not need to assume that. Instead we propose that as long as we know the dynamics of shills, we can simulate shills in our network and train it to detect shills in the real network where there are no prelabeled nodes. Therefore, we propose that we will train on networks that have normal users and some simulated shills added and labeled. We built multiple such graphs, and each node is labeled with whether or not it is a shill.

The novel neural network works as follows. We initially calculate embeddings for all of the nodes in each graph through Node2Vec [3]. Then we keep those embeddings fixed (as we can't modify them on the dev and test set), and feed the node and each neighbor to the neural network. Afterwards, we have 5 layers, each with a ReLU activation function in between as seen in figure 1. The first layer is a bilinear layer, taking in the node and a neighbor and combining them. Then the second linear layer allows for higher level features to be extracted. After that we have two BiLSTM layers to incorporate information from various neighbors. Finally, we take the final hidden state from the BiLSTM and feed it into a fully connected layer to reduce the features to a score. This score is run through a sigmoid funciton to normalize it between 0 and 1 where 1 represents that the network is certain the node is a shill.

Due to time constraints, we weren't able to get a model working. We believe that such a model would work because it would not only be able to derive the same information on a random walk of neighbors due to the LSTM structure, it would also be able to derive deep structural information from the node2vec vectors of the neighbors as well as the distance in different dimensions between the node and its neighbors. This high amount of information should allow for a neural network that is highly accurate.

4. Data Collection

We have been using two datasets, Epinions [4] and Amazon Movie Reviews [6]. For the Amazon dataset, we have turned it into binary 1, -1 edge weights by mapping a review of 5 to 1 and anything less than 5 to -1. We also used the Epinions dataset because we wanted to see if the techniques would generalize to humans rating humans. For Epinions, we allow each person to both be a rater and a ratee (but not both for the same edge that we are estimating).

To make the networks easier to process through, we went through a process of shrinking the networks to a more manageable size while keeping its basic properties, such as average node degree and clustering, the same. To do this, we choose a random node, and use a breath first to walk through the network, adding any node we encounter to the new reduced network. We then test over the network properties to make sure that they were not altered in any significant way. If they weren't, we use the new graph. Otherwise, we try a new node to generate a new reduced network.

On top of that, we proposed a method to add shills to the networks, and see if the shills influence the ratings of the network. Our current network has hundreds of shills targeting the same item. They all gives positive ratings to the target of the shilling and gives average ratings to all other items (picked randomly).

5. Results

To test our shill detection algorithms, we created shill infested networks from the shrunken versions of the Epinion and Amazon movie reviewer networks as described in the Data Collection section. We then ran our algorithms on these networks while keeping track of the which were actually shills, which weren't shills, which were predicted as shills, and which were predicted as not shills. Using these values, we calculated the proportion of shills detected, precision, recall, and f1 score.

5.1. Collaborative Filtering

To test our code, we took a subset of the reviews already in the network, ran our prediction process on the user-item pair from that review, and compared the predicted value to the original review. We ran out algorithms on the Epinions dataset, which resulted in the following:

```
Majority Vote:
  precision = 0.8998795802511612
  recall = 0.976370016425265
  f1 score = 0.9365656276296707
  accuracy 0.8853769006793918
  Average Error = 0.348653912025

Collaborative Filtering:
  precision = 0.9216101694915254
  recall = 0.9841628959276018
  f1 score = 0.9518599562363238
  accuracy = 0.912
  Average Error = 0.3180035239
```

As expected, Collaborative Filtering produced more accurate results on the raw epinions data set than the Majority vote approach. This makes sense as in the Collaborative Filtering we take into account the opinions of similarly minded people in addition to a blind overall critique of the item. Additionally, we can see that in the absence of shilling, Collaborative Filtering does a relatively good job of predicting peoples reviews. Next, we ran both methods on a smaller

	PageRank
Average Precision	1.0
Average Recall	0.95455930359086
Average F1 Score	0.987278911564625
Average Accuracy	0.96484004127967

Table 1. The statistics for PageRank shill detection algorithm.

set of edges in the epinions network before and after adding targeted shilling attacks to it. Due to time constraints, we had to use a slightly smaller set for these preliminary results for this test.

```
Majority vote, Original:
precision = N/A \# No positive predictions
recall = 0.0
f1 score = N/A \# No True Positives
accuracy = 0.9
Average Error = 0.533333333333

Majority Vote, Shilling Network:
precision = 0.1
recall = 1.0
f1 score = 0.181818181818182
accuracy = 0.1
Average Error = 1.74198473282

Collaborative Filtering, Original:
precision = 0.3333333333333333
recall = 1.0
f1 score = 0.5
accuracy = 0.8
Average Error = 0.464549671466

Collaborative Filtering, Shilling Network:
precision = 0.14285714285714285
recall = 1.0
f1 score = 0.25
accuracy = 0.4
Average Error = 1.23273314486
```

As we can see, Collaborative filtering, as is, is weak to shilling attacks, as the percent of correct predictions fell by roughly 40%. However, it is still much more resilient to shilling attacks than Majority Vote, which went from 90% correct to 10% correct. Note that we could only test a few edges, as we are limited to the number of people who actually reviewed an item.

5.2. Filter by PageRank

For page rank, we can see the following results:

This shows that the ratio of a personalized PageRank over a trusted set to a general PageRank makes for a great indicator for shills. However, we also know that the result

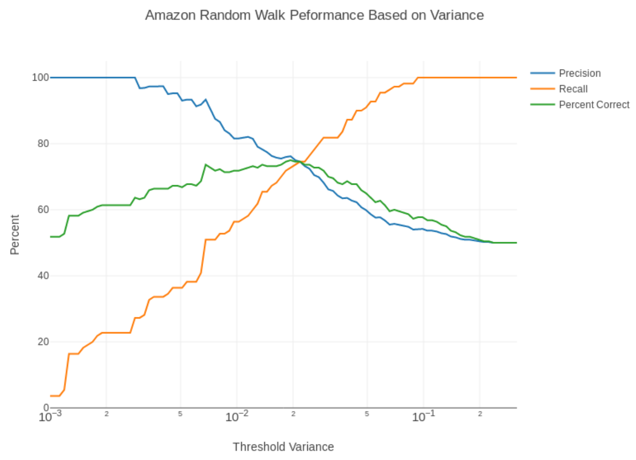

Figure 2. Graph of the accuracy, precision, and recall as the threshold is changed.

of the algorithm is extremely dependent on the size of the trusted set. If the trusted set is too small, other legitimate nodes are marked as shills. If the trusted set is too large, no shills are marked as shills, and the algorithm is useless. We believe that further refinements of the PageRank algorithm would yield better results.

However, we noticed that this method wasn't as stable as the others we listed. Although it worked really well for the epinions network, it yielded that nearly all users were shills for the amazon network, where out of 1718 users, 1616 were detected as shills. We believe that this is caused from the bipartite nature of the amazon movie network, as items cant link back to users. When we change the PageRank to run as if the network is undirected though, we get a similar result. This time however, it is mostly because the network is very highly connected. Therefore, the PageRank is distributed highly across all users, and items, thus giving mostly just the teleport set and its close neighbors high PageRank While the majority of the graph very low scores regardless of whether it is a user or shill.

5.3. Filter by Random Walks using Mean

Next, we discuss the results of the random walks using the mean of the similarity between the items a user has reviewed. We can see some of the results in figures 2 and 3.

As we can see from figure 2, our accuracy stays at roughly 70% as the threshold changes. The recall drops as the threshold increases while the precision increases with the threshold. This shows us that at roughly threshold of 0.2656 is when our differentiation between the shills and users is the highest for the amazon movie network. If the threshold deviates too much from this value, then our labeling of shills starts to mislabel users.

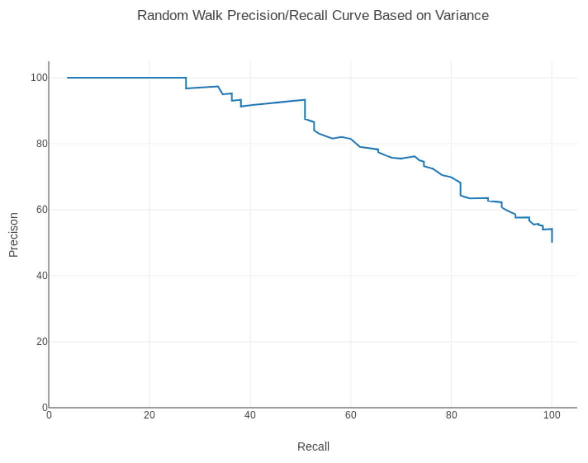

Figure 3. Graph of the precision vs. recall as the threshold of the shill detection is changed

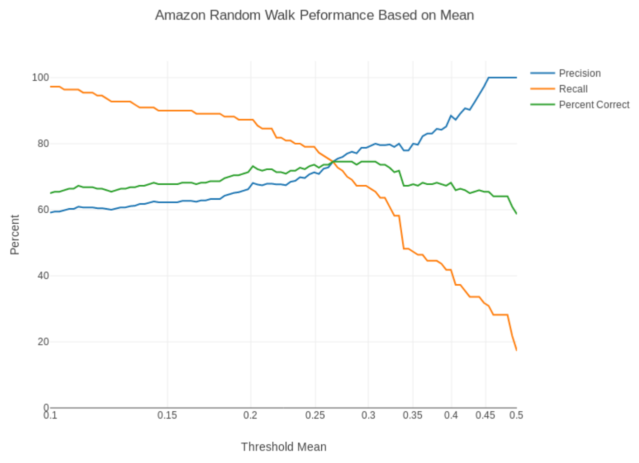

Figure 4. Graph of the accuracy, precision, and recall as the threshold is changed.

As we can see from table 2, the mean of the similarity between the random walk of items of a user does a relatively good job of indicating whether a user is a shill or not. We have a roughly 75% of shills detected, while having fairly low number of false positives and false negatives.

Although the results do match our expectations in that the mean is a good indicator, it follows a pattern opposite to our initial intuition. We predicted that the mean would be a good indicator since we thought that users would have exhibit high similarity between the items they choose to review. On the other hand, we thought that shills would have a low average similarity item scores since they will generally choose the items they review, other than the targeted item, by either reviewing similar items opposite to the targeted item, or just review different or random items altogether. However, the data shows that it is actually the other way around. Users have a tendency to have a much lower random walk similarity item scores than their shill counterparts. This gives us an interesting insight on user behavior. One possibility is that users tend to review the best or worst items of each type and just have many types of items they review. This would mean that the items they review will have low similarity since they only reviewed the one of each type that they felt strongly about. Shills, though, only had less of a pattern and just chose some items to look at. So the shill's mean item similarity was neither high nor low while the user's mean item similarity was exceptionally low.

5.4. Filter by Random Walks using Variance

Now, we discuss the results of the random walks using the variance of the similarity between the items a user has reviewed. We can see some of the results over all thresholds in figures 4 and 5.

We can see the results for the threshold with the highest

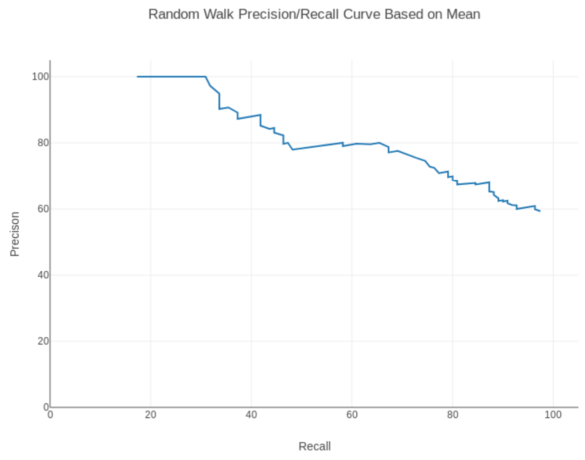

Figure 5. Graph of the precision vs. recall as the threshold of the shill detection is changed

	variance	mean
Precision	0.6746	0.7454
Recall	0.7727	0.7634
F1 Score	0.7203	0.7543
Accuracy	0.7441	0.7554

Table 2. The statistics for the threshold filter with the highest accuracy variance of the similarity.

accuracy in table 2.

As we can see, this method gives us relatively good shill detection for both the Epinions network and amazon movie reviewer network, with a roughly 80% and 70% detection rate respectively. Additionally, we can see that the precision, recall, and f1 score are also just as high. This shows that while we are detecting a nice proportion of the shills, we have a relatively low proportion of False Positives.

These results match our expectations. As expected, normal users have a patterns they follow in the items they review and generally only review items that all have high similarity, giving a much lower variance of similarity. On the Other hand, shills have a more varied similarity scores as some items they review have high similarity while others have very low similarity. Accuracy of this algorithm also makes sense as there could be users that have high variance just by having high interest in a handful of different item types. On the other hand, its possible for a shill to have all low similarity items, giving them a low variance just because all their similarity scores are low.

6. Conclusion

Detecting and combating shills through just network-based methods (no reviewing the user review text or other user based processing) is really difficult. Originally we thought that Collaborative filtering by itself would be a significant protection against shills, but it turned out not to be the case. Through the both the PageRank and the new random walk method, we are able to detect a significant majority of the shills, though it still needs much improvement. We also believe that using a neural network will yield great results, allowing for significant improvements in shilling detection.

7. Team Contributions

Both team members had roughly equal contributions.

References

[1] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, pages 43–52, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[2] F. Garcin, B. Faltings, R. Jurca, and N. Joswig. Rating aggregation in collaborative filtering systems. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 349–352, New York, NY, USA, 2009. ACM.

[3] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[4] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Signed networks in social media. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1361–1370. ACM, 2010.

[5] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, Jan. 2003.

[6] J. J. McAuley and J. Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *Proceedings of the 22nd international conference on World Wide Web*, pages 897–908. ACM, 2013.