Spammer Groups Detection Based on Communities Evolution

Robert Koon Ho Chan
r7chan@stanford.edu

Ye Xu
yexu15@stanford.edu

Yevhen Bochkov
ebochkov@stanford.edu

December 2016

Abstract

Online reviews have become increasingly important in this age of digital consumerism; however, dishonest reviews written by spammers skew product rating and mislead consumers' decision. Previous research to identify opinion spam mainly focused on review content using classification and text mining, but their applications are limited because spammers can disguise themselves as good reviewers by writing with proper grammar. Since then, network-based methods were introduced and have proven to be more flexible in spam detection. Nevertheless, these research focused on identifying individual spammer and they assumed review networks are static. In this project, an algorithm is proposed to identify spammer groups in continuous stream of data. This allows spam reviews to be detected and removed early, thus sellers can avoid losing money sooner and shoppers can have better shopping experience. The goal of this project is to produce the same result of spammer groups as Wang et al did in (Wang, 2015) but with dynamic data stream. Future research can be done to improve detection results.

1 Prior Work

In the past, people have tried to solve review spamming problem using supervised and unsupervised learning algorithms. Wang (2011) introduced the first network-based model to investigate identification of suspicious reviewers. He proposed an heterogeneous review graph concept to capture relationship among reviewers, reviews and stores being reviewed. The algorithm was proved to be effective in terms of computational complexity and precision (49 out of 100 specious reviewers are spammers). Concretely, the algorithm tries to estimate the trustiness of the reviewers, the honesty of the reviews and reliability of the products (or stores) iteratively, and the trustiness of reviewers will indicate whether a reviewer is a spammer after the algorithm converges. Akoglu (2013) proposed a more flexible method that exploits the network effect of review graph. Pairwise Markov Random Fields was used to construct the objective function, while score indicating good or spam review was estimated through maximization of network likelihood. Although the complexity of the algorithm was shown to be linear with the increase of network size, this would be an issue when the network size becomes extremely large. As the study of review spammers as groups rather than as individuals became popular, Wang (2015) presented the concept of loose spammer group, which resembles real-world spamming activities more closely. They derived a set of group spam indicators from structural as well as behavioral information of reviews, reviewers and products, and used a novel divide-and-conquer-based algorithm to generate loose spammer groups from a bipartite graph model.

2 Data Collection

2.1 Original dataset

We obtained our original dataset from Stanford course website for CS224W. This dataset was collected in the summer of 2006 by crawling Amazon website. It contains metadata for 548,552 products and information for 7,781,990 reviews (Leskovec 2007).

2.2 Labeled dataset

One challenge about building and evaluating review spammer detection model is that the dataset doesn't include ground truth labels that classify whether a review is created by spammer or not. Fortunately, we obtained from Wang a labeled dataset which he had three human evaluators identified 175 spammer groups out of the top 200 candidate groups that his algorithm generated. We will use this true labeled dataset to evaluate our model in this project.

2.3 Processed dataset

In this project, we only focus on the book reviews from our original dataset for two reasons: 1) our labeled dataset only applies to books, and 2) working with smaller dataset makes computation time more manageable. For each review, we extracted its product Id, reviewer Id and reviewed time from our original dataset. Some problematic data were removed, such as those with timestamp equal to 1970-12-01. This filtered dataset contains 278,217 different books, 661,954 reviewers and 4,591,303 reviews. Then, we assigned spammer or non-spammer label (and spammer group id if applicable) to each review based on Wang's labeled dataset. The final dataset contains 585 spam reviews and 71 spammer groups. Lastly, we sorted all reviews by their timestamp in ascending order, ready to be used in our algorithm.

3 Method

3.1 Group spam indicator

In this section, we will briefly review 8 spammer group indicators from (Wang 2015), which will be applied to our modified version of Wang's GSBP algorithm.

For this section we will use following notaion. R_g - set of reviwers in the group g. P_g is the set of target products in group g. A_p, L_p - dates when first and last appropriatry review was launched. To reduce contigency of small-sized groups, degree of uncontigency of group g is defined as $L(g) = \frac{1}{1+e^{-(|R_g|+|P_g|-3)}}$.

3.1.1 Review tightness (RT)

$$L(g) = \frac{1}{1+e^{-(|R_g|+|P_g|-3)}}$$

$$RT(g) = \frac{|V_g|}{|R_g||P_g|}L(g)$$

3.1.2 Neighbor tightness (NT)

$$NT(g) = avg_{r1,r2\in R_g}\frac{|P_r1 \cap P_r2|}{|P_r1 \cup P_r2|}$$

3.1.3 Product tightness (PT)

$$PT(g) = \frac{|\cap_{r\in R_g} P_r|}{|\cup_{r\in R_g} P_r|}$$

3.1.4 Average time window (TW)

$$TW_p(g,p) = \begin{cases} 1-\frac{L_p-F_p}{T}, & L_p-F_p \le T \\ 0, & L_p-F_p > T \end{cases}$$

$$TW(g) = avg_{p\in P_g} TW_p(g,p)L(g)$$

3.1.5 Rating variance (RV)

$$RV(g) = 2\Big(1 - \frac{1}{1+e^{-avg_{p\in P_g}S^2(p,g)}}\Big)$$

3.1.6 Product reviewer ratio (RR)

$$RR(g) = avg_{p\in P_g}\frac{|R_{g_p}|}{|R_p|}$$

3.1.7 Early review (ER)

$$ER(g) = avg_{p\in P_g}(g,p)L(g)$$

$$ER_p(g,p) = \begin{cases} 1-\frac{L_p-A_p}{T'}, & L_p-A_p \le T' \\ 0, & L_p-F_p > T \end{cases}$$

3.1.8 Group Size (GS)

$$GS(g) = \frac{1}{1+e^{-(|R_g|-3)}}$$

3.2 Algorithms

This project builds on top of the work by (Wang, 2015) in which a two-phase algorithm was proposed to detect spammer groups in a reviewer-product bipartite graph. First, a reviewer bipartition graph is constructed with nodes being reviewers and an edge indicates two reviewers have reviewed a common product. A weight is assigned to an edge for the number of common products two reviewers share. Then, all connected components are extracted from the reviewer bipartition graph for spammer group evaluation, where a set of spammer indicators are used to assign a group spam score to the extracted components. This evaluation process is done recursively by increasing the connectivity of the components based on the edge weight. This algorithm requires the scanning of the entire graph every time new data is added to the dataset even though only a subset of the graph is affected. Recognizing one of the properties of reviews is that it follows chronological order and that only data points surrounding the new data will need to be re-evaluated, we modified Wang's algorithm to process daily set of reviewer-product pairs to dynamically construct a bipartite graph as well as the reviewer bipartite graph projection. We applied the concept of time graph from (Kumar, 2005) to assign timestamp between a reviewer and a product, and we defined a time window to indicate when reviews become less important in detecting spammers and should be removed from the bipartite graph. We named our proposed algorithm as Online Group Spam detection via Bipartite Projection (OGSBP), and pseudo-code for it can be found in Algorithms 1 and Algorithms 2.

After we have implemented Wang's GSBP algorithm, we noticed our implementation generated a lot more spammer groups than Wang reported in his paper. Upon inspecting our results, we identified several false-positive patterns. One such pattern is that we flagged candidate groups with only two reviewers giving one review to the same product within a 30 day window and they did not review anything else. This would receive a very high spamicity score due to several high indicator values. To remove such cases, we implemented a filter to eliminate all candidate groups that have only two reviews. Another false-positive pattern is when two reviewers reviewed several popular products within a 30 day window but they each reviewed a large amount of products that are not in common outside of that window. Because GSBP is configured to analyze reviews within a 30 day period, it thinks that these two reviewers are in collusion. To handle this latter case, we generalized product tightness to consider all products reviewed by both reviewers and added a filter to remove all candidate groups having adjusted product tightness less than 0.90.

As we implement our online version of Wang's original algorithm (OGSBP), we realized our proposed modifications were not sufficient. One key difference between Wang's offline algorithm and our online variation is that all spammer groups are generated in one pass for the offline version while the online algorithm generates daily spammer groups. Therefore, we need a way to keep track of which groups are generated in the past, which groups are new and which groups have evolved.

4 Results and Findings

4.1 Data Analysis

The first step to generating spammer groups is to create a reviewer bipartition graph using Algorithm 1 in Section 3.2. In reviewer bipartition graph, nodes are all reviewers who gave review(s) to books. Two nodes are connected if the two corresponding reviewers have reviewed the same product within a specific time window. For analysis purpose, we created bipartition with two different time windows: 15 days and 30 days. That is, if the second reviewer created a review on a product within 15 (or 30) days after the first reviewer created a review on the same product, an edge will be created. To understand the structure of the reviewer bipartition graph, we plot the distribution of nodes degree (Figure 1) and size of weakly connected component (WCC) (Figure 2). Actually, we found bipartition graph of $\tau = 15$ vs. $\tau = 30$ are pretty similar, that is, number of reviewers (e.g. 571,760 vs. 661,954) and number of reviews (e.g. 12,353,560 vs. 17,937,846) as well as the distribution plots, so we only put distribution plot $\tau = 15$ here for illustration purpose.

We found that distribution of WCC size follows power-law distribution, except that a WCC has much

Algorithm 1 Online Group Spam detection via Bipartite Projection (OGSBP)

Input:
E: Daily set of edges of reviewer r1 and product p with timestamp t1
τ : review time window
Output: Spammer groups ordered by spamicity score
Description:
for each edge e1 of reviewer r1 and product p in E **do**
    for each edge e2 of reviewer r2 and product p in bipartite graph B **do**
        if e1.timestamp - e2.timestamp is less than time interval τ **then**
            Increment the weight of edge (r1, r2) in reviewer bipartition graph R by 1
        else// remove e2 as it is too old
            for each neighbor r3 of r2 in R **do**
                Decrement the weight of edge (r2, r3) in R
            end for
            Remove e2 from B
        end if
    end for
    Add e1 to B
    Mark g1-connected component g based on r1 in R as modified
end for
for each modified g1-connected component g in R **do**
    FindGroups(g,1)
end for
Output all spammer groups in descending order of group spam score;

Algorithm 2 FindGroups(g, k)

Input:
g: spammer group
k: k-connectivity threshold
Output: candidate spammer groups
Description:
if g.size > MAXSIZE **then**
    for each g^{k+1}-connected component g' in g **do**
        if g'.size > MAXSIZE **then**
            FindGroups(g', k+1)
        else if g'.spam $\geq \delta$ **then**
            Output g'
        else if g'.size > 2 **then**
            FindGroups(g', k+1)
        end if
    end for
else if g.spam $\geq \delta$ **then**
    Output g
else
    for each g^{k+1}-connected component g' in g **do**
        if g'.spam $\geq \delta$ **then**
            Output g'
        else if g'.size > 2 **then**
            FindGroups(g', k+1)
        end if
    end for
end if
Output all spammer groups in descending order of group spam score

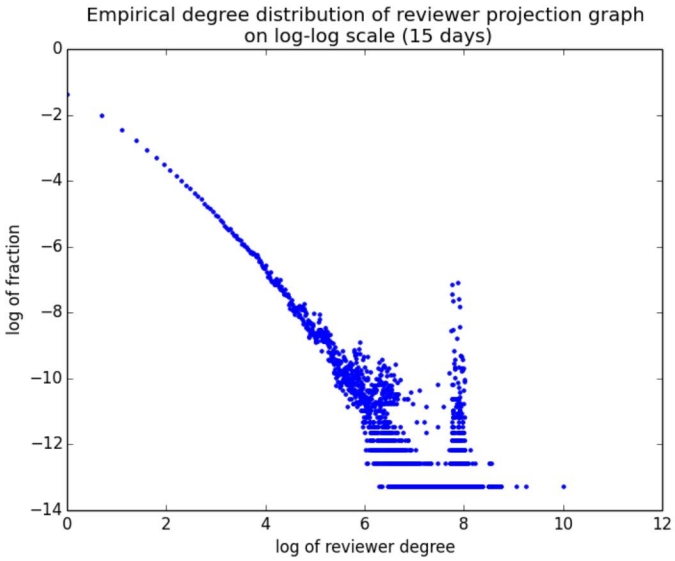


Figure 1: Degree distribution in reviewer bipartition graph when $\tau = 15$ days

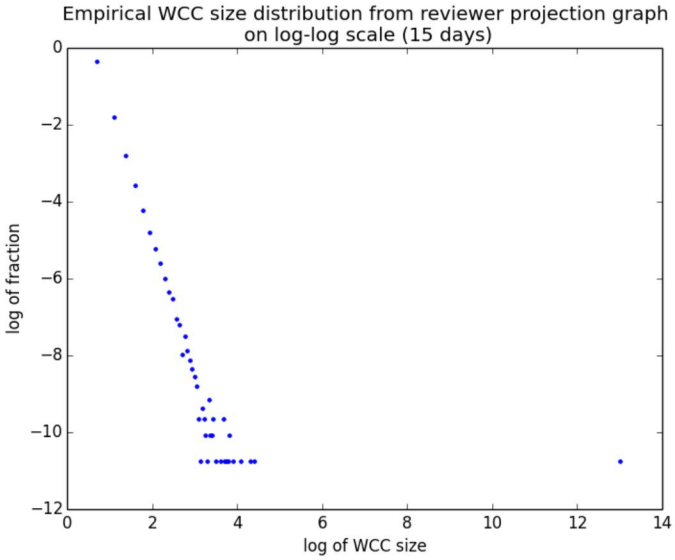


Figure 2: Size distribution of WCC in reviewer bipartition graph when $\tau = 15$ days

larger size than others. Most of the node degree also follows power-law distribution, but there is a spike in the tail where degree is very high. After investigating these reviewers, we found that each of them have many reviews (≥ 3000) to some popular books, and these popular books are reviewed by a lot of other users, that's why they have so many degree in reviewer bipartition graph. We will consider such reviewers as potential outliers, the impact to the model of these reviews will be evaluated in the remaining work of this project.

4.2 Implementation Findings

During the implementation of OGSBP, we encountered two issues when processing it with the Amazon dataset. One issue was that there were a few reviewers who wrote substantially large amount of reviews when compared with others. One reviewer wrote a total of 643185 reviews when the mean and median of

Table 1: Top 10 Reviewer With the Most Reviews

Reviewer ID	Number of Reviews
16	643185
265	154531
65	9589
414	5441
1713	3562
23	2055
6904	1651
1475	1535
5991	1508
2944	1469

reviews per person are 4 and 2 respectively. Table 1 shows the top 10 reviewers with the most number of reviews. In the end, we decided to blacklist reviewers who wrote 3000 reviews or more. Another issue we encountered was that there were reviewers who wrote reviews for the same product multiple times. The spammer indicators that Wang proposed in his paper did not handle duplicate reviews by the same person for the same product. Therefore, we had to remove all such reviews from the dataset and made sure every reviewer only has one review for any given product.

After we implemented OGSBP and ran it on Amazon dataset, we noticed its performance is far worst than the offline counterpart GSBP. Although the proposed pseudo-code looks straightforward, the implementation is anything but. Because OGSBP needs to maintain states across multiple days in order to evolve the in-memory graph, most of the inefficiency of the algorithm are related to states management. We identified three areas of OGSBP that contributed to the significant increase in runtime: 1) remove stale nodes from bipartite graph, 2) mark components as modified, and 3) detect duplicate candidate groups.

In order to deploy a scalable algorithm that can process constant stream of reviews, we need a way to purge old data as they become obsolete to free up resources for future data. In our proposed algorithm, we remove any reviews from the reviewer-product bipartite graph whenever their review timestamp is older than the review currently being processed by a factor of τ. (For our experiments, τ is set to 30 days.) In our OGSBP implementation, we used SNAP.TNEANet to construct the reviewer-product bipartite graph because we want to take advantage of its ability to manipulate graph easily; however, we found there are limitations to using SNAP.TNEANet and that it is not as performant as simple python dictionary, which we used to implement GSBP.

Another optimization of GSBP we proposed as we convert it to online mode is to compute spamicity score only for 1-connected components that are modified by the set of processed reviews, as suppose to studying all the 1-connected components in the reviewer-reviewer bipartition graph every time as suggested by GSBP. Unfortunately, there's no easy way to identify connected components, thus making it hard to keep track of which connected components have been modified.

Lastly, there is the challenge for OGSBP to detect which candidate spammer groups have already been identified, which groups have evolved, and which groups are newly identified between sessions. If we report every candidate group that the algorithm identifies, it will explode the size of candidate groups. This is not an issue with GSBP because GSBP assumes it knows the entirety of the dataset; whereas the dataset of OGSBP constantly increases.

Table 2: Comparison of outputs from different algorithm

	$\delta = 0.4$	$\delta = 0.6$	$\delta = 0.7$
MAXGROUPSIZE=5	163	163	168
MAXGROUPSIZE=10	165	165	170
MAXGROUPSIZE=20	165	165	171

Table 3: Comparison of outputs from different algorithm

	Original GSBP	Modified GSBP	OGSBP
Number of identified groups	2,253	1,475	1,687
Number of identified reviewers	9,010	5,868	6,746

4.3 Results

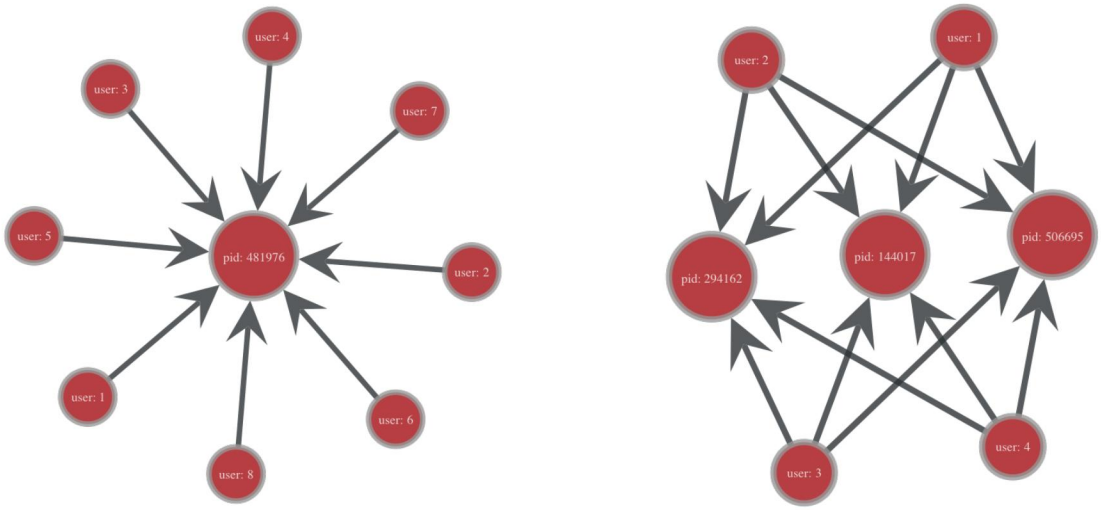
There are hyperparameters in the algorithm (e.g. MAXGROUPSIZE, δ). To save some time, we would like to understand their impacts on the final output on the relatively small dataset before running the algorithm on the entire networks.

We chose a small dataset with reviews which are created in Jun, 2005 to run the experiment with different value of MAXGROUPSIZE and δ. Number of output groups from each hyperparameter setting are compared to understand the impact of hyperparameter on the final output. From the results which are summarized in Table 2, the impact seems to be limited.

Next, we implemented GSBP algorithm proposed by Wang (2015). Then we implemented the modified GSBP and OGSBP as mentioned in Section 3.2. Here, we chose MAXGROUPSIZE=10, $\tau = 30$ and $\delta = 0.6$. Table 3 includes the summary of output from three algorithms. Modified GSBP and OGSBP produce fewer groups, this is because the modification described in 4.2 (e.g. adjusted product tightness and additional heuristic rule) has been implemented in both algorithm, to remove false positive groups. Moreover, we compared the output spammer groups identified by the modified GSBP and OGSBP. Over 99% of the identified groups by the modified GSBP are also included in the output by OGSBP, which implies that GSBP and OGSBP produce similar output groups.

The output groups were sorted by the decreasing order of spammity score (average value of spam indicators). We spent time to manually check top 50 spammer groups identified by GSBP and OGSBP repectively, and we found that all of them look like spammer groups (our judgement only relied on the meta data, since we don't have access to review content). Meanwhile, we discovered that pattern of spam campaign can be summarized into two general classes: one class is one product and multiple reviewers accounts were involved in a campaign, the other one is multiple products and reviewer accounts were involved in a compaign. We selected one example from each campaign pattern class and visualized in figure 3. In Figure 3a, 8 different spammer accounts reviewed a product (Id:181976) from 2005-06-24 to 2005-06-27. These eight accounts only reviews such one product, and do not have any review for other product in the entire dataset. Another type campaign as shown in 3b is that four spammers and three products are involved. Each spammer account reviews three products (in the figuare, the Id of products are 294162, 144017, 506695), these reviews were created on 2005-06-19 and 2005-06-22.

We also randomly checked groups with very low spammity score, we suspected that some may not be spammer groups, but we are not able to judge them at the moment with limited data



(a) Spam Campaign Pattern I

(b) Spam Campaign Pattern II

Figure 3: Two different spammer group patterns

5 Challenges

The biggest challenge to this project is the lack of labeled dataset of true spammer groups. Many previous work in detecting review spammers relied on human evaluation of the candidate sets their algorithms produced. This is a very time-consuming, labour-intensive step prone to human judgement. Fortunately, we were able to obtain a labeled dataset that Wang used in his work, which was constructed by three human evaluators analyzing the top 200 candidate sets generated by Wang's algorithm. However, his input dataset was different from the one we used in our work and we were not able to obtain his input dataset. To rectify this problem, we searched for the 175 identified spammers from Wang's labeled dataset in our input dataset and found 71 of them.

There is another challenge to this project which is that we cannot always verify the spammer groups we have identified. Although we obtained the labeled dataset from Wang, our algorithm can produce more candidate sets than the 200 that were analyzed. Also, some reviews have already been deleted on the Amazon website and our input dataset only contains metadata of reviews, not the review content; therefore, even if we want to manually evaluate the candidate sets our algorithm generates, we cannot easily do so.

Lastly, there is a challenge that applies to all review spammer detection algorithms and that is we cannot identify spammers who are not in any generated candidate sets. With over 600 thousand reviewers and 4 million reviews, it is impossible to manually evaluate all of them. This makes calculating metrics such as precision and recall very tricky.

6 Future Work

6.1 Parameters Tuning

A limitation of Wang's method is that weights of indicators are treated equally, and the difference in importance of each indicator were not studied yet. To solve the problem, we propose to applied supervised machine learning models to estimate weights. Concretely, we can choose sparmer groups that we have found

into training, validation, and test dataset. Meanwhile, we can generate non-spamer groups using a simlar approach as GSBP. To handle the highly imbalanced dataset, we downsample non-spammer groups with a fixed ratio of spammer vs. non-spammer groups for each of three datasets. Logistic regression (with L1 norm) will be used to estimate indicator weights. Hyperparamters will be determine, and use test dataset to evaluate the estimated weights performance. Bootstrapping can be used to understand variance of estimated weights, as well as overall prediction performance. If the prediction performance is promising, new weights will be employed in spammity score calculation in Algorithm 2.

7 Reference

G. Wang, S. Xie, B. Liu, P.S. Yu. Review graph based online store review spammer detection. In ICDM. 2011. 1242?1247.

L. Akoglu, R. Chandy, and C. Faloutsos. Opinion fraud detection in online reviews by network effects. In ICWSM, 2013

Z, Wang, T, Hou, D. Song, Z. Li and T. Kong. ?Detecting Review Spammer Groups via Bipar- tite Graph Projection.? The Computer Journal (2015): bxv068.