Fake reviewer group detection

Student Name: Sarthika Dhawan

Roll Number: 2015170

BTP report submitted in partial fulfillment of the requirements
for the Degree of B.Tech. in Computer Science Engineering
on April 15, 2019

BTP track
Research Track

BTP advisor
Dr. Tanmoy Chakraborty

Student's Declaration

I hereby declare that the work presented in the report entitled **Fake reviewer group detection** submitted by me for the partial fulfillment of the requirements for the degree of *Bachelor of Technology* in *Computer Science & Engineering* at Indraprastha Institute of Information Technology, Delhi, is an authentic record of my work carried out under guidance of **Dr. Tanmoy Chakraborty**. Due acknowledgements have been given in the report to all material used. This work has not been submitted anywhere else for the reward of any other degree.

..... **Place & Date: New Delhi, April 15, 2019**
Sarthika Dhawan

Certificate

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

..... **Place & Date: New Delhi, April 15, 2019**
Dr. Tanmoy Chakraborty.

Abstract

Online reviews play a crucial role in deciding the quality before purchasing any product. Unfortunately, spammers often take advantage of online review forums by writing fraud reviews to promote/demote certain products. It may turn out to be more detrimental when such spammers collude and collectively inject spam reviews as they can take complete control of users' sentiment due to the volume of fraud reviews they inject. Group spam detection is thus more challenging than individual-level fraud detection due to unclear definition of a group, variation of inter-group dynamics, scarcity of labeled group-level spam data, etc. Here, we propose DeFrauder, an unsupervised method to detect online fraud reviewer groups. It first detects candidate fraud groups by leveraging the underlying product review graph and incorporating several behavioral signals which model multi-faceted collaboration among reviewers. It then maps reviewers into an embedding space and assigns a spam score to each group such that groups comprising spammers with highly similar behavioral traits achieve high spam score. While comparing with five baselines on four real-world datasets (two of them were curated by us), DeFrauder shows superior performance by outperforming the best baseline with 17.64% higher NDCG@50 (on average) across datasets.

Keywords: information retrieval, natural language processing, graph theory

Acknowledgments

I would like to thank Dr. Tanmoy Chakraborty for his constant support and guidance. The project enabled me to find my interests in the field of Natural Language Processing and Graph theory and get a better insight into these areas. If it wasn't for his guidance and the involving discussions that we engaged in, the progress in the project would have been difficult. I thank him for showing me the way to do good research. I would also like to thank the members of Laboratory for Computational Social Systems and my family and friends for their constant support and encouragement.

Work Distribution

The major focus was on doing research and literature survey to devise a novel approach for finding candidate colluder groups and ranking them. Different approaches were tried to come with a pipeline that takes into consideration all the factors required for finding potential fake reviewer groups. Datasets were analyzed to see colluder behaviour in them in order to come with a generic model for the problem that outperforms the existing works in the domain. Extensive experiments have been done in Winter Semester of 2019 with modifications to the approach that was proposed in Monsoon Semester of 2018.

Contents

Chapter 1

Introduction

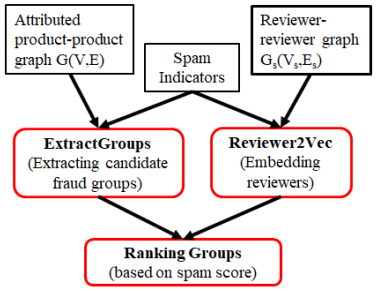
Nowadays, online reviews are becoming highly important for customers to take any purchase-related decisions. Driven by the immense financial profits from product reviews, several black-market syndicates facilitate to post deceptive reviews to promote/demote certain products. Groups of such fraud reviewers are hired to take complete control of the sentiment about products. Collective behaviour is therefore more subtle than individual behaviour. At individual level, activities might be normal; however, at the group level they might be substantively different from the normal behavior. Moreover, it is not possible to understand the actual dynamics of a group by aggregating the behaviour of its members due to the complicated, multi-faceted, and evolving nature of inter-personal dynamics. Spammers in such collusive groups also adopt intelligent strategies (such as paraphrasing reviews of each other, employing a subset of their resources to one product, etc.) to evade detection. Fig. 1.1(b) shows an example of such fraud reviewer group.

Previous studies mostly focused on individual-level fraud detection [1, 4, 5, 7, 8, 10, 14]. Few other studies which realized the detrimental effect of such collective activities detected groups simply based on Frequent Itemset Mining (FIM) [2, 17, 31]. They thus focused more on ranking fraud groups, paying less attention to judge the quality of the detected groups. Wang *et al.* [27] pointed out several limitations of FIM for group detection – high computational complexity at low minimum support, absence of temporal information, unable to capture overlapping groups, prone to detect small and tighter groups, etc.

In this paper, we propose DeFrauder [6][1] , a novel architecture for fraud reviewer group detection (see Fig. 1.1(a)). DeFrauder contributes equally to (i) the detection of potential fraud groups by incorporating several coherent behavioral signals of reviewers, and (ii) the ranking of groups based on their degree of spamicity by proposing a novel ranking strategy. Experiments on four real-world labeled datasets (two of them were prepared by us) show that DeFrauder significantly outperforms five baselines – it beats the best baseline by 11.92% higher accuracy for detecting groups, and 17.64% higher NDCG@50 for ranking groups (averaged over all datasets).

In short, our contributions are fourfold: (i) two novel datasets, (ii) novel method for reviewer

[1]DeFrauder: **De**tecting **Fraud** **R**eviewer G**r**oups . Codes and (partial) datasets are available in [22].



(a) Schematic diagram of DeFrauder.

Product: The Spotted Pig	Product: Buvette
Simple perfection, made with love, great service.... Sit at the bar if Chris is bartending, Teresa is also a great server... 2014-12-15 ★★★★★	Small place in the west village, loaded with charm and french ambiance. The food is great simple yet presented with sophistication and style. One of my favs! 2015-01-02 ★★★★★
Simple perfection, made with love, great service.... Sit at the bar if Chris is bartending, Teresa is also a great server... 2014-12-17 ★★★★★	Here is a simple review. Amazing food, fantastic ambiance! If you like the French way of life with smaller plates and great wine you need to go. 2014-12-25 ★★★★★
The burger is excellent and simple perfection. The waitress Teresa was fantastic. great attitude and friendly. 2014-12-15 ★★★★★	Classic French fare in a cozy setting. Great weekend brunch. 2014-12-31 ★★★★★
The place is expensive but I'm told to go back for the burgers. They were really good. Teresa was excellent - I wish I could have them again 2014-12-14 ★★★★★	Amazing food, fantastic ambiance! If you like the French way of life with smaller plates and great wine you need to go. 2015-01-01 ★★★★★

(b) Reviews and ratings of four spammers as a group on two products.

Figure 1.1: (a) Workflow of DeFrauder. (b) Coherent review patterns of a fraud reviewer group on two products. To evade detection, four reviewers in a group paraphrase reviews of each other keeping the underlying sentiment same.

group detection, (iii) novel method for ranking groups, and (iv) a comprehensive evaluation to show the superiority of DeFrauder.

Chapter 2

Related Work

Due to the abundance of literature [13] on online fraud detection, we restrict our discussion to *fraud user detection on product review sites*, which we deem as pertinent to this paper.

In user-level fraud detection, notable studies include Lim *et al.* [14] which proposed scoring methods to measure the degree of spamicity of a reviewer based on rating behaviors; Wang *et al.* [25] which developed a graph-based fraud reviewer detection model; Fei *et al.* [8] which exploited burstiness in reviews to detect fake reviewers. SpEagle [1] utilized clues from all metadata as well as relational data and harnessed them collectively under a unified framework. ASM [3] facilitated modeling spamicity as latent factor and exploited various behavioral footprints of reviewers. Wang *et al.* [26] argued that the dynamic behavioral signals can be captured through a heterogeneous reviewer graph by considering reviewers, reviews and products together. FraudEagle [1] spotted fraudsters and fake reviews simultaneously.

Few studies attempted to detect group-level fraud reviewer groups. GSRank [17] was the first method of this kind, which identified spam reviewer groups using FIM, and ranked groups based on the relationship among groups, products, and individual reviewers. Mukherjee *et al.* [17] argued that due to the scarcity of data, unsupervised method should be used to tackle this problem. Other studies largely focused on improving the ranking algorithm, ignoring the performance of the group detection method (they also used FIM for detecting groups) [2, 18, 19, 31]. Xu and Zhang [30] proposed an Expectation Maximization algorithm to compute the collusive score of each group detected using FIM. Wang *et al.* [27] argued that FIM tends to detect small-size and tighter groups. They proposed GSBP, a divide-and-conquer algorithm which emphasizes on the topological structure of the reviewer graph. GGSpam [29] directly generated spammer groups consisting of group members, target products, and spam reviews.

	GSBP	GGSpam	GSRank	DeFrauder
Temporal	✓	✓	✓	✓
Content			✓	✓
Graph	✓	✓		✓

Table 2.1: DeFrauder captures all three signals obtained from users attributes and underlying graph structure.

We propose DeFrauder that leverages all three signals – temporal pattern, content similarity and underlying graph structure to detect collusive fraud groups in review forums (see Table 2.1 for a comparison). Unlike others, DeFrauder contributes equally to both the components – group detection and ranking.

Chapter 3

Proposed Method

In this section, we explain DeFrauder which is a three-stage algorithm – detecting candidate fraud groups, measuring different fraud indicators of candidate groups, and ranking groups based on the group spam score (see Fig. 1.1(a) for a schematic architecture of DeFrauder).

3.1 Group Fraud Indicators

Here we present six fraud indicators which measure the spamicity of a group [17, 27, 29] by taking into account linguistic, behavioral, structural and temporal signals. All indicators are normalized to $[0, 1]$ – larger value indicates more spamming activity. Let R and P be the entire set of reviewers and products. Let $R(g)$ be the set of reviewers in a group g, and $P(g)$ be the set of target products reviewed by the reviewers in $R(g)$. Each reviewer i reviewed a set of products P_i. To reduce the contingency of small groups, we use a penalty function $L(g)$ which is a logistic function [27]: $L(g) = 1/(1 + e^{-(|R(g)|+|P(g)|-3)})$. We subtract 3 from the sum since we consider minimum number of reviewers and products within a group to be 2 and 1, respectively; therefore $L(g) \in [0.5, 1)$ [28].

(i) Review Tightness (RT): It is the ratio of the number of reviews in g (assuming each reviewer is allowed to write one review to each product) to the product of the size of reviewer set and product set in g.

$$RT(g) = \frac{\sum_{i \in R(g)} |P_i|}{|R(g)||P(g)|} \cdot L(g) \tag{3.1}$$

If significant proportion of people co-reviewed several products, then it may imply a spam group activity.

(ii) Neighbor Tightness (NT): It is defined as the average value of the Jaccard similarity

(JS) of the product sets of each reviewer pair.

$$NT(g) = \frac{\sum_{i,j \in R(g)} JS(P_i, P_j)}{\binom{|R(g)|}{2}} \cdot L(g) \tag{3.2}$$

If the product sets are highly similar, both the reviewers are then highly likely to be together in a collusive group.

(iii) Product Tightness (PT): It is the ratio of the number of common products to the number of products reviewed by all the members in g [27].

$$PT(g) = \frac{|\bigcap_{i \in R(g)} P_i|}{|\bigcup_{i \in R(g)} P_i|} L(g) = \frac{|\bigcap_{i \in R_g} P_i|}{|P(g)|} L(g) \tag{3.3}$$

Group members reviewing certain number of products and not reviewing any other products are more likely to indulge in fraud activities.

(iv) Rating Variance (RV): Group spammers tend to give similar ratings while reviewing any product. Let $S^2(p, g)$ be the variance of the rating scores of product p by reviewers in g. We take the average variance for all target products. The variance degree $[0.5, 1)$ is converted into spam score between $(0, 1]$.

$$RV(g) = 2L(g)\left(1 - \frac{1}{1 + e^{-\frac{\sum_{p \in P(g)} S^2(p,g)}{|P(g)|}}}\right) \tag{3.4}$$

(v) Product Reviewer Ratio (RR): It is defined as the maximum value of the ratio of the number of reviewers in $R(g)$ who reviewed product $p \in P(g)$ to the number of all the reviewers of p, denoted by $|Rev(p)|$:

$$RR(g) = \max_{p \in P(g)} \frac{\sum_{i \in R(G)} \{1 : p \in P_i\}}{|Rev(p)|} \tag{3.5}$$

If a product is mainly reviewed by the reviewers in g, then the group is highly likely to be a spammer group.

(vi) Time Window (TW): Fraudsters in a group are likely to post fake reviews during a short-time interval. Given a group g, and a product $p \in P_g$, we define the time-window based spamicity as

$$TW(g, p) = \begin{cases} 1 - \frac{SD_p}{T}, & \text{if } SD_p <= T \\ 0, & \text{SD}_p > T \end{cases}$$

$$TW(g) = \frac{\sum_{p \epsilon P_g} TW(g, p)}{|P(g)|} \cdot L(g), \tag{3.6}$$

where SD_p is the standard deviation of review time for a product p reviewed by reviewers in

group g. T is a time threshold (set to 30 days in our experiments as suggested in [29]).

Algorithm 1 `ExtractGroups`

Initialize:
  $CCgroups \leftarrow$ set() ▷ Set of candidate groups
  $G(V, E) \leftarrow$ Edge attributed graph
  $CSet \leftarrow \{\}$ ▷ Potential merged groups
$G, CCgroups \leftarrow$ `GroupDetector`$(G, CCgroups)$
Iterate:
  $G'(V', E') \leftarrow$ Attributed line graph of G
  $G', CCgroups \leftarrow$ `GroupDetector`$(G', CCgroups)$
  $G \leftarrow G'$
until $|E| > 1$
return $CCgroups$
function GROUPDETECTOR$(G, CCgroups)$
  for each isolated node v_i in G **do**
    $CCgroups.add(v_i)$ and remove v_i from G
  for each pair $(e_i, e_j) \in E \times E$ **do**
    if $a_i^e \subset a_j^e$ **then**
      if $\frac{\bigcap_{m \in a_j^e} P_m}{\bigcup_{m \in a_j^e} P_m} > 0.5$ **then**
        $CSet(a_i^e) = CSet(a_i^e) \cup a_j^e$
      else
        if $\frac{\bigcap_{m \in \{a_j^e \setminus a_i^e\}} P_m}{\bigcup_{m \in \{a_j^e \setminus a_i^e\}} P_m} > 0.5$ **then**
          $CCgroups.add(a_j^e \setminus a_i^e)$
  for each $e_i \in E$ **do**
    k=$CSet(a_i^e)$
    $CCgroups$.add(k) and remove k from G
  for each connected component c with $|c| > 2$ **do**
    $CCgroups.add(c)$ and remove c from G
  for each group $g \in CCgroups$ **do**
    if $CollectiveScore(g) <= \tau_{spam}$ **then**
      $CCgroups.remove(g)$
  return $G, CCgroups$

3.2 Detection of Candidate Fraud Groups

We propose a novel graph-based candidate group detection method based on the "coherence principle".

Hypothesis 1 (Coherence Principle) *Fraudsters in a group are coherent in terms of – (a) the products they review, (b) ratings they give to the products, and (b) the time of reviewing the products.*

We show that each component of this hypothesis is statistically significant (see Sec. 3.5). We incorporate these three factors into a novel **attributed product-product graph** construction $G(V, E)$ such that each $v_{ij} \in V$ indicates a product-rating pair (p_i, r_j) and its attribute a_{ij}^v

consists of the set of reviewers who rated p_i with r_j. An edge $e_{(ij,mn)} = \langle u_{ij}, v_{mn} \rangle \in E$ indicates the co-reviewing and co-rating patterns of two products p_i and p_m with rating r_j and r_n, respectively. The edge attribute $a^e_{(ij,mn)}$ indicates the set of co-reviewers $R_{(ij,mn)}$ who reviewed both p_i and p_m and gave same ratings r_j and r_n respectively at the same time τ_t (defined in Sec. 3.3). Note that edge $e_{(ij,in)}$ connecting same product with different ratings wont exist in G as we assume that a reviewer is not allowed to give multiple reviews/ratings to a single product.

We then propose `ExtractGroups` (pseudecode in Algo 1, a toy example in 3.2.2), a novel group detection algorithm that takes G as an input and executes a series of operations through `GroupDetector`() (Line 12) – isolated nodes are first removed (Lines 13-14), edge attributes are then merged and removed if Jaccard similarity (JS) of product sets that the corresponding reviewers reviewed is greater than a threshold (set as 0.5). Any group of reviewers eliminated due to the consideration of only common reviewers during the creation of edges is also checked through JS in order to avoid loosing any potential candidate groups (Lines 15-21). Before proceeding to the next iteration, connected components containing more than two nodes are also removed (Lines 25-26). We define $CollectiveScore$ as the average of six group level indicators defined in Sec. 3.1, and consider those as potential groups whose $CollectiveScore$ exceeds τ_{spam} (Lines 27-29, defined in Sec. 3.5.2).

It then converts the remaining structure of G into an attributed line graph (edges converted into vertices and vice versa) $G'(V', E')$ as follows: $v'_{(ij,mn)} \in V'$ corresponds to $e_{(ij,mn)}$ in G and $a^{v'}_{(ij,mn)} = a^e_{(ij,mn)}$; an edge $e'_{(ij,mn,ab)} = \langle v'_{(ij,mn)}, v'_{(ij,ab)} \rangle$ represents co-reviewing and co-rating patterns of products p_i, p_m and p_a; the corresponding edge attribute is $a_{(ij,mn,ab)} = a^e_{(ij,mn)} \cap a^e_{(ij,ab)}$. G' is again fed into `GroupDetector` in the next iteration. Essentially, in each iteration, we keep clubbing reviewers together, who exhibit coherent reviewing patterns. The iteration continues until none of the edges remain in the resultant graph, and a set of candidate fraud groups are returned.

Theorem 3.2.1 (Theorem of convergence) `ExtractGroups` *will converge within a finite number of iterations.*

Proof Sketch: **Case 1: Isolated nodes.** The algorithm removes isolated nodes at each iteration.
Case 2: Connected components with size greater than 2. The whole component will be removed from the graph.
Case 3: Connected components with size less than or equal to 2. In the next iteration, it will get converted to either of the following three structures:
(i) An isolated node (if the edge does not have a common reviewer with any other edge in the graph). Then Case 1 will follow.
(ii) A connected component with two nodes (if the edge has a common reviewer with only one adjacent edge). Then Case 2 will follow.
(iii) A connected component with more than two nodes (if the edge has a common reviewer with more than one adjacent edges). Then Case 3 will follow.

Lines 15 - 21 do not change the graph and will therefore not come under any of cases stated above. The procedure will eventually lead to a network with no edge.

3.2.1 Time Complexity of `ExtractGroups`

The worst case **time complexity** of `ExtractGroups` is $\mathcal{O}(k|E|^2)$, where k is the number of iterations, and $|E|$ is the number of edges in G.

Let E be the edge set of the attributed product-product network. For each iteration after conversion to line graph, there can be at most E isolated nodes or at most $\frac{|E|}{2}$ connected components with 2 nodes leaving the rest as connected components with more than 2 nodes, which will be removed in the current iteration and will not be proceed further. Therefore at most E edges will be proceed to the next iteration making complexity of each iteration as less than or equal to $|E|^2$. Lines 15 - 21 will also have worst case as $|E|^2$ (when all edge attributes are distinct) complexity. Therefore, the total time complexity is $\mathcal{O}(k|E|^2)$, where k is the number of iterations.

3.2.2 Demonstration of `ExtractGroups`

We provide a schematic diagram of the Algorithm 1 in Figure 2. Left panel is a bipartite graph view of the overall system with edges linking reviewers with their reviewed products. We create attributed product-product graph $G(V, E)$ such that each $v_{ij} \in V$ indicates a product-rating pair (p_i, r_j) and its attribute a^v_{ij} consists of the set of reviewers who rated p_i with r_j. An edge $e_{(ij,mn)} = \langle u_{ij}, v_{mn} \rangle \in E$ indicates the co-reviewing and co-rating patterns of two products p_i and p_m with rating r_j and r_n, respectively. The edge attribute $a^e_{(ij,mn)}$ indicates the set of co-reviewers $R_{(ij,mn)}$ who reviewed both p_i and p_m and gave same ratings r_j and r_n respectively at the same time τ_t. Note that edge $e_{(ij,in)}$ connecting same product with different ratings wont exist in G as we assume that a reviewer is not allowed to give multiple reviews/ratings to a single product. Here P_1 has been attributed by $\{R_1, R_2, R_3\}$ as they have given it rating 1 within time τ_t . In a similar fashion we associate a list of users for each node and connect 2 nodes if they have common reviewers between them and store those reviewers as edge attributes. (Step1).

`ExtractGroups` takes this graph as input and executes a series of operations via `GroupDetector`. Isolated nodes are first removed (Lines 13-14), but currently there are none. The edges with Jaccard similarity (JS) of product sets that the corresponding reviewers reviewed greater than a threshold (set as 0.5) are merged. For eg. here R_3 is subset of $\{R_1, R_2, R_3\}$. If JS of product sets of their union, i.e., $\{R_1, R_2, R_3\}$ is greater than 0.5 we will consider $\{R_1, R_2, R_3\}$ as potential group subject to merging with others as the algorithm progresses (Line 15-18). If the union's JS is less than the threshold we report $\{R_1, R_2\}$ who have co reviewed P_1 and P_3 (subject to JS condition) as candidate groups (Line 20-21). All connected components with more than 2 nodes are removed. Here component corresponding to R_3 will be removed. Similarly either $\{R_6, R_7, R_8\}$ will be considered or R_6 if they satisfy the conditions else we move to the next iteration.

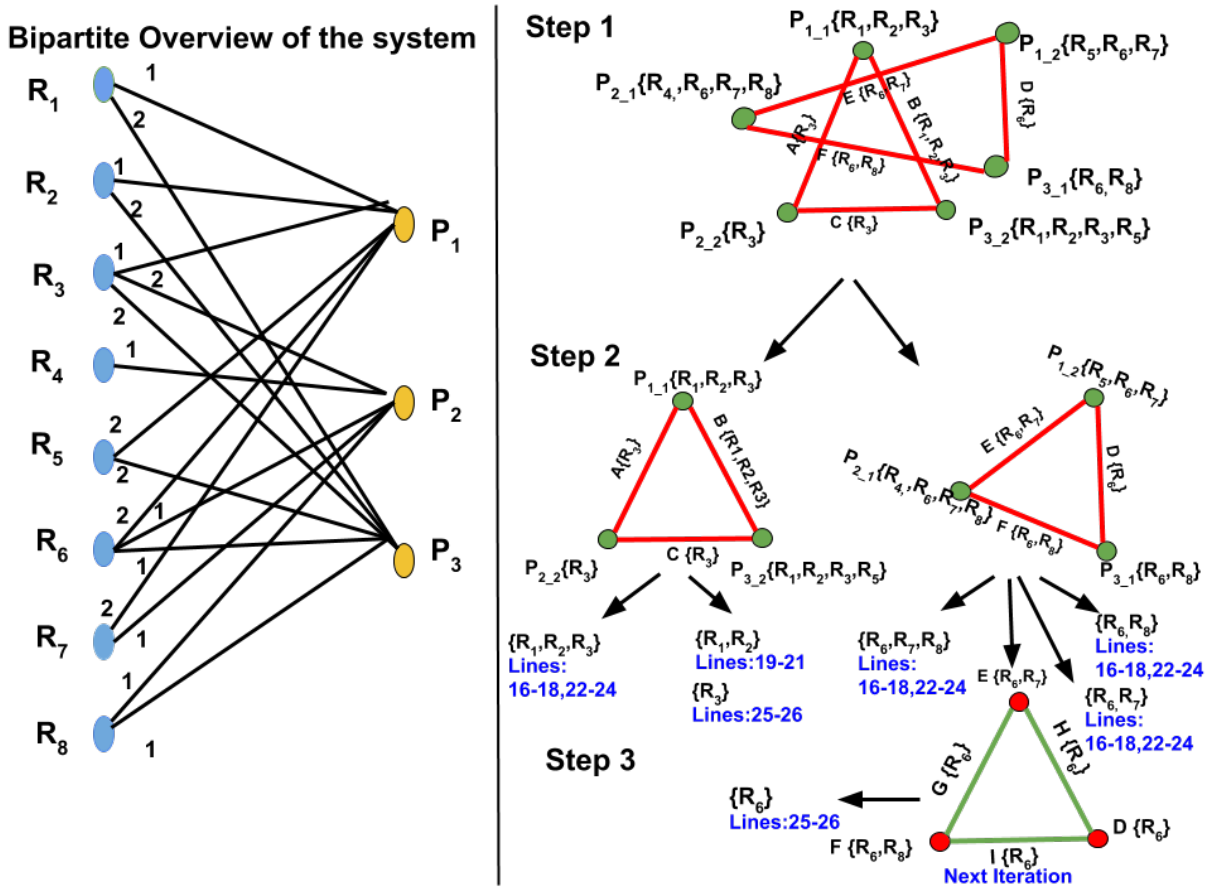


Figure 3.1: Demonstration of `ExtractGroups`.

The remaining structure of G is converted into an attributed line graph (edges converted into vertex and vice versa) $G'(V', E')$ as follows: $v'_{(ij,mn)} \in V'$ corresponds to $e_{(ij,mn)}$ in G and $a^{v'}_{(ij,mn)} = a^{e}_{(ij,mn)}$; an edge $e'_{(ij,mn,ab)} = \langle v'_{(ij,mn)}, v'_{(ij,ab)} \rangle$ represents co-reviewing and co-rating patterns of products p_i, p_m and p_a; the corresponding edge attribute is $a_{(ij,mn,ab)} = a^{e}_{(ij,mn)} \cap a^{e}_{(ij,ab)}$ (Figure). G' is again fed into `GroupDetector` in the next iteration. Again no isolated nodes are there. Edges D, E and F all have R_6 in common, so they will form a connected component with more than 2 nodes and will be reported as a candidate group, It will be removed from the graph thus removing all the edges (termination condition, Line 10).

We define *CollectiveScore* as the average of six group level indicators defined and consider those as potential groups whose *CollectiveScore* exceeds τ_{spam}. Let us assume that $\{R_1, R_2\}$, R_3 and R_6 are the groups that satisfied the given conditions. So the will be reported as the potential candidate groups on termination.

3.3 Ranking of Candidate Fraud Groups

Once candidate fraud groups are detected, DeFrauder ranks these groups based on their spamicity. It involves two steps – mapping reviewers into an embedding space based on their co-

reviewing patterns, and ranking groups based on how close the constituent reviewers are in the embedding space.

3.3.1 `Reviewer2Vec`: Embedding Reviewers

Our proposed embedding method, `Reviewer2Vec` is motivated by Wang *et al.* [29] (pseudecode in Algo 2). Given two reviewers i and j co-reviewing a product $p \in P$ by writing reviews c_p^i and c_p^j with the rating r_p^i and r_p^j at time t_p^i and t_p^j respectively, we define the *collusive spamicity* of i, j w.r.t. p as:

$$Coll(i,j,p) = \begin{cases} 0, & |t_i^p - t_j^p| > \tau_t \vee |r_i^p - r_j^p| \geq \tau_r \\ \zeta(i,j,p), & \text{otherwise} \end{cases} \tag{3.7}$$

where,

$$\zeta(i,j,p) = s^p \Bigg[\alpha \left(1 - \frac{|t_p^i - t_p^j|}{\tau_t}\right) + \beta \left(1 - \frac{|r_p^i - r_p^j|}{\tau_r}\right) + \gamma . Cosine(c_p^i, c_p^j) \Bigg]$$

Where $s^p = \frac{2}{1+e^{-(\max\limits_{q \in P} Rev(q) - Rev(p))^\theta + 2^\theta}} - 1$. Coefficients α, β and γ control the importance of time, rating and the similarity of review content, respectively ($0 \leq \alpha, \beta, \gamma \leq 1$ and $\alpha + \beta + \gamma = 1$). We set $\gamma > \alpha, \beta$ as same review content signifies more collusion as compared to the coherence in ratings and time [29]. s^p is the degree of suspicion of product p. τ_t, τ_r and θ are the parameters of the model. If the posting time difference among reviewers i and j on p is beyond τ_t or their rating on p deviates beyond τ_r (where $\tau_r = (\max - \min)x\%$, where max and min are the maximum and minimum ratings that a product can achieve respectively), we do not consider this co-reviewing pattern. `ExtractGroups` achieves best results with $\tau_t = 20$ and $\tau_r = (\max - \min)20\%$ (see Sec. 3.5.3). $\zeta(i,j,p)$ is the collusiveness between two reviewers w.r.t. product p they co-reviewed; however there might be other reviewers who reviewed p as well. Lesser the number of reviewers of p, more is the probability that i and j colluded. This factor is handled by s^p after passing it through a sigmoid function. θ is a normalizing factor which ranges between $[0, 1]$ (set as 0.4 [29]). We take the cosine similarly of two reviews c_p^i and c_p^j after mapping them into embedding space using Word2Vec [16].

Combining the collusive spamicity of a pair of reviewers across all the products they co-reviewed, we obtain the overall collusive spamicity between two reviewers:

$$\Phi(i,j) = \frac{2}{1+e^{-\sigma(i,j)}} - 1$$

$$\text{where } \sigma(i,j) = \left[\sum_{p \in \{P_i \cap P_j\}} Coll(i,j,p) \right] \frac{|P_i \cap P_j|}{|P_i \cup P_j|}$$

We then create a reviewer-reviewer spammer graph [29] which is a bi-connected and weighted graph $G_s = (V_s, E_s)$, where V_s corresponds to the set of reviewers R, and two reviewers i and

j are connected by an edge $e^s_{ij} \in E_s$ with the weight $W^s_{ij} = \Phi(i,j)$. Once G_s is created, we use state-of-the-art node embedding method to generate node (reviewer) embeddings (see Sec. 3.5.3).

Algorithm 2 `Reviewer2Vec`

Output:
    User Embeddings
Initialize:
    $w(i,j) \leftarrow 0$ for any i,j ϵ R
    $G_s \leftarrow$ empty graph
for each reviewer i in R: **do**
    for each reviewer j in R: **do**
        if $i! = j$ and i and j have reviewed product p: **then**
            $w(i,j) = w(i,j) + Coll(i,j,p)$
for each i,j do **do**
    $\sigma(i,j) \leftarrow w(i,j)\frac{|Pi \cap Pj|}{|Pi \cup Pj|}$
    $\Phi(i,j) \leftarrow \frac{2}{1+e^{\sigma(i,j)}} - 1$
    $G_s.addedge(i,j,\Phi(i,j))$
return user embeddings of graph(G_s)

3.3.2 Ranking Groups

For each detected group, we calculate the density of the group based on (Euclidean) distance of each reviewer in the group with the group's centroid in the embedding space (pseudecode in Algo 3). An average of all distances is taken as a measure of spamicity of the group. Let $\vec{i}$ be the vector representation of reviewer i in the embedding space. The group spam score $Spam(g)$ of group g is measured as:

$$Spam(g) = \frac{1}{|R(g)|} \sum_{i \in R(g)} \left[\vec{i} - \left[\frac{1}{|R(g)|} \sum_{i \in R(g)} \vec{i} \right] \right]^2 \tag{3.8}$$

Algorithm 3 GroupRanking

Output:
    Ranked Groups
Initialize:
    $G \leftarrow$ `ExtractGroups`()
    $UE \leftarrow$ `Reviewer2Vec`()
for each group g in G: **do**
    $Spam(g) = \frac{1}{|R(g)|} \sum_{i \in R(g)} \left[UE(i) - \left[\frac{1}{|R(g)|} \sum_{i \in R(g)} UE(i) \right] \right]^2$
return groups sorted on $Spam$ in increasing order

Table 3.1: Statistics of four datasets.

Dataset	# Reviews	# Reviewers	# Products
YelpNYC	359052	160225	923
YelpZIP	608598	260227	5044
Amazon	10261	1429	900
Playstore	332223	321437	193

Method	YelpNYC				YelpZIP				Amazon				Playstore			
	$\|G\|$	GS	RCS	ND	$\|G\|$	GS	RCS	ND	$\|G\|$	GS	RCS	ND	$\|G\|$	GS	RCS	ND
GGSpam	1218	0.574	0.218	0.567	1167	0.629	0.219	0.563	144	0.131	0.250	0.230	1213	0.749	0.010	0.464
GSBP	809	0.562	0.173	0.521	807	0.478	0.265	0.520	115	0.416	0.260	0.689	250	0.744	0.016	0.474
GSRank	998	0.102	0.313	0.569	1223	0.132	0.054	**0.706**	2922	0.293	0.309	0.144	994	0.577	0.018	0.476
$\text{DeFrauder}_\text{R}$	4399	0.124	0.021	—	6815	0.139	0.031	—	197	0.234	0.290	—	385	0.372	0.005	—
$\text{DeFrauder}_\text{T}$	152	0.237	0.069	—	3666	0.648	0.271	—	807	0.698	0.207	—	200	0.458	0.007	—
DeFrauder	1118	**0.731**	**0.348**	**0.603**	4574	**0.667**	**0.287**	0.602	713	**0.718**	**0.314**	**0.768**	940	**0.841**	**0.018**	**0.789**

Table 3.2: Performance of the competing methods: GGSpam [29], GSBP [27], GSRank [17], $\text{DeFrauder}_\text{R}$, $\text{DeFrauder}_\text{T}$, and DeFrauder. Number of groups detected ($|G|$) are mentioned after removing groups of size less then 3 (cyan regions). Accuracy for the group detection (white regions) and ranking (gray regions) is reported in terms of EMD (the higher, the better), and NDCG@50 (ND) respectively. $\text{DeFrauder}_\text{R}$ and $\text{DeFrauder}_\text{T}$ are used only for group detection. The ranking methods of all baselines are run on the groups detection by DeFrauder.

3.4 Datasets

We collected four real-world datasets – **YelpNYC**: hotel/restaurant reviews of New York city [21]; **YelpZip**: aggregation of reviews on restaurants/hotels from a number of areas with continuous zip codes starting from New York city [21]; **Amazon**: reviews on musical instruments [11], and **Playstore**: reviews of different applications available on Google Playstore. Fake reviews and spammers are already marked in both the Yelp datasets [21]. For the remaining datasets, we employed three human experts[1] to label spammers based on the instructions mentioned in [3, 12, 23]. They were also given full liberty to apply their own experience. Among their annotations, we took annotations for Amazon and Playstore for which at least two annotators agreed on their labels. The inter-rater agreement was 0.81 and 0.79 (Fleiss multi-rater kappa) for Amazon and Playstore, respectively. Table 3.1 shows the statistics of the datasets.

3.5 Experimental Results

Statistical Validation: To measure the statistical significance of each component (say, products reviewed by the group members) of Hypothesis 1 (Sec. 3.2), we randomly generate pairs of reviewers (irrespective of the groups they belong to, and form SET 1) and measure how their co-reviewing patterns (cumulative distribution) are different from the case if a pair of reviewers co-occur together in the same group (SET 2). We consider 10,000 such pairs for each group. We hypothesize that both these patterns are different, whereas null hypothesis rejects our hypothesis. We consider three variations for statistical analysis:

[1]They were social media experts, and their age ranged between 25-35.

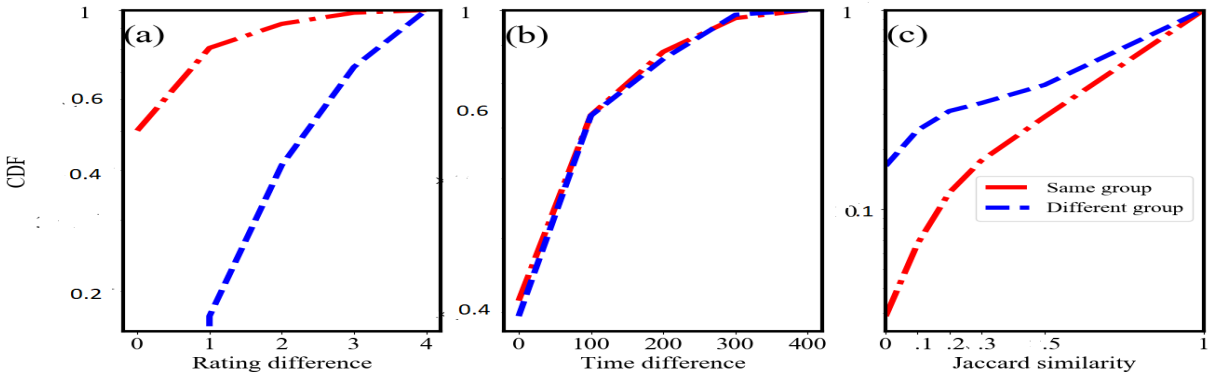


Figure 3.2: CDF of (a) rating deviation (P value<0.02),(b) Time deviation (P value=0.005) and (c) Jaccard similarity of product sets (P value=0.009) of 10,000 random pairs from both SET 1 (blue lines) and SET 2 (red lines) for YelpNYC.

1. Rating deviation: Keeping the product same, we select pairs and plot the CDF curves for rating differences between both the groups (Figure 3.2(a)).

2. Temporal deviation: Keeping the product same we select pairs and plot the CDF curves for time deviations both the groups (Figure 3.2(b))

3. Jaccard similarity: Distribution of the Jaccard similarity (JS) of the product sets of pairs separately (Figure 3.2(c))

Rating and temporal deviation of reviewers belonging to the same group (SET 2) are lesser as compared to those in different groups (SET 1) while JS is higher in the former which can be validated through the plots. We observe that the difference is statistically significant for time deviations ($p < 0.01$) which signifies that temporal coherence is more important than rating coherence in detecting potential groups. JS within group is higher as more percentage of groups are having high values.

We then perform evaluation in two stages – quality of the detected groups, and quality of the ranking algorithms.

3.5.1 Baseline Methods

We consider three existing methods (see Sec. 2 for their details) as baselines for both the evaluations – (i) **GSBP** [27], utilizes the behavioral patterns between two reviewers based on the topological structure of the reviewer graph [27]; (ii) **GGSpam** [29], models the collusive behavior between two reviewers by considering both the review time interval and the rating score deviation [29]; and (iii) **GSRank** [17]. uses frequent pattern mining to detect groups and relation models based on the relationships among groups to identify spam groups [17].

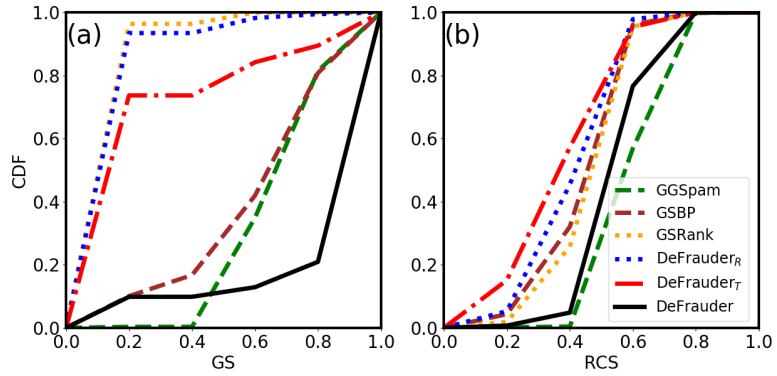
3.5.2 Evaluating Candidate Groups

Along with the three baselines mentioned above, we also consider two variations of DeFrauder as baselines for group detection: **DeFrauder$_{\mathbf{R}}$** constructs the attributed product-product graph G based only on co-rating without considering the time of reviewing; and **DeFrauder$_{\mathbf{T}}$** constructs G based only on same co-reviewing time without considering co-rating. This also helps in justifying why both time and rating are important in constructing G for group detection.

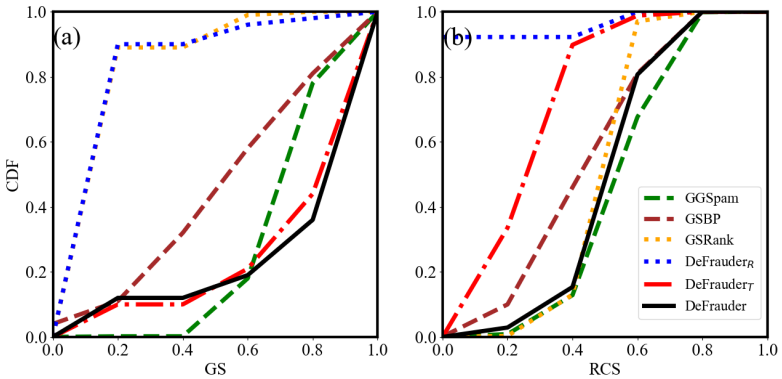
Mukherjee *et al.* [17] suggested to use cumulative distribution (CDF) of *group size* (GS) and *review content similarity* (RCS) to evaluate the quality of the spam groups. Here we discard groups with size less than 2. **Group Size (GS)** favors large fraud groups as large groups are more damaging than smaller ones: $GS(g) = \frac{1}{1+e^{-(|R(g)|-2)}}$. **Review Content Similarity (RCS)** captures inter-reviewer review content similarity (as spammers copy reviews of each other): $RCS(g) = \max_{p \in P(g)} \{\frac{1}{|R(g)|^2} \sum_{(i,j) \in R(g) \times R(g)} cosine(c_p^i, c_p^j)\}$. The larger the deviation of each distribution from the vertical axis (measured in terms of Earth Mover's Distance (EMD)), the better the quality of the detected method [17].

Comparison: We choose the following parameter values as default based on our parameter selection strategy (Fig. 3.5): τ_t: 20 days, $\tau_{spam} = 0.4$. The number of groups we obtain from different datasets is reported in Table 3.2. Fig. 3.3 shows the CDF of GS and RCS for all the competing methods on YelpNYC[2], which is further summarized quantitatively (in terms of EMD) in Table 3.2. DeFrauder outperforms the best baseline by 11.35% and 4.67% higher relative EMD (averaged over four datasets) for GS and RCS, respectively. We also notice that DeFrauder$_{\mathrm{T}}$ performs better than DeFrauder$_{\mathrm{R}}$, indicating that temporal coherence is more important than rating coherence in detecting potential groups.

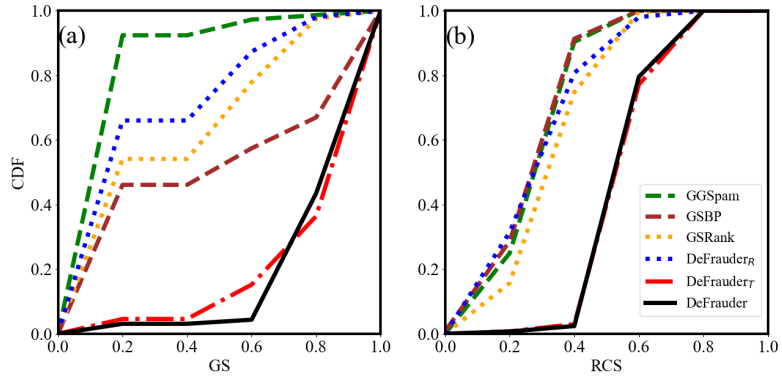
[2]Results are similar on the other datasets, and thus omitted.



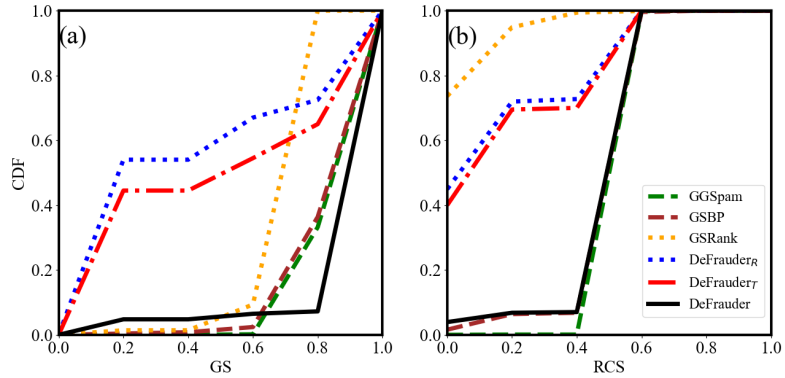
(a) YelpNYC



(b) YelpZip



(c) Amazon



(d) Playstore

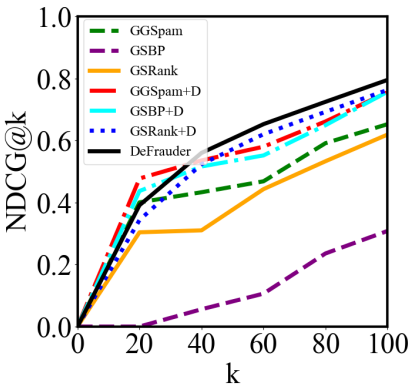
Figure 3.3: CDF of (a) GS and (b) RCS for four datasets. The more the distance of a CDF (corresponding to a method) from y-axis, the better the performance of the method.

3.5.3 Evaluating Ranking Algorithm

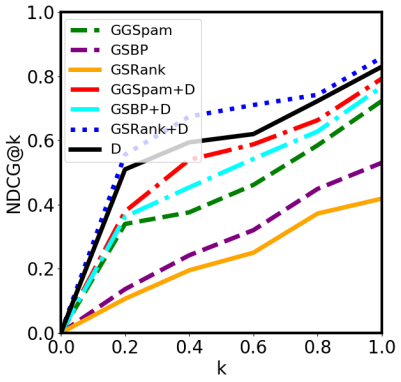
We use **NDCG@k**, a standard graded evaluation metric. Since each reviewer was labeled as fraud/genuine, we consider the graded relevance value (ground-truth relevance score used in Normalized Discounted Cumulative Gain (NDCG) [15]) for each group as the fraction of reviewers in a group, who are marked as fraud. The candidate groups are then ranked by each competing method, and top k groups are judged based on NDCG.

Comparison: We choose the following parameter values as default based on our parameter selection strategy (Fig. 3.5): $\alpha = \beta = 0.3$, $\gamma = 0.4$, $\tau_t = 20$ days, $\tau_r = (\max - \min)20\%$ $\tau_{spam} = 0.4$. We use Node2Vec [9] for embedding[3]. Fig. 3.4 shows the performance of the competing methods for different values of k (top k groups returned by each method). Since DeFrauder produces better groups (Sec. 3.5.2), we also check how the ranking method of each baseline performs on the groups detected of DeFrauder. Fig. 3.4 shows that with DeFrauder's group structure, GGSpam and GSBP show better performance than DeFrauder till $k = 40$; after that DeFrauder dominates others. However, all the baselines perform poor with their own detected groups. This result also indicates the efficiency of our group detection method. Table 3.2 reports that DeFrauder beats other methods across all the datasets except YelpZIP on which GSRank performs better with DeFrauder's detected groups. Interestingly, no single baseline turns out to be the best baseline across datasets. Nevertheless, DeFrauder outperforms the best baseline (varies across datasets) by 17.11% higher relative NDCG@50 (averaged over all the datasets).

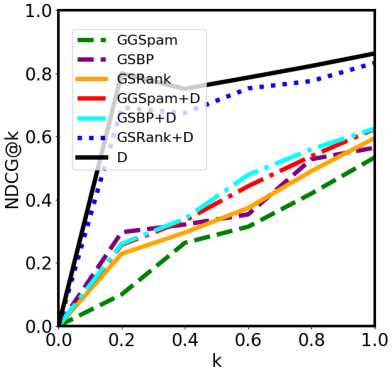
[3]We tried with DeepWalk [20] and LINE [24] and obtained worse results compared to Node2Vec.



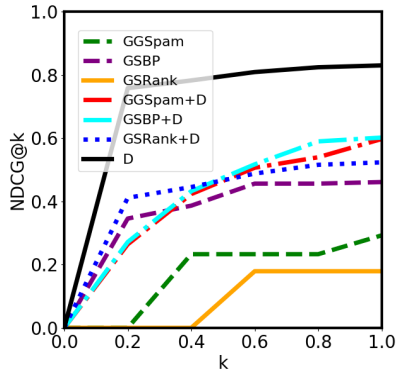
(a) YelpNYC



(b) YelpZip



(c) Amazon



(d) Playstore

Figure 3.4: Performance on all datasets. Baselines are run with their detected groups as well as with the groups (+D) detected by DeFrauder (same naming convention as in Table 3.2).

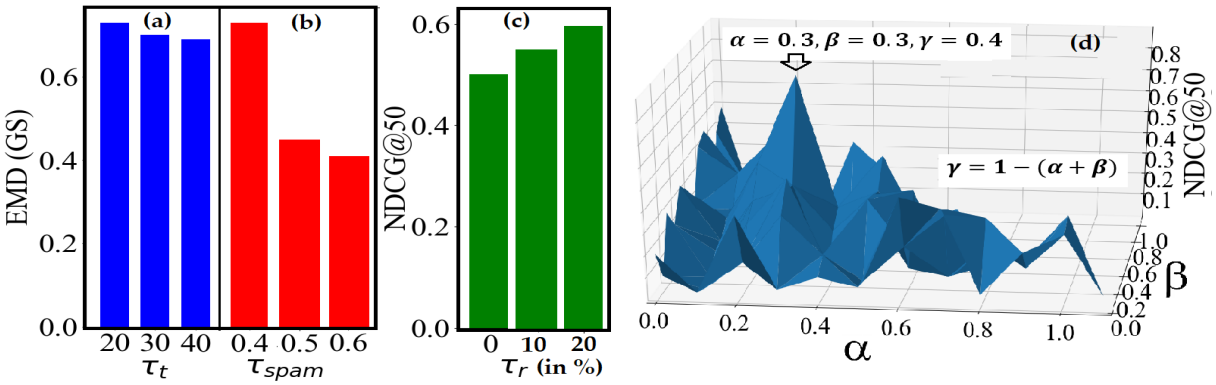


Figure 3.5: Parameter selection on YelpNYC (Results are similar on the other datasets, and thus omitted.). We vary each parameter keeping others as default. Since $\tau_t = 20$ produces best results for group detection, we kept this value for ranking as well.

3.6 Conclusion

In this paper, we studied the problem of fraud reviewer group detection in customer reviews. We established the principle of coherence among fraud reviewers in a group in terms of their co-reviewing patterns. This paper contributed in four directions: **Datasets:** We collected and annotated two new datasets which would be useful to the cybersecurity community; **Characterization:** We explored several group-level behavioral traits to model inter-personal collusive dynamics in a group; **Method:** We proposed DeFrauder, a novel method to detect and rank fraud reviewer groups; **Evaluation:** Exhaustive experiments were performed on four datasets to show the superiority of DeFrauder compared to five baselines.

Bibliography

[1] Leman Akoglu, Rishi Chandy, and Christos Faloutsos. Opinion fraud detection in online reviews by network effects. In *ICWSM*, pages 985–994, 2013.

[2] Mohammad Allahbakhsh, Aleksandar Ignjatovic, Boualem Benatallah, Elisa Bertino, Norman Foo, et al. Collusion detection in online rating systems. In *Asia-Pacific Web Conference*, pages 196–207. Springer, 2013.

[3] Bing Liu Junhui Wang Meichun Hsu Malu Castellanos and Riddhiman Ghosh Arjun Mukherjee, Abhinav Kumar. Spotting opinion spammers using behavioral footprints. In *SIGKDD*, Chicago,USA, August 2013.

[4] Udit Arora, William Scott Paka, and Tanmoy Chakraborty. Multitask learning for blackmarket tweet detection. *arXiv preprint arXiv:1907.04072*, 2019.

[5] Aditya Chetan, Brihi Joshi, Hridoy Sankar Dutta, and Tanmoy Chakraborty. Corerank: Ranking to detect users involved in blackmarket-based collusive retweeting activities. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 330–338. ACM, 2019.

[6] Sarthika Dhawan, Siva Charan Reddy Gangireddy, Shiv Kumar, and Tanmoy Chakraborty. Spotting collusive behaviour of online fraud groups in customer reviews. *arXiv preprint arXiv:1905.13649*, 2019.

[7] Hridoy Sankar Dutta, Aditya Chetan, Brihi Joshi, and Tanmoy Chakraborty. Retweet us, we will retweet you: Spotting collusive retweeters involved in blackmarket services. In *IEEE/ACM 2018 International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2018, Barcelona, Spain, August 28-31, 2018*, pages 242–249, 2018.

[8] Geli Fei, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. Exploiting burstiness in reviews for review spammer detection. In *ICWSM*, 2013.

[9] Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. In *SIGKDD*, pages 855–864, 2016.

[10] Sonu Gupta, Ponnurangam Kumaraguru, and Tanmoy Chakraborty. Malreg: Detecting and analyzing malicious retweeter groups. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, pages 61–69. ACM, 2019.

[11] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*, pages 507–517.

[12] Catey Hill. 10 secrets to uncovering which online reviews are fake. https://www.marketwatch.com/story/10-secrets-to-uncovering-which-online-reviews-are-fake-2018-09-21, 2018.

[13] Yufeng Kou, Chang-Tien Lu, Sirirat Sirwongwattana, and Yo-Ping Huang. Survey of fraud detection techniques. In *ICNSC*, pages 749–754, 2004.

[14] Ee-Peng Lim, Viet-An Nguyen, Nitin Jindal, Bing Liu, and Hady Wirawan Lauw. Detecting product review spammers using rating behaviors. In *CIKM*, pages 939–948. ACM, 2010.

[15] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103, 2010.

[16] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[17] Arjun Mukherjee, Bing Liu, and Natalie Glance. Spotting fake reviewer groups in consumer reviews. In *WWW*, pages 191–200, New York, April 2012.

[18] Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie Glance. What yelp fake review filter might be doing? In *ICWSM*, pages 1–12, 2013.

[19] Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T Hancock. Finding deceptive opinion spam by any stretch of the imagination. In *ACL-HLT*, pages 309–319, 2011.

[20] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *SIGKDD*, pages 701–710, 2014.

[21] Shebuti Rayana and Leman Akoglu. Collective opinion spam detection: Bridging review networks and metadata. In *SIGKDD*, pages 985–994, Sydney, NSW, Australia, August 2015.

[22] DeFrauder: Anonymized repository: code & data & Supplementary. https://tinyurl.com/defrauder19, 2019.

[23] Somayeh Shojaee, Azreen Azman, Masrah Murad, Nurfadhlina Sharef, and Nasir Sulaiman. A framework for fake review annotation. In *UKSIM-AMSS*, pages 153–158, 2015.

[24] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *WWW*, pages 1067–1077, 2015.

[25] Guan Wang, Sihong Xie, Bing Liu, and S Yu Philip. Review graph based online store review spammer detection. In *ICDM*, pages 1242–1247. IEEE, 2011.

[26] Guan Wang, Sihong Xie, Bing Liu, and Philip S Yu. Identify online store review spammers via social review graph. *ACM TIST*, 3(4):61:1–61:21, 2012.

[27] Zhuo Wang, Tingting Hou, Dawei Song, Zhun Li, and Tianqi Kong. Detecting review spammer groups via bipartite graph projection. *The Computer Journal*, 59(6):861–874, 2016.

[28] Zhuo Wang, Songmin Gu, and Xiaowei Xu. Gslda: Lda-based group spamming detection in product reviews. *Applied Intelligence*, 48(9):3094–3107, 2018.

[29] Zhuo Wang, Songmin Gu, Xiangnan Zhao, and Xiaowei Xu. Graph-based review spammer group detection. *KIAS*, 55(3):571–597, Jun 2018.

[30] Chang Xu and Jie Zhang. Towards collusive fraud detection in online reviews. In *ICDM*, pages 1051–1056, 2015.

[31] Chang Xu, Jie Zhang, Kuiyu Chang, and Chong Long. Uncovering collusive spammers in chinese review websites. In *CIKM*, pages 979–988, 2013.