Towards characterization and detection of collusive reputation fraud in crowdsourced opinions

Xu, Chang

2017

Xu, C. (2017). Towards characterization and detection of collusive reputation fraud in crowdsourced opinions. Doctoral thesis, Nanyang Technological University, Singapore.

<http://hdl.handle.net/10356/69708>

<https://doi.org/10.32657/10356/69708>

TOWARDS CHARACTERIZATION AND DETECTION OF COLLUSIVE REPUTATION FRAUD IN CROWDSOURCED OPINIONS

A thesis submitted to
School of Computer Science and Engineering
Nanyang Technological University

by

Chang Xu

in fulfilment of the requirements
for the degree of Doctor of Philosophy

2016

Abstract

Crowdsourced opinion portals allow people to share experiences and assessments about products or services by posting reviews online, and have become an informative source for consumers to make purchase decisions. This, however, endows the generated crowdsourced opinions with the ability to affect the profitability of involved businesses, with positive reviews building up a business’s reputation to bring increase in sales and negative reviews damaging the public image to cause business failure. Unfortunately, such monetary incentive has spurred the rise of an underground market, where spammers are hired to manipulate (e.g., promote or degrade) a business’s reputation by crafting fake, misleading reviews. Furthermore, this blackhat practice is increasing in sophistication by launching coordinated campaigns to produce reviews in collusion, which is considered much more harmful and difficult to detect due to the ability to dominate the overall sentiments quickly while dilute suspicious footprints through workload distribution. To address the collusive reputation fraud problem, the first objective of this thesis is to understand the behaviors of collusive spammers or we call colluders, through characterizing their inner structures in spam campaigns. Moreover, to eventually diminish the adversarial impact, the second objective is to design effective and efficient algorithms for collusion detection, such that the generated fake opinions can be accurately and timely removed from the system.

Firstly, a novel collusion pattern is disclosed where multiple tiny spammer groups cooperate with each other on partially overlapping targets. To handle this new attack vector, two graph-based classification algorithms are proposed to spot colluders in supervised settings. These algorithms are designed to capture colluders’ correlations by defining neighborhoods for users, such that the detection can benefit from the information collected from the neighborhood. Experimental results have shown that both algorithms outperform existing detectors that do not utilize relations between colluders.

Secondly, to further study the underlying colluder correlations, a suite of pairwise collusive behavior measures are proposed and leveraged for detection. Compared to existing pointwise measures, pairwise measures are more fine-grained to directly reflect the relations between colluders. Moreover, a novel random walk-based detection framework FRAUDINFORMER is proposed to cooperate with the pairwise measures, which can work in unsupervised settings where no prior knowledge of collusion instances is needed. Extensive experiments have been conducted to evaluate the effectiveness of the proposed pairwise measures and detection framework.

Thirdly, to remedy the deficiency of existing unsupervised detectors in collusion prediction, a novel statistical framework is proposed to perform both collusion inference and prediction together. The key perspective is to take a hybrid generative and discriminative probabilistic approach where both generative and discriminative processes can benefit from each other when performing collusion inference and prediction in unsupervised settings. In addition, a suite of homogeneity-based collusive behavior measures is proposed to capture the connections between colluders by measuring their intra-group similarities. Experiments on two real-world datasets have demonstrated the effectiveness of the proposed method and its improvements in learning and predictive abilities.

Finally, an incremental detection framework FRAUDSCAN is proposed to counter collusive reputation fraud on temporal dimension, so as to quickly respond to newly-emerging campaign activities. The task of online reputation fraud campaign detection is formally defined. Then, a unified and scalable optimization framework is proposed which can adapt built detection model to emerging fraud patterns over time. Empirical analysis on real data has shown such a temporal-oriented detection paradigm can significantly increase the accuracy of fraud campaign detection.

In summary, the proposed collusive behavior measures in this thesis aim to characterize the behaviors of colluders in reputation fraud campaigns, providing insights towards the development of algorithms for filtering collusive reputation fraud in crowdsourced opinions. The proposed detection frameworks can perform both collusion prediction and inference in supervised and unsupervised settings, and are designed to work in batch and incremental fashion for different detection scenarios.

Acknowledgments

First of all, I would like to express my sincere gratitude towards my supervisor, **Dr. Zhang Jie**. Under his elaborative supervision, I have not only learned the cutting-edge techniques on artificial intelligent and trust management, but also acquired a conscientious research attitude through critical thinking. Thank you for all your efforts in time, thoughts, and fundings to make my Ph.D. life fruitful and exhilarating. Your advice on both research and my career have been invaluable.

Many thanks to all my coauthors: Dr Carol Fung, Dr. Chong Long, Dr. Kuiyu Chang, and Miss Huanhuan Zhang. I appreciate all their efforts on the research work and I enjoy the experience of the collaborations with them all the time. I would also like to appreciate my colleagues in Computational Intelligence Lab (CIL) for their friendly support and help. They include Dr. Siwei Jiang, Dr. Guibing Guo, Dr. Hui Fang, Dr. Yuan Liu, Dr. Athirai A. Irissappane, Mr. Zhiguang Cao, Ms. Zhu Sun, Ms. Dongxia Wang, Mr. Zehong Hu, and Mr. Guozhu Meng.

Lastly, I would like to thank my family, my mother and father, for all the sacrifices that they have made for me. Thank you for always supporting me and encouraging me to follow my dreams.

Contents

Abstract	i
Acknowledgements	iii
List of Figures	x
List of Tables	xv
1 Introduction	1
1.1 Limitations of Existing Approaches	3
1.2 The Proposed Approaches	6
1.3 Contributions	10
1.4 Organization	10
2 Literature Survey	15
2.1 Terminology	15
2.2 Reputation Fraud Characterization	18
2.2.1 Review-centric Characterization	18
2.2.2 User-centric Characterization	20
2.2.3 Campaign-centric Characterization	22
2.3 Supervised Reputation Fraud Detection	24
2.4 Unsupervised Reputation Fraud Detection	26
2.4.1 Individual Spammer Detection	27
2.4.2 Fraud Campaign Detection	31
2.5 Other Collusive Fraud Activities on the Web	33
2.6 Summary	36

3 Graph-based Classification Algorithms for Collusive Reputation Fraud

Detection	39
3.1 Data Acquisition and Preparation	40
3.1.1 Colluder Sampling	40
3.1.2 Annotation	41
3.1.3 Statistics	43
3.2 Empirical Analysis of State-of-the-art Reputation Fraud Indicators	45
3.2.1 Linguistic Indicators	46
3.2.2 Individual Behavioral Indicators	48
3.2.3 Collusive Behavioral Indicators	51
3.2.4 Indicators Comparison and Discussion	53
3.3 The KNN-based Method	56
3.4 The Graph-based Classification Method	60
3.4.1 Colluder Random Field (CoRF)	61
3.4.2 Collective Inference Algorithm	63
3.5 Evaluational Analysis	66
3.5.1 Evaluation of FRAUDKNN	66
3.5.2 Evaluation of the Graph-based Method	70
3.5.3 Performance Comparison	75
3.5.4 Generalization Ability	78
3.6 Summary	83

4 FraudInformer: An Unsupervised Framework for Collusive Reputation

Fraud Detection	85
4.1 Problem Formulation	86
4.2 The FraudInformer Framework	86
4.2.1 Mutual Disclosure Modeling	87
4.2.2 Confidence Weighting	89
4.2.3 Global Ranking	90
4.3 Collusiveness Measure: Pairwise Features	91
4.3.1 Product-based Sentiment Deviation (PSD)	91

4.3.2	Product-based Time Deviation (PTD)	92
4.3.3	Product-based Review Text Similarity (PTS)	92
4.3.4	Brand-based Sentiment Deviation (BSD)	93
4.3.5	Brand-based Time Deviation (BTD)	94
4.3.6	Brand-based Review Text Similarity (BTS)	94
4.3.7	Reviewing Activity Homophily (RAH)	94
4.3.8	Reviewing Lifetime Homophily (RLH)	95
4.3.9	Pairwise Feature Combination	95
4.4	Empirical Analysis	96
4.4.1	Dataset	96
4.4.2	Evaluation Criteria	97
4.4.3	Effects of Pairwise Features	97
4.4.4	Effects of Investigating Range	98
4.4.5	Effects of Confidence Weighting	102
4.4.6	Comparison Analysis	104
4.4.7	Generalization Evaluation	106
4.5	The Adversarial Offense	108
4.5.1	Aggressiveness Control	108
4.5.2	Camouflage at Rush-hour	110
4.6	Summary	111

5 LCM: A Latent Variable Model for Collusion Prediction and Inference in Crowdsourced Opinions 113

5.1	Collusive Behaviors within Campaigns	114
5.1.1	Target-based h-CBMs	116
5.1.2	Rating-based h-CBMs	117
5.1.3	Temporal-based h-CBMs	117
5.1.4	Activity-based h-CBMs	119
5.2	Problem Formulation	121
5.3	The Proposed Statistical Model	121
5.3.1	Two Views of Collusive Reputation Fraud Modeling	122

5.3.2	The Chaining-based Model	125
5.3.3	The Unified Model	128
5.3.4	Collusion Inference and Prediction	133
5.4	Experimental Analysis	134
5.4.1	Datasets	134
5.4.2	Effectiveness of h-CBMs	134
5.4.3	Prediction Performance	136
5.4.4	Inference Performance	140
5.4.5	Learned Collusive Behavioral Patterns	142
5.5	Summary	144
6	FraudScan: An Incremental Framework for Reputation Fraud Campaign Detection	147
6.1	Problem Formulation and Challenges	148
6.2	The Proposed Framework	149
6.2.1	Campaign Activity Monitoring	150
6.2.2	Fraud Campaign Embedding	152
6.2.3	Online Campaign Detection	153
6.3	Experimental Analysis	155
6.3.1	Datasets	156
6.3.2	Evaluation Protocol	156
6.3.3	Performance Evaluation	156
6.3.4	Earliness of Detection	159
6.3.5	Sensitivity Analysis	160
6.3.6	Efficiency Analysis	162
6.4	Conclusions	163
7	Conclusions and Future Work	165
7.1	Conclusions	165
7.2	Future Work	168
7.2.1	Extension of the Graph-based Classification Algorithms	168
7.2.2	Extension of FraudInformer	169

7.2.3	Extension of LCM	169
7.2.4	Extension of FraudScan	171
References		175

List of Figures

1.1	A real reputation fraud campaign in the wild.	2
1.2	A novel formulation of collusive attack in crowdsourced opinions. Boxes denote items and circles denote colluders. Dashed boxes denote reputation fraud campaigns comprising multiple colluders.	4
1.3	The organization of the thesis.	11
2.1	The illustration of reputation fraud campaigns, where two campaigns both with size two are shown. $P1 - 5$ are the targeted items. Colluders are colored in red, blue, and yellow, while normal users are not colored. . . .	17
3.1	Cumulative distributions of the average size of the groups a colluder (red) or non-colluder (blue) belongs to on AmazonCn.	44
3.2	Distributions of the number of products reviews by a colluder (red) or non-colluder (blue) on AmazonCn.	44
3.3	Cumulative distributions of colluders (solid/red) and non-colluders (dashed/blue) vs. indicator scores.	47
3.4	Cumulative distributions of colluders (solid) and non-colluders (dashed) vs. indicator scores.	49
3.5	Cumulative distributions of colluders (solid) and non-colluders (dashed) vs. indicator scores.	52
3.6	Areas under ROC curves (AUC) for all the 24 evaluated indicators (exclude TP) and their combinations. Larger AUC score denotes better detection trade-off between true and false positives based on the corresponding indicator. For name dis-ambiguity, GD(I) as General Deviation, GD(C) as Group Deviation.	54

3.7	4 groups (dashed boxes) of colluders (circles) collaboratively review six products (squares).	55
3.8	Distributions of the three types of pairwise similarity measures over the three user pairs, i.e., positive-positive pair (in red/dashed line), negative-positive pair (in black/dashed line), and negative-negative pair (in blue/solid line).	67
3.9	Distributions of the overall pairwise similarity values (sv) over the three user pairs, i.e., positive-positive pair (in red/dashed line), negative-positive pair (in black/dashed line), and negative-negative pair (in blue/solid line).	68
3.10	Distribution of the number of the neighbors with non-zero sv for each user.	69
3.11	Performance (F1-score and MCC) of FRAUDKNN vs. the number of nearest neighbor k . Error bars show the standard error of the mean.	69
3.12	Classification performance (F1-score and MCC) of the graph-based classification method with different time windows. Error bars show the standard error of the mean.	72
3.13	The distribution of the weights ($cr \cdot cd$) of the colluder graph edges vs. different time window settings with $\Delta t = 7, 42, 84, 119$	73
3.14	Distribution of the time intervals associated with different types of edges in the colluder graph with $\Delta t = +\infty$. The median values are displayed above the corresponding boxes.	74
3.15	Performance comparison between the proposed methods (FRAUDKNN and CoRF) and the three baseline classifiers (SRD, KNN and SVM) on AmazonCn dataset. Our proposed methods improve over the baseline classifiers at the confident level of 95% based on two-tailed t-test.	77
3.16	Cumulative distributions of the average size of the groups a colluder (red) or a non-colluder (blue) belongs to on YelpZip.	80
3.17	Distributions of the number of products reviews by a colluder (red) or a non-colluder (blue) on YelpZip.	81
3.18	Performance comparison between the proposed methods (FRAUDKNN and CoRF) and the three baseline classifiers (SRD, KNN and SVM) on YelpZip dataset. Our proposed methods improve over the baseline classifiers at the confident level of 95% based on two-tailed t-test.	82

4.1	An illustration of chronologically-ordered review lists. It shows four lists of reviewers in chronological order for products P1–P4.	87
4.2	Histograms of pairwise features over three types of pairs: colluder to colluder (C-C), non-colluder to colluder (NC-C), non-colluder to non-colluder (NC-NC). The x-axis denotes the pairwise feature scores (collusiveness) and the y-axis denotes the corresponding frequencies. χ^2 statistics is shown in the parenthesis in each sub-caption.	99
4.3	Ranking performance for different investigating range (IR) settings. Uniform kernel function is used as the confidence function.	100
4.4	The rise of TYPE I noise from the collusion weight allocation in FRAUD-INFORMER.	100
4.5	Evaluation of different confidence weighting functions.	103
4.6	Performance comparison between FRAUDINFORMER and GSRank in NDCG@ k ($k \in [100, 1000]$) on AmazonCn dataset.	105
4.7	Performance comparison between FRAUDINFORMER and GSRank in NDCG@ k ($k \in [100, 1000]$) on YelpZip dataset.	107
5.1	Graphical model representation of unified LCM. Light circles denote the latent random variables $\{z_n\}_{n=1}^N$ and model parameters $\{\mathbf{w}, \boldsymbol{\lambda}\}$, and dark circles denote the observed random variables $\{\phi_n\}_{n=1}^N$. The left plate indexed by N denotes the groups in $\mathcal{U}$, and the right plate indexed by “2” denotes the h-CBM distributions for colluder ($k = 1$) and non-colluder ($k = 0$) groups in $\mathcal{U}$	130
5.2	Classification performance (AUC) of h-CBMs on AmazonCn and YelpZip.	135
5.3	Collusion prediction performance (in Average Precision and AUC) of LCM compared with GSRank on AmazonCn and YelpZip datasets.	138
5.4	Collusion inference performance (in Average Precision and AUC) of LCM compared with GSRank on AmazonCn and YelpZip datasets.	141

5.5	The learned collusive behavioral patterns on AmazonCn and YelpZip datasets. Each behavioral pattern is modeled as a Beta density function (PDF) and corresponds to each h-CBM dimension $\phi \in \{\text{TC, RC, TS, ROS, FS, ALS, AC, WS}\}$. The density functions for colluders and non-colluders are shown in red/solid lines and blue/dotted lines respectively. The expected values for each learned behavioral distributions are plotted as horizontal lines and also reported in the plot captions.	143
6.1	User projections in the embedded space (the first two principal components are used). Samples consisting of 100 campaigned spammers (in red/circle) and 100 normal users (in blue/triangle) are shown.	157
6.2	Effectiveness comparison between FRAUDSCAN and state-of-the-art approaches on the two datasets.	158
6.3	Histograms of detection time lags for spammers caught by the compared methods. The histograms are truncated at 90-day lag due to the long-tail shape.	160
6.4	The effects of parameters α_1 , α_2 , α_3 , and k on the detection performance (AUC).	161
6.5	The cumulative distribution of detection latency.	162

List of Tables

3.1	Statistic summary of our Amazon.cn dataset.	43
3.2	Statistics of colluder graphs with different time window settings. The statistics of the colluder graphs corresponding to $\Delta t = 14, 28, 42, 56, 70, 84, 98, 112$ are not shown due to the space constraints.	71
3.3	Statistic summary of the YelpZip dataset.	79
4.1	Performance comparison between FRAUDINFORMER and the learning to rank algorithms on AmazonCn dataset. The “gain” in the parenthesis means the improved margin of our method over the baseline approaches.	105
4.2	Performance comparison between FRAUDINFORMER and the learning to rank algorithms on YelpZip dataset. The “gain” in the parenthesis means the improved margin of our method over the baseline approaches.	107

Chapter 1

Introduction

As the booming of Web 2.0 technologies over the past decade, user-generated information has become increasingly valuable to both service providers and Web users. One of the most successful examples is the prevalence of crowdsourced opinion portals, where people can share experiences and assessments about particular items such as products and services in the form of review postings. These crowdsourced opinions are devoted to break the information asymmetry between businesses and consumers, and have become an informative source for consumers to make purchase decisions [7, 33]. However, as becoming more influential, crowdsourced opinions have acquired the ability to affect the profitability of involved items [16, 12]. In particular, positive reviews could often build up an item's reputation and further bring increase in sales, while negative commentary, on the other hand, could damage its public image and lead to eventual business failure. This monetary incentive, unfortunately, has spurred the rise of an underground market, where a group of people or we call *spammers* are hired to manipulate (either promote or demote) the reputations of designated items by crafting fake, misleading reviews. It has been estimated that about 16% of Yelp reviews written for the restaurants in the metropolitan Boston area are fake [34]. The volume of possibly fictitious reviews on yelp.com rose from 5% in 2006 to 20% in 2013 [34]. Moreover, remarkably about 1/3 of consumer reviews are estimated to be suspicious, if not forged, on the Internet [49]. A

growing number of businesses have been driven by the desire to achieve positive public impression through receiving online praises, which further stimulates the shady markets for crafting phony crowdsourced reviews.

To make the situation even worse, spammers are constantly evolving in specialization so as to stay ahead of defensive infrastructure. Recently, they are found to rely on *astroturfing* [55] that can produce massive reviews through collaboration. With popular crowdsourcing services, a group of spammers or we call *colluders* can easily orchestrate reputation fraud campaigns comprising many users with different demographics [13]. Figure 1.1 shows a real reputation fraud campaign found in a crowdsourcing platform. In this example, the campaign is described as a crowdsourcing task that instructs each worker to leave a positive rating (with monetary rewards) for a targeted item during a specific time period.

Write a Product Review on Amazon.com
Employer: Sunncry | Job ID: 30785 | Category: Sale & Marketing
Job Status: Open (4 days left) **Bidding Started:** May 09, 2016 23:54
Job Reward: US\$5.00 **Bidding Ends:** May 17, 2016 23:54
Description:
1. You should have an account and have purchase history on amazon.com.
2. One account and one IP can only write one review.
3. I will send you the review and you will only need to copy and paste.
4. Send us a screen shot once you finish it!

Figure 1.1: A real reputation fraud campaign in the wild.

Compared to individual spammers, such *collusive fraud* is considered much more damaging for the following reasons. First, *domination*: it has richer manpower to manipulate, so as to quickly dominate the reputation of a target item. Second, *covertiness*: the overall workload of a campaign can be undertook by all the involved spammers, such that each of them would not appear suspiciously aggressive anymore. In this thesis, the aim is to address such collusive reputation fraud problem by designing algorithms that can effectively, efficiently, and promptly detect the most recent and damaging reputation fraud practices, so as to minimize the caused damages as much as possible.

1.1 Limitations of Existing Approaches

Various approaches have been proposed to detect (collusive) reputation fraud in crowd-sourced opinions. In general, these approaches conduct detection in three procedures. The first is to collect and annotate a set of representative, real instances of reputation fraud in the wild (e.g., fake reviews, spammers, or colluders), which are used to prepare a *golden-standard* dataset for subsequent characterization and algorithmic design. Secondly, a feature space is constructed to characterize reputation fraud practice based on linguistic analysis on the review contents and/or behavioral profiling on reviewing activities. Finally, various models or algorithms are built to detect suspicious instances based on the devised feature space.

Given the created golden-standard datasets, supervised approaches are proposed to train different kinds of classifiers based on the reputation fraud annotations [24, 30, 42, 42]. In these approaches, the design of effective feature space is crucial to the ultimate detection performance. However, in the war against anti-fraud infrastructures, colluders can learn and change their tactics accordingly, and eventually adapt to evade existing systems that equip with those features. We have found in today’s popular review portals that colluders have evolved to operate in small scale, well-coordinated attacks, which fails most of the state-of-the-art features for capturing reputation fraud. To illustrate, Figure 1.2 shows the orchestration of a group of real campaigns in collusive reputation fraud. It can be seen that eight small campaigns are organized together to target a specific set of items. It is worth noting that the item sets targeted by each group and the member sets of each group are only partially overlapped, so as to avoid the formation of more blatant clusters that can be easily detected. However, most of the existing supervised approaches only focus on individual spammer characterization and detection, and are not designed to tackle such collusive attacks. This thesis focuses on the design of new methods to characterize the newly-emerged attack vectors and to effectively detect

those stealthy colluders, which leads to the proposal of a novel graph-based model for collusive reputation fraud detection in Chapter 3.

Figure 1.2: A novel formulation of collusive attack in crowdsourced opinions. Boxes denote items and circles denote colluders. Dashed boxes denote reputation fraud campaigns comprising multiple colluders.

The main problem of supervised detection models is that they rely heavily on the annotated fake review data. Different from other annotating tasks in related problems such as sentiment analysis or relevance evaluation, the identification of fake reviews is at least as difficult as the detection of human lies [42]. There is a lack of ground truth in practice whether a review is fabricated or not [24, 31, 56]. Therefore, unsupervised solutions have recently been proposed for collusive reputation fraud detection [38, 37]. These approaches operate in two steps. First, groups of spammers that co-review multiple items together are generated for collective behavior characterization. Then, ranking-based algorithms are developed to score the suspicion levels of those spammer groups. However, such group-level characterization can be problematic to handle colluders who use camouflage to cover up their malicious behaviors. For a spammer, the camouflage can be achieved by creating authentic reviews for some random items, so as to be grouped together with normal users. Then, for large groups, the judgments on the involved normal

users can potentially be affected by the spammers in the same groups, which could render these normal users be mistakenly treated as spammers and cause more false alarms. Moreover, in these approaches, the detection algorithms are strongly coupled with the proposed features, resulting in the lack of flexibility to incorporate new features for model improving. In this thesis, more fine-grained features are explored and more flexible detection frameworks are developed to tackle collusive reputation fraud in unsupervised settings, which will be discussed in Chapter 4.

Another problem of existing unsupervised approaches is that these approaches cannot learn from existing data for making predictions about emerging collusion events. In fact, predictive models are crucial to fraud detection tasks as they are deployable in real-time scenarios, where emerging fraud practices can be promptly detected and removed so as to minimize the caused damages. Existing unsupervised detection algorithms can only detect occurred collusive fraud activities in the given datasets, and are not suitable for online collusive fraud monitoring and detection. Moreover, being a deterministic model by nature, most of the existing approaches can be sensitive to the variability in the feature computations, which could be incurred by different reasons, such as specific parameter settings and noisy input data. In this thesis, these gaps are filled by seeking for collusive fraud detection solutions from a stochastic perspective which can naturally handle the uncertainty in the feature measurements. Moreover, from a modeling point of view, we design a model to not only infer occurred collusion in existing data as unsupervised models, but also learn to make collusion predictions as supervised models. Chapter 5 will elaborate the proposed statistical framework for collusive reputation fraud detection.

As discussed above, predictive models are plausible for collusive reputation fraud detection as they are deployable in real-time scenarios and can quickly respond to emerging fraud practices. However, the proposed predictive model in this thesis and other existing approaches rely on offline behavior analysis and build batch models on static rating history [37, 60]. As a result, the built batch models may quickly become obsolete once the

campaigns adopt different strategies. First, they can change attack timing with various intentions. For example, early attacks can be launched to hijack and preconceive an item's initial reputation [31, 37]. They can also generate massive ratings with opposite polarity right after dissenting votes to timely inverse the opinion orientation [24]. A built model may need frequent updates to handle attacks with changed timings. Second, reputation fraud campaigns can also change their constituents. To whitewash established attack histories, new accounts with clean background can easily be created or bought for new campaigns. Again, the batch approaches need to train a new model to account for campaigns with changed members. Finally, from an algorithmic view, existing batch approaches scan the input data multiple times for model building upon each single update, rendering it too costly to quickly respond to evolving campaigns. To overcome the above issues, this thesis will examine in Chapter 6 the paradigm of online detection of collusive reputation fraud, and propose a unified and scalable framework that can efficiently adapt built models to newly-emerging fraud patterns.

1.2 The Proposed Approaches

To address the problem of collusive reputation fraud in crowdsourced opinions, four pieces of work have been proposed in this thesis to effectively and efficiently detect reputation fraud campaigns. They are summarized as follows:

- **Graph-based classification algorithms for collusive reputation fraud detection [61]:** To defend against the recently discovered attack vector where multiple tiny colluder groups cooperate with each other on partially overlapped item sets, two novel classification algorithms are proposed which investigate and utilize the relationships between related users or user groups. The first is a KNN-based method, named FRAUDKNN, in which three pairwise similarity measures are proposed to define a neighborhood for collusive groups. In particular, these measures

examine the commonalities between groups in their member sets, target sets, and the review sentiments towards those targets. Then, for each user of interest, the classification is performed by considering the class labels of other users from similar groups. In this way, even though colluders can split into multiple tiny groups to evade detection, the relations between these tiny groups can still be captured by the proposed commonality measures used in the KNN classifier. The second is based on a probabilistic graphical modeling approach. In this approach, users' transaction correlations are utilized to associate potential colluders. The transaction correlations are formed once two users have co-reviewed any common item during the account lifetimes. This leads to the development of a probabilistic model named Colluder Random Field (CoRF), in which the attributes of users and their transaction correlations are jointly modeled for relational classification. A novel approximate inference algorithm is also developed to conduct collective classification on the model.

- **An unsupervised framework for collusive reputation fraud detection based on pairwise behavioral characterization [59]:** To mitigate the reliability of supervised models on fake review annotations which can often be noisy and hard to obtain, we have proposed a novel unsupervised detection framework, FRAUDINFORMER, for collusive reputation fraud detection. In this framework, the orchestrations of fraud campaigns are intuitively captured by a stochastic process that stimulates a mutual disclosure procedure among users. Through this procedure, colluders who are from the same campaigns would ultimately give each other away. Moreover, a suite of heterogeneous pairwise features are proposed to collect the collusion evidence. Different from existing group-based features, the characterization of pairwise features is more fine-grained, and can find stealthy colluders whose behaviors may not seem suspicious at group level. Finally, the proposed

framework decouples the processes of colluder characterization and detection, such that newly-designed features and specialized detection algorithms can easily and flexibly be integrated to achieve further performance improvements. Theoretical analysis has also been conducted to study the robustness of the proposed method in defending against adversarial challenge. We prove that FRAUDINFORMER can significantly limit the damage caused by a campaign even when it attempts to compromise the system.

- **A latent variable model for collusion prediction and inference in crowd-sourced opinions** [60]: To facilitate predictive models for emerging fraud campaign detection, we propose a novel statistical framework, Latent Collusion Model (LCM), to detect collusive review fraud from a hybrid of probabilistic generative and discriminative perspectives. Not only can LCM perform collusion inference as unsupervised models, but it can also make collusion predictions as supervised models. The key perspective of LCM is based on the appreciation of revealed characteristics that can differentiate colluders from non-colluders in particular feature space (e.g., [37, 61]). Colluders have been found to exhibit *unique collective behavior* patterns that result from their collaborations. It is then possible for LCM to build models for both tasks of collusion inference and prediction by taking a hybrid of generative and discriminative approaches. Specifically, the generative perspective enables LCM to formulate collusion inference as a clustering problem where collusions can be inferred by clustering their distinctive collective behavior patterns in feature space. The discriminative perspective, on the other hand, equips LCM the ability to learn a predictive distribution of user class labels for identifying potential collusions in the future reviews. Moreover, to complement existing collusion-related features and expand the feature space for collusive fraud characterization, a suite of *homogeneity-based* collusive behavior measures (h-CBMs) are also proposed to

distinguish colluders from non-colluders. Different from existing features used for colluder modeling, h-CBMs focus on the internal properties of fraud campaigns and capture intrinsic connections between colluders by correlating the behaviors of related colluders. In this way, even though colluders can restrict the campaign sizes or dominate the targets by review-flooding, the actions they need to finish the common tasks are inevitably homogeneous, which renders them detectable.

- **An incremental framework for reputation fraud campaign detection:**

To further address emerging campaign activities, we propose an online detection paradigm for collusive reputation fraud which can efficiently adapt built detection models to constantly-changing attack strategies. In particular, a unified optimization framework, FRAUDSCAN, based on *online learning* has been proposed to monitor and identify potential campaigns across time. The proposed framework is built on regularized matrix factorization models to capture spammers from the same campaigns among the large and sparse rating data. To effectively characterize emerging fraud campaigns, the notion of campaign context is introduced to monitor immediate interactions between campaigned spammers. Moreover, to accurately model users' sparse rating behaviors, an embedding-based approach is employed which can compactly encode user-rating-item relations with dense representations. In this approach, the proposed campaign context is integrated as model regularizers to map campaigned spammers to proximate regions in the embedded space so as to facilitate detection. Finally, to reflect the dynamics of emerging campaigns, a novel campaign-aware optimization scheme is proposed for online update. This framework is also fully unsupervised and operates without the need of annotations.

1.3 Contributions

The contributions of the proposed approaches are summarized below:

- Two novel supervised models are proposed based on probabilistic graphical models to address a new collusive attack vector that fails most of the existing approaches to reputation fraud detection.
- A unified detection framework based on stochastic process is proposed for unsupervised collusive reputation fraud detection. A suite of heterogeneous pairwise features are proposed to characterize collusive behaviors from different dimensions.
- The collusion prediction problem is addressed by proposing a statistical framework that can perform both collusion inference and prediction in unsupervised settings.
- The problem of emerging reputation fraud campaign is formally defined and a principled online detection framework is proposed which can adapt built detection models to newly-emerging adversarial patterns.

In summary, this thesis is focused on addressing the problem of collusive reputation fraud in online crowdsourced opinions in two scenarios (supervised and unsupervised settings) through collusive behavior characterization and designing detection algorithms that operate in two modes (offline and online modes). The ultimate goal of this thesis is to explore and analyze the motivation, orchestration, and weaknesses of the most recent and damaging collusive reputation fraud practices, and design effective and efficient detection algorithms to identify collusive reputation fraud as early as possible.

1.4 Organization

The outline of this thesis is shown in Figure 1.3. This thesis contains seven chapters. Chapter 1 introduces the background of the studied problem and the research problems.

Figure 1.3: The organization of the thesis.

Chapter 2 reviews related literature. Chapter 3 presents the supervised models for tackling stealthy collusive reputation fraud in crowdsourced opinions. Chapter 4 discusses the unsupervised detection framework and the pairwise features for collusive reputation fraud characterization and detection. Chapter 5 introduces the statistical framework for both collusion prediction and inference in crowdsourced opinions. Chapter 6 presents the incremental optimization framework for online collusive reputation fraud detection. Finally, Chapter 7 concludes the thesis and discusses future work. We briefly summary the subsequent chapters in this thesis as follows:

- Chapter 2, **Literature survey**: This chapter first reviews existing solutions to reputation fraud characterization in crowdsourced opinions, whose limitations in capturing collusive reputation fraud are discussed. It then surveys existing methods for reputation fraud detection in supervised settings, with the detection mechanisms elaborated and weaknesses in spotting collusive reputation fraud identified.

Next, unsupervised approaches for reputation fraud detection are reviewed and their drawbacks are pointed out. Finally, collusive reputation fraud uncovered in other Web 2.0 applications are reviewed, and the defense mechanisms and their shortcomings in addressing our problem are discussed.

- Chapter 3, **Graph-based classification algorithms for collusive reputation fraud detection (FraudKNN and CoRF)**: In this chapter, a new collusive reputation fraud pattern is uncovered and analyzed. Two graph-based classification algorithms are proposed to address this new attack vector in supervised settings, which utilize behavioral correlations between colluders for improving original classification performance. Extensive experimental results on real-world datasets validate the effectiveness of the proposed algorithms not only in general colluder detection, but also in addressing the newly-emerging adversarial pattern.
- Chapter 4, **An unsupervised framework for collusive reputation fraud detection (FraudInformer)**: In this chapter, to overcome the shortcomings of supervised models in collusive reputation fraud detection, an unsupervised detection framework is proposed which can also handle stealthy collusive attacks. Theoretical analysis has shown its robustness in defending against adversarial challenge. Real data-based evaluations confirm the effectiveness of the proposed framework and the superior performance over state-of-the-art competitors by spotting colluders with more concealed behaviors.
- Chapter 5, **A latent variable model for collusion prediction and inference in crowdsourced opinions (LCM)**: This chapter introduces the statistical detection framework to address the collusion prediction problem. The task of collusive reputation fraud detection from a stochastic perspective is identified and formally

identified. The framework is formulated based on the hybrid of probabilistic generative and discriminative views, which enables both collusion prediction and inference in a unified way. A suite of homogeneity-based collusive behavior measures (h-CBMs) are also proposed for characterization. Experimental results on real data show the effectiveness of the proposed h-CBMs and the superiority of LCM over state-of-the-art competitors in collusion inference and predictive abilities.

- Chapter 6, **An incremental framework for reputation fraud campaign detection (FraudScan)**: In this chapter, a principled and unified optimization framework is proposed for online reputation fraud campaign detection, so as to adapt built detection model to newly-emerging fraud patterns. It is based on the online learning scheme, and operates by monitoring reputation fraud campaigns across time. A novel concept - campaign context - is proposed for emerging campaign characterization. Extensive experimental results on real-world datasets show that the proposed framework is more accurate and efficient than state-of-the-art batch-based detectors.
- Chapter 7, **Conclusions and future work**: This chapter concludes the thesis and points out potential directions for future studies.

Chapter 2

Literature Survey

In this chapter, we survey related studies on reputation fraud detection in crowdsourced opinions. First, we introduce the terminology used in this chapter and throughout the thesis. Then, existing solutions to characterizing reputation fraud activities in crowdsourced opinions are reviewed. Their limitations in capturing collusive fraud activities are analyzed. Thirdly, this chapter surveys existing models for reputation fraud detection in supervised scenarios, where annotations of fraud instances are utilized for model building. It explains the mechanisms of these approaches in details and identifies their weaknesses in spotting collusive reputation fraud. Fourthly, algorithms developed for unsupervised settings are also reviewed, where no annotations are needed for detection. The superiority over supervised models is discussed, and their drawbacks on detecting collusive reputation fraud are dissected. Finally, collusive fraud practices discovered in other domains on the Web are also covered. The similarities and differences between different application scenarios are discussed, and the specific detection frameworks are introduced and compared.

2.1 Terminology

Before the presentation of our survey, we first introduce the terminology used in this chapter and throughout the thesis.

The aim of this thesis is to study the behaviors of collusive reputation fraud. Different from the general reputation fraud, collusive reputation fraud is one particular type where multiple spammers are asked to write fake reviews for the same set of items. In the real world, such collective practices are often formalized and organized as fraud campaigns.

Given a typical crowdsourced opinion platform such as review site or community, there are certain entities serving as the core to the entire reviewing process.

Definition 1 (Users, Items, Reviews, and Reputations). ***Users** are the subjects who write reviews for some items. **Items** are the objects receiving the reviews. **Reviews** are the vehicles for users to express their opinions towards the items about how good (in quality) they are, and **reputations** are the perceived public opinions of the items. A well functional and formatted user-generated review is typically comprised of several components. Without the loss of generality, in this thesis, we will consider the most common components of typical user-generated reviews that are available on many review sites. They are the **rating** expressing numerical opinions, the **text** expressing linguistic opinions, and the **timestamp** recoding the time associated with the reviewing event. It worth noting that due to their functionality in forming item reputations, these components are often abused by spammers for reputation manipulation. In the literature, the terms “reviewer” and “product” have also been used in the same way as the terms “user” and “item” respectively. In this thesis, we will use the terms “user” and “reviewer”, “product” and “item” interchangeably.*

Definition 2 (General Reputation Fraud and Spammer). *In general, **reputation fraud** abuses the de facto review mechanism in crowdsourced platforms, which involves a **spammer** being asked to write a deceptive/fake review with positive or negative opinion to a designated item, with the goal of either promoting or demoting the reputation of the item.*

Definition 3 (Collusive Reputation Fraud, Reputation Fraud Campaign, and Colluder). *This thesis studies collusive reputation fraud. In collusive reputation fraud, a group of spammers are organized and formed into a **reputation fraud campaign**. In a typical reputation fraud campaign, or **campaign** for short, spammers are instructed to conduct reputation fraud towards a set of designated items within a particular time period (see Figure 1.1 as an example). To further highlight the difference from the general spammers, in this thesis, the members of reputation fraud campaigns are called **colluders** who collude with each other in the campaigns.*

Figure 2.1: The illustration of reputation fraud campaigns, where two campaigns both with size two are shown. $P1 - 5$ are the targeted items. Colluders are colored in red, blue, and yellow, while normal users are not colored.

Figure 2.1 illustrates the orchestration of reputation fraud campaigns. Two campaigns are shown (i.e., A and B), both consisting of two colluders. They aim at different sets of items. Specifically, campaign A targets items $P1$, $P2$, $P3$ while campaign B targets

items $P3$, $P4$, $P5$. It worth noting that there is an overlap between these two campaigns; they both aim at the same item $P3$ and also share a common member (the user in red).

2.2 Reputation Fraud Characterization

In the literature, for both supervised and unsupervised scenarios, a large number of features have been proposed to characterize reputation fraud activities on various dimensions. In particular, the discriminating abilities of the features are vital to the ultimate detection performance. In general, these features can be divided into three categories according to the types of the entities from which these features are extracted. They are features focusing on the review objects, the users creating the reviews, and the fraud campaigns consisting of multiple users.

2.2.1 Review-centric Characterization

In reputation fraud, online reviews are the main vehicle for committing the fraud practice, thus rich evidence can be found in the fake reviews about the fraud activities that create these reviews. For example, a set of review-based features are used in [24, 31, 30, 42, 39] to build classification or ranking algorithms for fake review detection. In general, these features are focused on various aspects of online reviews on dimensions in terms of textual review contents and numerical review ratings.

Content-based features. In [24, 42], word or phrase level features are used to capture the traits of languages used in fake reviews. For example, the percent of positive (e.g., “beautiful”, “great”) and negative (e.g., “bad”, “poor”) opinion-based words is used to capture the intuition that fake reviews may use a lot of words in such kind to praise or criticize the reviewed items for reputation promotion or demotion. In addition, n-grams (e.g., uni-gram or bi-gram) of review text are used in [30, 42] to capture the wording patterns of fake reviews on vocabulary and phrase level. Based on the observation that,

for the sake of time and effort saving, spammers tend to copy or slightly modify existing fake reviews to create new ones, many studies propose features based on measuring the similarity between review contents [24, 31, 39]. In this case, *cosine similarity* between the n-grams of two reviews is commonly used to measure the degree of their overlapping in word or phrase level. Finally, other linguistic-based features such as the frequency distribution of part-of-speech (POS) in a review text and a set of keywords for psycholinguistic deception detection are employed for classification model building in [42].

Rating-based features. As an important channel for opinion expression, review ratings are widely used for feature construction. In [24, 31, 39], deviation-based features are proposed to evaluate how a particular rating is different from the item’s average rating. The assumption is that, while normal users would generally agree on the quality of an item by giving similar ratings, a spammer’s rating can be very different from others due to the objective of promotion or demotion. The larger the deviation, the more suspicious the rating is. The extremity of a rating is also examined [39]. On a 5-star rating scale, a rating with 1- or 5-star is considered extreme. For efficient promotion or demotion, spammers tend to give extreme ratings to aggressively alter the reputation of a target item (perceived as the average item rating).

Limitations. We analyze the limitations of the review-centric features in characterizing collusive reputation fraud activities.

For the content-based features, the major problem is that the text of a review can easily be manipulated so as to avoid detection. Strategic spammers are also human users who can carefully imitate the writing styles or wording patterns of normal users. For example, in [50], the authors show that one can easily synthesize a review that looks genuine by simply organizing sentences extracted from existing authentic reviews written by experienced users. As a result, many state-of-the-art fake review classifiers built on content-based features are shown to fail to detect such synthesis reviews. In

addition, from a psychological perspective, it has also been shown that humans are poor at identifying deceptive reviews by just reading the review text [42].

Second, the major problem of rating deviation-based features is that, in collusive reputation fraud, by taking advantage of manpower, colluders can give many ratings with similar stars at early stages so as to dominate the average rating of a target item. As a result, their ratings become the majority and are very close to the item’s average rating, rendering deviation-based measures invalid to indicate deceptive ratings.

In this thesis, we have proposed methods that not only can incorporate content-based features for collusive reputation fraud characterization and detection, but can also exploit features derived from user reviewing behaviors that can be more robust to manipulation. On the other hand, to address the problem brought by the deviation-based features, we have developed a set of homogeneity-based features (Chapter 5) by studying the interrelationships between colluders, which can still be useful when the deviation-based features become obsolete.

2.2.2 User-centric Characterization

Another category of features is imposed on the authors of fake reviews. The philosophy behind is that the collective examination of a user’s all historical reviews would give us more information about the suspiciousness of his/her behaviors and thus enrich reviewing behavior characterization. Moreover, it also facilitates the comparison between different users. To this end, different dimensions of reviewing behaviors of users are explored in the literature.

Content-based features. Different from the utility of the text of a single review, the characterization on users can exploit all the reviews created by a specific user. In [31, 39, 14], the authors evaluate the *cosine similarity* between any pair of the reviews contributed by a user, so as to account for the observation that spammers are likely to copy or make minor adjustments to their own reviews across similar items.

Rating-based features. Similar to the case of charactering a single review, deviation-based features are used to measure in general to what extent a user would give different ratings from others on the rated items [24, 31, 39, 14]. Normally, maximum or average functions are used for summarization. Another way of utilizing a user's ratings is to compute the standard deviation or variance of those ratings [24, 30]. This is to capture those spammers who always give good or bad ratings for reputation promotion or demotion. In such case, the standard deviation of their ratings will be very low. In addition, to account for the situation where a spammer may give multiple similar ratings to a target item, in [31], the authors measure the similarity between the ratings given by a user to the same items. Users with a lot of ratings involved as multiple similar ratings to items are considered suspicious.

Temporal-based features. When considering the historical reviews posted by a specific user, the timing of these postings becomes important to indicate reputation fraud.

The first kind is to capture the **burstiness** of reputation fraud activities. The burstiness is embodied in two aspects. First, the fraud activities would occur within a specific time period, which would lead to sudden concentrations of fake reviews posted in this time interval. To capture this observation, in [14], the authors first locate the time intervals in bursts, then for each user, they compute the ratio of the reviews in bursts to the total number of reviews. The higher the proportion, the more likely the user is to be a spammer. Second, the usage cycle of a spammer account can also be bursty, since spammers are commonly not longtime users of a site [37]. Therefore, the usage cycle of a spammer account would be relatively shorter than that of normal users. In [39], the difference between the posting times of the first and last reviews of a user is used to evaluate the persistence of the account.

The second kind of features that focuses on review-posting timing is to capture the **earliness** of reputation fraud activities. To maximize the manipulation impacts and

attract more attentions, spammers would post fake reviews as soon as the target items are available, such that the overall item reputations can be quickly dominated. Existing studies propose features with different formulations to capture this observation. In [24, 39], first-reviews are used as an indicator for earliness characterization. First-reviews refer to the earliest reviews posted for the products. In both studies, for each user, the ratio of the number of his/her first-reviews to the total number of reviews is used for feature construction. Different from the plain counting, in [31], a weight based on exponential decay is used to indicate the earliness of a review being given; the early the review posts, the higher the weight is.

Limitations. We discuss the limitations of the user-centric features in characterizing collusive reputation fraud. First, these features are designed to operate on individual users and cannot capture the collective behaviors of a group of spammers who work for the same campaigns. In collusive reputation fraud, multiple colluders can share the overall spamming workloads so as to not appear aggressive and behave as normal users. Moreover, the user-centric features may also suffer from the domination problem as discussed in the case of review-centric features, where colluders can easily become the majority on the overall opinion about an item by posting many fake reviews at early stages. In this case, when the deviation-based features are used, any subsequent normal review with a different opinion may be regarded as an outlier.

In this thesis, our main focus is on the design of group-based features that capture the collective behaviors of colluders. Moreover, as discussed before, we characterize and exploit the interrelationships between colluders to solve the reputation domination problem caused by deviation-based features.

2.2.3 Campaign-centric Characterization

The most effective way of characterizing collusive reputation fraud is to design features that work on multiple related users, so as to capture the underlying collusive behav-

iors. In this respect, existing studies propose to characterize the collective behaviors of a group of users. For example, Mukherjee et al. [37] propose a set of Group Spam Behavior Indicators (GSBIs) for suspicious collective behavior modeling. These features are imposed on the groups of users who have co-reviewed multiple items together, which can be generated by Frequent Itemset Mining (FIM) algorithms. In [37], the resulting groups contain at least two users who have co-reviewed at least three items. To further distinguish between malicious groups (i.e., colluders) and normal groups which form due to similar user preferences, different dimensions of collective behaviors of user groups are investigated. In temporal aspect, both burstiness and earliness properties of collusive reputation fraud are examined. Specifically, the *Group Time Window* is designed to capture the burstiness of collusive fraud by measuring the activity window of the members of a group. In terms of ratings, the deviation can now be computed by comparing between the ratings given by the members of a group and others outside the group (*Group Deviation*). The earliness of collusive fraud activities can also be aggregated at group level, using summarization functions such as *max* or *average* (*Group Early Time Frame*). In addition, regarding review contents, textual similarity can be calculated and aggregated on the reviews posted by a group to capture the review composing behaviors of group members (*Group Content Similarity* and *Group Member Content Similarity*). Finally, the properties of a group as a whole can also be exploited, such as the size of a group (*Group Size Ratio* and *Group Size*) and the number of items reviewed by a group (*Group Support Count*). In particular, groups with large sizes and/or target sets are considered more suspicious in [37], since the scale of normal groups that randomly form due to users' common preferences is found to be small.

Limitations. The proposed GSBIs in [37] are the first effort to tackle collusive reputation fraud and shown to be effective in capturing the collective behaviors of campaigned spammers. The main problem of GSBIs stems from the underlying assumption that

collusive fraud activities are distinctive in two aspects: (1) they are rare; thus they can be treated as minorities whose behavior patterns deviate largely from those of the general public, and (2) they are aggressive; thus groups with more manpower are considered more likely to be related to collusive fraud. This assumption, however, would render GSBIIs suffer from more stealthier attacks discovered recently [61] where multiple tiny groups are observed to cooperate with each other on targets that are only partially overlapping. In this case, the *Group Size* and *Group Support Count* that focus on the *external* properties of groups would fail to characterize such tiny groups. Moreover, deviation-based features such as *Group Deviation* and *Group Size Ratio* would also suffer from the same problems identified in the cases of review- and user-centric features. By leveraging the inherent advantage in manpower, colluders can easily turn themselves into the majority via overwhelming the items, such that the deviations are eliminated.

In this thesis, we have developed a suit of homogeneity-based features that capture the *internal* structures of fraud activities (Chapter 5). Compared to the external properties of colluder groups, their internal structures are much harder to fabricate given their collective efforts on the same fraud campaigns.

2.3 Supervised Reputation Fraud Detection

There has been a large body of work on the detection of reputation fraud through training supervised models. In these approaches, the detection task is formulated as a binary classification problem where fraud activities (e.g., fake reviews, spammers, or spammer groups) and normal activities are regarded as positive and negative instances respectively. The classification task can further be boiled down into three sub-tasks. The first is to find a set of representative, real instances of reputation fraud in the wild, which are used to prepare a *golden-standard* training set for subsequent classification model building. Then, a feature space is constructed to characterize reputation fraud activities. After

this phase, each object of interest (e.g., can be a review, a user, or a user group) in the training set is represented as a feature vector in the feature space. Finally, classifiers are built utilizing the training set; once trained, the classifiers are used as detectors to label a new object as positive (i.e., reputation fraud instance) or negative.

There are studies building classification models for fake review detection. In [24], the authors build classifiers based on logistic regression to detect various types of spam reviews, including deceptive reviews (i.e. fake reviews in our context), non-specific reviews (reviews not on the specific items), and non-reviews (e.g., ads). The hardest task is to detect deceptive reviews written for misleading users' perception. As annotating true deceptive reviews is difficult and unreliable, the authors use duplicate reviews as the golden standard. However, the main disadvantage of this model is that not all deceptive reviews are duplicate ones. Thus, the model built on duplicate reviews only captures partial characteristics of deceptive reviews which are the overlap between these two kinds. Ott et al. [42] study the effectiveness of classifiers trained on different kinds of review textural features, such as n-grams, POS tags, and psycholinguistic word bags. In this work, the golden standard fake reviews used for training are built based on crowdsourcing services. Specifically, a group of workers on Amazon Mechanical Turk are paid to write deceptive reviews for 20 hotels. For classification model building, Naive Bayes and Support Vector Machine (SVM) are employed, which are popular classifiers used in text categorization literature. In [30], semi-supervised approach is used to exploit the large number of unlabeled review data to boost the classification performance. Specifically, a two-view semi-supervised method is proposed which considers both review- and user-centric features at the same time. The model is learned based on a co-training scheme, where the model is incrementally trained in the sense that each time after training, the model is immediately applied to the unlabeled data, so as to generate more annotations of fake or truthful reviews.

Limitations. In terms of collusive reputation fraud detection, the main drawback of these supervised approach is that they ignore the interactions between campaigned spammers; the proposed classification models are all trained on individual instances and cannot incorporate relational information extracted from multiple users into model building. Moreover, these supervised models inherently hinge heavily on the annotations of fake reviews or spammers. Different from other annotating tasks in related problems such as sentiment analysis or relevance evaluation, the identification of fake reviews is at least as difficult as the exposure of human lies [42]. There is a lack of ground truth in practice whether a review is fabricated or not [24, 31, 56].

In this thesis, we have proposed some novel classification frameworks (Chapter 3) that not only can utilize the attributes of individual instances as traditional approaches, but can also take collusive relationships between multiple colluders into account. Moreover, to overcome the difficulty of ground-truth data acquisition, we have proposed three novel unsupervised approaches (Chapter 4~6) that can work without the need of annotations.

2.4 Unsupervised Reputation Fraud Detection

To overcome the disadvantage of supervised detectors in hinging on fake review annotation, many recent approaches are proposed to work in unsupervised settings, where the detection models are not trained using labels. Instead, to conduct detection, these approaches design algorithms that associate each entity of interest (e.g., a review, a user, or a user group) in the data with a numerical score. These scores or spamicities called in several pioneering works essentially quantify the suspiciousness of the entities in committing reputation fraud. Finally, a ranking can be generated based on the spamicities and those who are ranked at the top are considered as the most suspicious ones.

2.4.1 Individual Spammer Detection

A large body of studies are focused on the detection of individual spammers. Lim et al. [31] design different user-centric features to model the behaviors of individual spammers. Specifically, all the feature scores are normalized between $[0, 1]$, with values closer to 1 indicating more severer fraud behaviors. A weighting scheme is used to obtain the overall user spamicities based on all the proposed features. Then, the users are ranked according to their spamicities in descending order. For ranking performance evaluation, Normalized Discounted Cumulative Gain (NDCG) is used which favors rankings where all spammers are ranked higher than all non-spammers. The primary disadvantage of this approach is that there is a lack of principled and optimal ways of determining the weight assigned to each feature in the feature combination.

Wang et al. [54] propose a heterogeneous graph model for review spammer detection. In particular, the intrinsic relationships between the three entities in a review site, i.e., reviews, users, and stores, are extensively examined. The critical observation they make is that a user is more trustworthy if she has written more honesty reviews, a store is more reliable if it has more positive reviews from trustworthy reviewers, and a review is more honest if it is supported by many other honest reviews. Different sets of features are then designed to model the three important concepts, i.e., trustiness of users, the honesty of reviews, and the reliability of stores, which are coupled and affect each other. Moreover, the dependencies between trustiness, honesty, and reliability are also captured using logistic functions. For example, the dependency of the trustiness of a user u on the honesty of the reviews she writes is modeled as follows,

$$T(u) = \frac{2}{1 + e^{-H_u}} - 1 \quad (\text{Eq. 2.1})$$

where $T(u)$ is the trustiness of u , H_u is the honesty summation of all the reviews written by u . Finally, an iterative framework based on power iteration is proposed to compute

the spamicities for all the users, reviews, and stores in the data, based on the modeling of their interrelationships. From the technical point of view, this is the first time a graph-based model is used for detection. The benefit is that it largely extends the space for searching clues for spammer detection; to decide whether a user is a spammer or not, the graph-based approach not only examines her own reviews and the reviewed items, but also indirectly considers all the information about other users who have also reviewed the same items. In contrast, in previous studies, only the one-hop information is used for modeling. However, when it comes to collusive fraud detection, the approach used in this work does not capture or build models on the relationships between users for campaigned spammer detection.

A similar propagation-based computation framework is proposed in [8], which adopts the approach of anomaly detection by assigning an anomaly score for each user and item. In this work, the basic assumption is that spammers should be the minority who hold different opinions with those of the general users who are assumed to be the majority. To further detect spammers, they make two observations that normal users are honest in general, who would give positive ratings to good items and negative ratings to bad items. On the other hand, spammers are dishonest in that they would give negative ratings to good items for demotion and positive ratings to bad items for promotion. Given such principles, the anomaly score computation can be realized in such way that, the anomaly score of a user is positively correlated (multiply by a positive factor) with those of the items (s)he agrees (i.e., giving it a positive rating), and negatively correlated (multiply by a negative factor) with those of the items (s)he disagrees (i.e., giving it a negative rating). The same principle applies to the computation of item anomaly scores. The computation proceeds in a similar way as the HITS algorithm where the anomaly score of a(n) user (item) is obtained by taking the average over those of the items (users) (s)he (it) connects to. However, this approach also suffers from the same problem that it is

not optimized to catch collusive reputation fraud where colluders can easily become the majority by investing more manpower.

In recent years, more graph-based approaches are proposed for reputation fraud detection. Fei et al. [14] propose a probabilistic graphic model based on Markov random field (MRF) to capture the burstiness of fraud activities. In this work, the time intervals within which concentrative activities are observed are identified as burstiness areas. Users who post reviews in the burstiness areas are considered as candidates of review spammers. Then, a MRF model is proposed to find true spammers by connecting users who are in the same burstiness areas in a probabilistic graph. In this model, different features are designed to formulate the potential functions used for model inference. Finally, a loopy belief propagation algorithm (LBP) is developed to solve the spammer inference problem defined on the MRF model. Compared to the previous work on graph model design [54], the MRF-based approach works in a more principled manner; it formulates review spammer detection problem as a probabilistic inference problem, and demonstrates how to model spammer behavior from a probabilistic perspective. However, the main drawback is that only the burstiness property of reputation fraud is utilized for detection, which may miss other spammers who do not post reviews in a concentrative way.

Another similar but more general approach based on Markov random field is the FRAUDEAGLE framework proposed in [1]. In this work, a signed bipartite review network is created to link users, reviews, and items. In the network, each edge represents a review that links the author to the reviewed item. Different from the MRF model in [14], the potential functions in FRAUDEAGLE capture the intuition that the probability of a honest user giving positive (negative) reviews to good (bad) item, or the probability of a spammer giving positive (negative) reviews to bad (good) items should be higher than their counterparts. The model inference is also based on loopy belief propagation, and at the end the probability of each entity (i.e., a user, a review, or an item) being related to fraud is computed and ranked.

A more recent model built on Markov random field is proposed in [45]. This model, named SPEAGLE, is based on the former framework FRAUDEAGLE. Different from FRAUDEAGLE, more side information such review contents, review timings, and ratings are incorporated into the model for achieving further improvement. In particular, all the explored side information is used as prior potentials in MRF for estimating the prior probabilities of users, reviews, and items of being related to fraud. From a modeling perspective, reviews are also modeled as random variables in the model while in FRAUDEAGLE they are represented as edges that do not have their own distributions. This enables the estimation of spamicity scores for reviews and facilitates the formulation of prior potentials for reviews.

Mukherjee et al. [39] propose a generative probabilistic model to capture the generation process of fake reviews from review spammers. In this model, the spamicity of a user is modeled as the latent factor that causes (generates) her observed spamming behaviors (substantialized as the features discussed in Section 2.2), which are modeled as binary random variables with Beta distributions. Finally, the inference and estimation of the model yield the spamicities of users and the distributions of the behavior variables.

However, a major problem of the unsupervised approaches discussed above is that they cannot learn from existing data for making predictions about emerging fraud activities. In fact, predictive models are crucial to fraud detection tasks as they are deployable in real-time scenarios where emerging fraud practices can be timely detected and removed so as to minimize the caused damages. Given a collection of historical reviews as input, these approaches can only detect the occurred fraud activities in the provided data. No models are learned for predictive purposes. Moreover, all the proposed frameworks are designed to operate in the batch-mode; the models need to be rebuilt upon the arrival of new reviews so as to identify emerging fraud activities. It thus becomes too costly for these approaches to quickly respond to evolving fraud activities.

In this thesis, we have developed unsupervised approaches (Chapter 5) that not only can perform collusion inferences as traditional unsupervised models, but can also conduct collusion predictions by learning predictive models from existing data. Furthermore, we have proposed a novel online detection framework to efficiently and effectively adapt and respond to emerging fraud activities (Chapter 6).

2.4.2 Fraud Campaign Detection

The most related work to ours is the solutions proposed to tackle reputation fraud campaigns, where a group of colluders collaborate on the same fraud task. Campaign based reputation fraud is a novel attack vector uncovered in recent years, and has not received much attention so far. However, it is considered more damaging than fraud conducted by individuals, as it has more manpower to quickly dominate the reputations of the targets. Moreover, the colluders can disguise themselves as normal users by spreading the overall workloads over multiple peers.

In this realm, a specific type of colluders are noted and explored in [58] which is called *singletons*. Singletons are users who post very few reviews within their lifetimes. Most of the features used in previous studies would fail to capture such singletons. For example, the mean and standard deviation of ratings given by a user become meaningless if this user has only one review. To address this issue, the authors examine the behaviors of users on temporal dimension. The observation is that to maximize the utility of a campaign, singletons are more likely to aggregate in a short time interval, where the average rating of a target item, the number of reviews it receives, and the ratio of singleton reviews it receives would also change considerably. A detection framework based on temporal curve fitting and longest common sub-sequence (LCS) matching is developed to find those abnormal intervals that are temporally correlated across the parallel time series built on the above three quantities. However, a major problem of this approach is that it does not

consider the quality of the stores. It may generate too many false alarms when popular items or items in promotion are sold at a store, since such events may also cause the burstiness of all the three quantities.

The work of [37] is the most related to ours on reputation fraud campaign detection. In this work, frequent itemset mining algorithms are first used to obtain candidate user groups who co-review multiple items together. Provided the fact that some of the candidate groups may just form due to users' co-preferences and thus should be treated as innocent ones, a set of Group Spam Behavior Indicators (GSBIs) are then proposed to further capture the suspicious collective behaviors of all candidate groups (see Section 2.2.3 for more details). Finally, an HITS-like power-iteration algorithm, named GSRank, is proposed to generate spamicity scores for the candidate groups by iteratively propagating GSBIs among three entities - groups, group members, and items. From a technical point of view, a major problem of this propagation-based approach is that it may become problematic to handle colluders who use camouflage to cover up their malicious behaviors. For a spammer, the camouflage can be achieved by creating many authentic reviews for some random items, so as to make themselves be clustered together with the (normal) users who have reviewed the same items. As such, the normal users who are in the same groups with spammers may gain undeserving high spamicities from the spamicity-propagation through those spammers if they accidentally share similar preferences or opinions towards the same items, which would result in higher false positive rate. Another problem of this approach is that, the features (i.e., GSBIs) and the detection algorithm (i.e., GSRank) are tightly coupled with each other in the detection, which makes it inconvenient to incorporate new features for performance improvements.

In this thesis, to overcome the mentioned drawbacks of spamicity-propagation based methods (e.g., GSRank), we have proposed a different way of utilizing group-based indicators, which yields a unified statistical framework that directly models the distributional

patterns of suspicious collective behaviors and the normal/accidental ones (Chapter 5). In addition, to address the problem of tight coupling between the features and the algorithms/models in existing methods, we have proposed approaches that can flexibly incorporate any features available for model building.

2.5 Other Collusive Fraud Activities on the Web

Apart from the domain of online crowdsourced opinions, collusive fraud has also been unveiled in other scenarios, such as e-mail systems [27, 48, 28], the Web [57, 2, 6, 20], and more recently, the social networks [18, 19, 62].

In email systems, the work of [27] reveals the orchestrations of email spam campaigns which consist of a target list of email addresses and a set of templates for creating spam emails. Other attributes of email spam campaigns are exposed such as the durations and scales, and most importantly the underlying infrastructures that support the spam campaigns. They also show the dynamics of how such campaigns unfold, including the analysis of the targeting strategies of different campaigns, the delivery failure rate, etc. In comparison, abstractly, reputation fraud campaigns also have item lists to target and spammers may have templates for creating fake reviews. However, the financial incentive of reputation fraud is different. For email spam campaigns, the goal is to effectively and efficiently spread spam emails as much as possible, while for reputation fraud campaigns the aim is to inflate or defame the reputations of the targets through review manipulation. It is not necessary for these campaigns to post too many reviews, but it is a must to maximize the impacts brought by the posted fake reviews.

The goal of Web spam campaigns [57, 2, 6, 20, 9] is to deliberately manipulate search engine indexes so as to promote the ranks of spam pages in the search engine. The means includes the *content spam* and *link spam*. Content spam involves the addition of irrelevant keywords in Web pages to fool search engines. However, review spammers do not add

irrelevant words in their reviews since their goal is to deceive potential readers. Link spam is about embedding hyper-links in Web pages so as to fool ranking algorithms used by search engines such as Pagerank [43]. However, deceptive reviews typically do not include hyper-links. In [9], a novel notion called extreme rank anomalous collections (ERAC) is defined to capture Web spammer hosts that adopt similar strategies to promote their Web pages. When they are ranked by various features, they are found to be ranked toward the top or bottom area on particular features collectively. Apriori-like algorithms are then developed to find the top-K anomalous collections that contain a larger number of entities ranked at more extreme positions on more features. However, due to the focus on large anomalous collections, this approach may fail to handle stealthy attacks that involve many small collusive collections.

In social networks, Beutel et al. [3] defend against collusive fraud in Facebook Page Likes. They represent a Page Like dataset as a bipartite graph that establishes connections between the pages and the users who Like them. An iterative optimization algorithm is then implemented to infer synchronized group attackers who co-Like a non-trivial number of pages in a short time period. Yang et al. [62] implement a pagerank-like algorithm to infer malicious user accounts in Twitter where some criminal accounts are found to cover for each other through mutual endorsements. Another propagation-based approach is also proposed to identify collusive link farming in Twitter [19]. Another kind of collusive adversarial practice found in social networks is the Sybil attack, where an attacker can create a large number of dummy accounts with the goal of undertaking illegitimate activities such as spreading spam messages or generating fake Likes. To defend against Sybil attacks, the work of [10] invents the concept of coherent anomaly collection to capture and group similar accounts that exhibit coherent anomalous behaviors. The coherency is used to capture multiple disjoint subsets of an anomaly collection whose members (i.e., user accounts) are more likely to be controlled by different spammers. A

heuristic algorithm based on information theory is then proposed to perform detection. In [64], a ranking-based detection framework is proposed to improve existing anti-Sybil systems by utilizing trust and distrust information in social networks. In this framework, a graph pruning-based method is first proposed to generate reliable Sybil seeds by exploiting local structures of social networks. Then, a ranking algorithm based on Pagerank is developed to score the trustworthiness of accounts in the social network by propagating trust and distrust information from the Sybil seeds across the entire network, such that suspicious accounts would obtain low trustworthiness scores. In [65], a semi-supervised model is proposed to enhance existing defense systems by exploiting relational information between accounts in social networks. The underlying assumption is that the class labels of similar accounts in the network should also be similar. The similarity between user accounts is mainly captured by the existence of their direct friendships and the number of their co-friends. However, as no friendship links exist among users in most review sites, the above approaches cannot directly apply to the context of online reviews. Moreover, all the aforementioned approaches are inference-based, and still suffer from the same issue that they cannot learn from existing data for predictive purposes. Finally, these approaches are not designed to work freely with other useful collusion features since the features are hard-coded in the corresponding detection algorithms.

Collusive attacks have also been studied in reputation systems, where crowdsourced opinions (typically in the form of ratings) are collected and accumulated to dynamically compute a reputation score for each object of interest (e.g. buyers, seller, services, or goods) in the system based on a specific reputation algorithm. Instead of detecting reputation fraud directly, most of the studies in this realm focus on *designing reputation systems that are robust to collusive attacks*. In [32], a clustering-based approach named iCLUB is proposed to handle unfair ratings in reputation systems. The system is shown to be robust to a specific class of collusive attacks where sellers collude with buyers

to generate unfair ratings for the sellers. Jiang et al. [23] propose a novel reputation algorithm based on evolutionary computation which is robust to many kinds of reputation attacks. The class of collusive attacks studied and evaluated is similar to that in [32]. In [53], the robustness of reputation systems to collusive attacks is theoretically analyzed and quantified based on information theory. Different types of collusive attacks are analyzed and the strongest possible collusive attacks (from an information theoretic view) are identified. A novel reputation system is then proposed which is robust to such attacks. The research problem in this thesis complements the above studies by designing algorithms that can directly *detect* collusive reputation fraud in crowdsourced opinions. In practice, any reputation system can be equipped with the supervised and unsupervised detection algorithms proposed in this thesis to tackle collusive reputation fraud in both offline and online scenarios.

2.6 Summary

In this chapter, we extensively review related researches in the literature on reputation fraud detection in crowdsourced opinion platforms and other Web applications. In the literature, the solutions to the detection of reputation fraud in crowdsourced opinions involve two sub-tasks – fraudulent behavior characterization and fraud activity detection. Specifically, various kinds of features have been proposed to characterize reputation fraud activities in different granularities, including review contents, user reviewing behaviors, and collective behaviors of user groups. However, it is still insufficient to rely on these features to deal with more recent and advanced collusive attack formulations. On the other hand, different detection algorithms have been developed to identify reputation fraud activities, however, most of them have their own deficiencies in collusive behavior modeling, collusion predictive ability, feature incorporation flexibility, and emerging

campaign sensitivity. All the identified issues will be tackled by our proposed solutions to collusive reputation fraud detection that are to be discussed in the subsequent chapters.

Chapter 3

Graph-based Classification Algorithms for Collusive Reputation Fraud Detection

As mentioned in earlier chapters, the ultimate goal of this thesis is to effectively identify collusive reputation fraud so as to diminish the adversarial impact to online review portals. Traditionally, the detection of collusive reputation fraud is approached via being formulated as a binary classification problem where colluders and non-colluders are treated as the positive and negative instances respectively. During this process, the design of effective feature space is crucial to the ultimate detection performance. In this chapter, to study the latest collusion fraud practices in online review portals, we collect and create a large golden-standard dataset from a popular online review portal (Section 3.1). Using this dataset, we empirically analyze 25 state-of-the-art collusion features proposed in previous studies, to see whether they are effective to uncover more recent collusive fraud practices (Section 3.2). Notably a novel collusion pattern is disclosed from our analysis where multiple tiny groups are observed to cooperate with each other on targets that are only partially overlapping, which causes the failure of many existing features to correctly classify colluders and non-colluders. To overcome the new attack vector, two graph-based classification algorithms are proposed to spot collusive groups

in supervised settings by capturing correlations between collusive groups (Section 3.3 and 3.4). Experimental results are shown in Section 3.5, and Section 3.6 summarizes this chapter.

3.1 Data Acquisition and Preparation

To study the latest collusion reputation fraud practice in the wild, we need to create a gold-standard dataset that contains annotated colluders and non-colluders. In particular, we turn to consumer reviews from Amazon.cn¹, by crawling a snapshot of all manufacturing product reviews till August 20, 2012. We name this dataset as AmazonCn which contains 1,205,125 reviews written by 645,072 users on 136,785 products (e.g., electronics, housewares). Each review has 6 attributes, i.e., user ID, Product ID, Product Brand, Rating, Date and Review Text. We select Amazon.cn because: 1) Popularity: it is one of the most popular e-commerce websites in China; 2) Abundance and Variety: the huge number of consumer reviews cover a wide range of products; 3) Comparability: many existing studies such as [31, 15, 54] used datasets from Amazon.com for investigation and evaluation. Thus, we believe that the analysis on the review data from Amazon.cn, which shares similar scheme with Amazon.com, should be comparable to previous studies.

3.1.1 Colluder Sampling

To create a dataset that holds sufficient colluders for evaluation, the first task is to search for the places where colluders would probably be found. A good way to achieve this is to use frequent itemset mining (FIM), similar with [37]. In such context, user IDs are regarded as *items*, each transaction is the set of user IDs who have reviewed a particular product. Through FIM, groups of users who have reviewed multiple common products can be found. Here we use *maximal* frequent itemset mining (MFIM) to discover

¹The Chinese counterpart of Amazon.com (NASDAQ: AMZN), which is one of the most popular electronic commerce sites in China.

groups with maximal size since we focus on the worst fraud activities in our dataset. Following the same parameter settings as in [37], totally 8,915 groups are found, with each represented as a mapping from a set of members (≥ 2) to a set of commonly reviewed products (≥ 3). Finally, we merge all the members of each group into a collection $\mathbf{V}$ which consists of 5,055 users in all.

3.1.2 Annotation

Each user in $\mathbf{V}$ should be annotated as either colluder or non-colluder. It is acknowledged that users who express fake opinions in review websites are difficult to identify due to the lack of ground truth. Fortunately we noticed that Amazon.cn is practically clearing up the displayed consumer reviews periodically. We also confirmed that typically a displayed review will be deleted if it has been regarded as fake or if its content has been confirmed to be irrelevant to the specific product, by the website moderators. Thus it would facilitate the annotation process by locating those deleted reviews in our dataset. We then re-crawl all the reviews written by users in $\mathbf{V}$ on March 25th, 2013, and find that 1,822 out of 5,055 users have at least one of their reviews being deleted during the near seven-month time period². However, as we find that not all fake reviews are removed from the system, manual annotation is also introduced to refine the initial annotations. Identifying suspicious fake reviews is a very challenging process which involves considerable efforts on user behavior analysis via searching, collecting, and comparing related reviewing histories. However, the identification of fake reviews posted by colluders might be relatively easier since one can compare the reviews posted by the group members. The manual annotation process is thus carefully conducted and strictly guided by a list of guidelines, which specify signals that are considered to be closely related to collusive fraud. These guidelines are desirable since some of them have been used in previous studies for the

²In Amazon.cn, users themselves cannot delete reviews.

same kind of annotating task [37, 24, 54], and others are obtained from various online resources compiled by experts with substantial experiences on opinion fraud detection. Some examples are listed as follows:

- *A group is suspicious if the sentiments of the reviews posted by the group members towards most of their common products are the same.* While it is reasonable for a group of users to either like or dislike one or two common products at the same time, it can be problematic if they agree on the same products almost all the time.
- *A group is suspicious if the texts of the reviews posted by the group members for most of their common products are similar.* For example, if multiple users always use lexically or semantically similar words or phrases to express their opinions towards the same products, it is very likely that they are instructed to do so as a group.
- *A group is suspicious if each of the group members always gives either positive or negative ratings.* It has been revealed that the ratings given by general spammers are often extreme and highly skewed [24, 39]. If the members of a candidate group all exhibit such a pattern, this group is very likely to be a colluder group.
- *A group is suspicious if each of the group members posts reviews at concentrative times.* Posting reviews within short time intervals is another trait of general spammers [14, 61]. One reason is that, after being created, the spammer accounts are found to be used mostly for conducting a series of spamming tasks subsequently. If the group members all have such an aggregative usage pattern, the entire group can be suspicious.

As a result, a total of 3,118 non-colluders and 1,937 colluders are identified. We argue that nowadays for many popular review websites like Yelp and Amazon, it is common to have anti-fake teams to harness fake/spam reviews over the sites. They

typically hold information that would be more useful for fake review detection (e.g., IP Information of users) which is not publicly available. Nevertheless, for those sites that do not continuously clean up fake reviews, the practical fake review related signals presented in aforementioned sources may facilitate the annotation process to a great extent.

3.1.3 Statistics

Here we show basic statistics about the obtained Amazon.cn dataset, which are summarized in Table 3.1.

Statistics	Value
Number of users (Number of colluders, Number of non-colluders)	5055 (1937, 3118)
Number of products	4898
Group size (Min., Mean, Median, Max.)	2, 3.5, 2, 26
Number of products reviewed by a group (Min., Mean, Median, Max.)	3, 3.4, 3, 20
Number of groups a user belongs to (Min., Mean, Median, Max.)	1, 5.9, 1, 261

Table 3.1: Statistic summary of our Amazon.cn dataset.

Beside the basic statistics, we also show statistics about the distributions of the annotated two classes, i.e., colluders and non-colluders. First, we are concerned about the scales of the groups a colluder or non-colluder belongs to. It has been reported that the colluder groups are generally larger than the non-colluder ones which are usually formed by random users [37]. As a user can be involved in multiple groups, we compute the average group size. Figure 3.1 shows the cumulative distributions of the average size of groups a colluder or non-colluder belongs to. It can be seen that the result is consistent with the aforementioned findings; there is a clear gap between the two distributions, with the curve of colluders leaning towards the large-value region.

In addition, we also examine the general difference between the numbers of products reviewed by colluders and non-colluders. The result is shown in Figure 3.2. It can be seen that in this dataset colluders in general review less products than non-colluders.

Figure 3.1: Cumulative distributions of the average size of the groups a colluder (red) or non-colluder (blue) belongs to on AmazonCn.

Figure 3.2: Distributions of the number of products reviews by a colluder (red) or non-colluder (blue) on AmazonCn.

3.2 Empirical Analysis of State-of-the-art Reputation Fraud Indicators

Effective indicators to collusive reputation fraud are crucial to accurate colluder detection. Prior to our work, existing studies have proposed a variety of indicators from different perspectives of the review context. In this section, we perform empirical analysis of these indicators on our AmazonCn dataset from a *colluder-centric* view, i.e., the indicators are categorized into three categories - linguistic indicators, individual behavioral indicators and collusive behavioral indicators. Specifically, linguistic indicators characterize the text of reviews written by a user. Individual behavioral indicators capture reviewing-related behaviors of a user, such as the way a user gives review ratings, the timing of review postings, and etc. Collusive behavioral indicators focus on the collective behaviors of multiple users, such as the count of common items a certain number of users review together. For the spotted anomalies, they are presented together with the evaluation of each indicator. Cumulative histograms (CHs) are used to show the two distributions of the indicator scores over colluders and non-colluders for all the indicators, which provides a visual intuition for how well each indicator discriminates colluders from non-colluders. The larger the gaps between the curves of the two distributions are, the better the discrimination capabilities of the corresponding indicators are. Note that although in practice indicators may not be used individually, our individual evaluation can give a more fine-grained perspective of how the performance of each indicator would be affected by the changes of the collusion patterns in the data, which is helpful to further investigation and assessment of potential correlations among the indicators in practice. Numerically, all the indicator scores are normalized to $[0,1]$.

3.2.1 Linguistic Indicators

Colluders are considered to express opinions differently from regular users. We evaluate 10 linguistic indicators (LIs) proposed in [42, 24, 30] on fake review detection. For the text of a review, the following indicators are investigated:

- Uni-grams (after Chinese word segmentation);
- POS (part-of-speech) tags frequencies;
- Positive sentiment word frequency (PWF);
- Negative sentiment word frequency (NWF);
- Subjective word frequency (SWF);
- Cosine similarity of the review and product descriptions (SRD);
- Brand name mentioned frequency (BNF);
- First person word frequency (FPF);
- Second person word frequency (SPF);
- Squared average length (SAL).

Note that reviews in Chinese characters are different from those in English words, where the text is written without any spaces between words. To deal with this case, *word segmentation* is used to split Chinese sentences into a sequence of meaningful words. We also remove stop-words and punctuations from the review text, and obtain 100 uni-grams by using feature selection metric χ_2 . POS tagging is performed based on the uni-grams. Positive/Negative sentiment words and subjective words are identified by HowNet³. For

³http://www.keenage.com/zhiwang/e_zhiwang.html

each user we aggregate all his/her reviews into one collection and represent it with these linguistic indicators. The CHs of each indicator (exclude Uni-grams and POS) are shown in Figure 3.3.

Figure 3.3: Cumulative distributions of colluders (solid/red) and non-colluders (dashed/blue) vs. indicator scores.

As shown in the figure, none of the cumulative histograms of the examined linguistic indicators exhibit very clear gaps between the distribution curves of colluders and non-colluders. This gives us a hint that nowadays colluders generally express in a similar way with ordinary users, or we can say that they behave *normally* in linguistic level. In fact, this is not that surprising because writing reviews differently (e.g., more passionate or exaggerated) would not bring any benefit. On one hand, website moderators may find these reviews suspicious. On the other hand, readers may find a review untrustworthy if it contains too many bombastic words, then choose to ignore it.

It is worth noting that in the plot of Brand name mentioned frequency (BNF), the curve of non-colluders lies closer to the left axis, suggesting that non-colluders mention brand names more often than colluders. Jindal and Liu [24] consider the reviews that *only* comment on the brands as fake because they believe this kind of reviews is often

biased. While in our case, we find it very likely that regular users often mention brand names just for reference. Not a few users buy products based greatly on the reputations of specific brands; it is thus natural to comment on brands or just use brands as pronouns. In contrast, colluders would craft reviews in a more general fashion, they may even not mention product names in their reviews.

For Squared average length (SAL), we can see that non-colluders generally post longer reviews than colluders. In our further investigation, many non-colluders are identified as very active users who work hard on their comments. The average squared length of the reviews written by colluders is 4.03, lower than the first quartile of that over all reviews in our dataset, which is 4.12. Colluders often write reviews with moderate length probably for effort-saving and the desire to quickly move on to subsequent tasks to pursue more profits.

3.2.2 Individual Behavioral Indicators

A set of individual behavioral indicators (IBIs) are used in [31, 30] which are designed to capture reviewing-related behaviors of a user. Due to heterogeneous data attribute schema, only those applicable to our dataset are selected:

- Review Count (RC);
- Brand Deviation Score (BDS);
- Rating Deviation Score (RDS).
- Targeting Products (TP);
- Targeting Product Groups (TPG);
- General Deviation (GD);
- Early Deviation (ED).

For each user in the dataset, brand deviation score (BDS) measures the deviation in the counts of his/her reviews over different brands, while rating deviation score (RDS) measures the variance of his/her ratings given over different brands. Targeting products (TP) evaluates on a particular product how similar all his/her reviews are in terms of ratings and contents. Targeting product groups (TPG) measures the pattern of ratings towards a set of products sharing common attributes (e.g., brand) within a short time interval. General deviation (GD) measures the average difference between the user rating on one product versus the product’s average rating, and Early deviation (ED) measures how early the ratings of a product deviate from the average rating of that product. Among these IBIs, targeting products (TP) is meaningless in our case as only a very small portion of users are found to post multiple reviews for the same products. The CHs of the remaining IBIs are shown in Figure 3.4.

Figure 3.4: Cumulative distributions of colluders (solid) and non-colluders (dashed) vs. indicator scores.

In the figure, we can see that targeting product groups (TPG), brand deviation score

(BDS) and rating deviation score (RDS) exhibit larger gaps than other three. Specifically, colluders have a much higher TPG score than non-colluders at any cumulative percentage, indicating that colluders tend to review products of the same brand within a short time period (e.g., 1 day). Also, the colluder curves for BDS and RDS lie closer to the left axis, meaning that colluders exhibit less variations in the number of brands and ratings given to each brand. We find that in our dataset, colluders seem to target a limited number of brands, and rate all the products under the same brand similarly (e.g., consistently high ratings of 4 or 5 stars). Despite the fairly clear separation in general deviation (GD), the GD scores of colluders are generally lower. This might indicate that ratings given by colluders are more consistent with the product average ratings, which contradicts the expectation that colluders are the ones give outlier ratings. Such a consistency is possible if the compromised products are overwhelmed by a number of colluders who give similar ratings together. We also find that most of the compromised products are the less popular ones, thus ratings from a small portion of genuine users would have little impact on the average ratings. For the plot of early deviation (ED), we can hardly see separate curves. In fact, as will be confirmed by group early time frame (GETF) in the next section, colluders typically post reviews early. ED performs poorly because the rating deviation of colluders is marginal (with low GD scores). Thus the low rating deviation scores would effectively neutralize the weights associated with the early reviews of colluders, resulting in similar ED scores with non-colluders. Finally, in terms of review count (RC), we see from the plot that the numbers of reviews posted by colluders are generally less than those of non-colluders. Colluders seem not to post too many reviews, probably because they are capable of dispersing the reviews over multiple accounts they control.

3.2.3 Collusive Behavioral Indicators

Since colluders collaboratively post fake reviews on many different products, their collusive behaviors may sell them out. The collusive behavioral indicators are intended to capture such collective patterns of colluders. In this section, we assess all eight collusive behavioral indicators (CBIs) from [37] which is a pioneer work on collusive opinion fraud detection:

- Group Time Window (GTW);
- Group Deviation (GD);
- Group Content Similarity (GCS);
- Group Member Content Similarity (GMCS);
- Group Early Time Frame (GETF);
- Group Size Ratio (GSR);
- Group Size (GS);
- Group Support Count (GSUP).

Among these indicators, group time window (GTW) evaluates how close, in time, the members of a group write reviews for a particular product. Group deviation (GD) flags a group as suspicious if its ratings diverge significantly from other users on a particular product. Group content similarity (GCS) and group member content similarity (GMCS) calculate the maximum average cosine similarities among inter-member reviews and among individual member's reviews, respectively. Group early time frame (GETF) reveals how early a group posts reviews for common products. Group size ratio (GSR) measures, over the common products reviewed by a group, the averaged ratio of the

number of members in the group (i.e., group size) to the actual number of users on the product. Group support count (GSUP) records the number of common products reviewed by a group.

It worth noting that the selected indicators above are group-based in the original study while in our case the indicators should act on each user. To convert a group-based indicator to a user-based version, we first generate the group-based indicator scores for each group (see Section 3.1.1), then each member of the group will inherit the group indicator score. For those who are involved in multiple groups, the highest score over the groups will be chosen, as we intend to capture the worst collusive behaviors of colluders.

Figure 3.5 shows the cumulative distributions of the CBIs.

Figure 3.5: Cumulative distributions of colluders (solid) and non-colluders (dashed) vs. indicator scores.

As shown in the figure, we can see that group time window (GTW), group early time frame (GETF), and group size ratio (GSR) perform quite well in our case. The largest margin is observed in the plot of GTW, which was also spotted in [37], verifying that in reality the duration of a fake opinion activity would not be too long. GETF also performs well, which reveals the fact that colluders typically post fake reviews early so

as to make bigger impact by dominating the sentiments towards the products when the numbers of reviews are not significant. The plot of GS shows a clear gap when cumulative percentage ≥ 0.3 . It is not surprising that the sizes of colluder groups are typically larger. However, we also notice the overlap appearing at the beginning of both curves in GS, which indicates that quite a few colluder groups have similarly small sizes with regular ones that may form by random users. GSR performs fine as most compromised products are also the less popular ones having fewer users, making the group size ratio become larger. The plots of group content similarity (GCS) and group member content similarity (GMCS) show interesting results about the review contents. In GCS, neither the result of colluders nor non-colluders is more statistically significant than each other. In GMCS even colluders achieve lower scores than non-colluders, implying that nowadays colluders tend not to copy from neither their own prior reviews nor those of other accomplices. In fact, we find in the wild that the instructions of many fraud campaigns explicitly state that copying is not allowed during the entire campaigns, so we speculate that colluders may not get paid if they simply cut-and-paste prior reviews. For the case of group deviation (GD), it fails for the same reason as General Deviation evaluated in Section 3.2.2. Finally, group support count (GSUP) performs the worst, implying that colluders are not likely to work together on too many common products in our dataset.

3.2.4 Indicators Comparison and Discussion

In this section, we conduct comparison analysis on the classification ability of the evaluated indicators above. The results are shown in Figure 3.6. We obtain the AUC score for each indicator using a Logistic Regression classifier. We can see that the relative AUC ranking is roughly consistent with the results shown in the cumulative histograms. Linguistic indicators (LIs) generally perform poorly in this classification setting, which was also observed in [37]. In terms of the combinations, IBI+CBI performs the best which

Figure 3.6: Areas under ROC curves (AUC) for all the 24 evaluated indicators (exclude TP) and their combinations. Larger AUC score denotes better detection trade-off between true and false positives based on the corresponding indicator. For name disambiguity, GD(I) as General Deviation, GD(C) as Group Deviation.

is in fact the combination of all behavioral indicators. For colluders, it seems easier to comment just as regular users, however, it is much harder to fake or even change the fraud actions which are mostly driven by the instructions of the fraud campaigns. In the following, we investigate further why those eight behavioral indicators (i.e., General Deviation, ED, RC, Group Deviation, GCS, GMCS, GS, and GSUP) perform poorly.

In fact, when computing CBI scores for each candidate group in Section 3.2.3, we observe that 63.4% of colluders belong to more than one group while only 25.5% non-colluders meet this criteria. For illustration, we randomly pick a colluder who is the member of 13 groups and make the following observations about the groups:

- (o1) *Low group support count (GSUP)*: 12 out of the 13 (92.3%) have only 3 common products while the remaining one has 4;
- (o2) *Low group size (GS)*: five have size 2, three have size 3, four have size 4, and one has size 5;
- (o3) *Low brand variations*: all 13 groups target products of one particular brand;
- (o4) *Similar ratings*: among all 124 product-rating pairs, 21 (16.9%) are 4 stars, and 103 (83.1%) are 5 stars. There are no ratings lesser than 4 stars;

- (o5) *Overlapping colluders*: among all 78 group-group pairs, 69 (88.5%) have one common member, 8 have two, and one pair even has three common members;
- (o6) *Overlapping products*: among all 78 group-group pairs, 57 (73.1%) have at least one common product. Note that these groups have very low group support count (GSUP).

To better understand how these tiny groups cooperate with each other, we draw a bipartite graph to represent the relationship among groups, colluders and products. Figure 3.7 shows an example, in which four groups of colluders share a common member “Z4”. Groups C_1 and C_2 both review products “1C” and “88” while groups C_3 and C_4 both review products “7Y” and “1C”. Such a stealthy arrangement with partially overlapping group members and target sets is sufficient to evade the capture of the evaluated behavioral indicators:

Figure 3.7: 4 groups (dashed boxes) of colluders (circles) collaboratively review six products (squares).

- Groups C_1 and C_2 (likewise C_3 and C_4) would have been merged into one bigger group if both groups post reviews for at least three common products (see Section 3.1.1 for the parameter settings of FIM algorithms). Now that they are single groups, group size (GS) will fail to distinguish them from normal ones that are formed by random users;
- The group support count (GSUP) of these groups are also in the same scale as normal groups in which the members may coincidentally review a few common products due to common preferences;
- From (o4) we know that groups that target products of the same brand would give consistently high/low ratings. If a less popular product is overwhelmed by many such groups, the majority rule followed by early deviation (ED), general deviation (GD(I)) and group deviation (GD(C)) would become invalid because the ratings given by colluders will be closer to the average ratings of the compromised products.

3.3 The KNN-based Method

The above analysis shows that the reliability of many existing indicators can be questioned when used to counter more sophisticated colluders. During the continuous combats against anti-fraud systems, colluders can learn to change their behaviors and eventually adapt to evade existing detectors. In this section, we attempt to solve this problem from a new perspective which can bypass the potential defects of ineffective indicators.

Based on the observations in Section 3.2.4, suspicious groups are found to be highly similar with each other, in terms of overlapped group members and targeted products, and similar ratings. This is essentially a side-effect of applying the frequent itemset mining algorithm. Given a dense reviewer-product bipartite graph, by using the frequent itemset mining (FIM) algorithm with fixed parameter settings, the entire graph would

be decomposed into many small sub-graphs (i.e., groups) that may have overlapped nodes (i.e., members or products) (see Figure 3.7 as an example). Such small sub-graphs would eventually dilute the effectiveness of some of the existing indicators discussed in Section 3.2.4. However, if used properly, these tiny groups can become favorable to detecting colluders in a novel fashion. To this end, we propose a k -nearest neighbor-based method, named FRAUDKNN, to detect colluders, by utilizing the *similarities* between such tiny groups.

Let $\{c_j\}$ be a set of groups and $\{u_i\}$ be a set of users with each associated with a feature vector $\mathbf{f}_i$. By modeling the colluder detection problem as a binary classification problem, our goal is to assign each user u_i with a class label $l_i \in \{pos, neg\}$ ⁴. The idea of FRAUDKNN is that, given a set of groups, the users who are involved in “similar” groups may be more likely to have the same class labels. In other words, the class label of a particular user u_i can be determined collectively (e.g., by voting) by a set of k users who belong to the groups that are the most “similar” to the groups u_i belongs to. The similarity here essentially encodes our belief about the connections between the groups with overlapped colluders. In particular, in line with the six observations made in Section 3.2.4, we design three similarity measures to identify similar groups.

- **Common Member Ratio:** The 5th observation in section 3.2.4 reveals that most of the groups share at least one common members with each other. As discussed, such an overlap between group members is largely caused by the side-effect of applying frequent itemset mining algorithms. To capture this pattern, we propose the Common Member Ratio that measures the *Jaccard similarity* between the set of members of two groups. The range of this measure is between 0 and 1, with a larger value indicating more similar. The formulation is shown as below:

$$s_{\text{cmr}}(j, j') = \frac{\#(M_j \cap M_{j'})}{\#(M_j \cup M_{j'})} \quad (\text{Eq. 3.1})$$

⁴Users labeled as colluders are positive instances.

where M_j and $M_{j'}$ are the sets of members of groups c_j and $c_{j'}$.

- **Common Product Ratio:** Another similar side-effect of applying frequent item-set mining algorithms is that the targeted products of two groups can also overlap, as described by the 6th observation in Section 3.2.4. In order to take this observation into account, we develop the Common Product Ratio measure, which evaluates to what extent two groups review common products together. Specifically, it computes the ratio of the sum of the numbers of products under the same brands targeted by both groups to the sum of the numbers of products targeted by each group separately:

$$s_{\text{cpr}}(j, j') = \max_{b \in B} \frac{\#(P_{b,j}) + \#(P_{b,j'})}{\#(P_j) + \#(P_{j'})} \quad (\text{Eq. 3.2})$$

where B is the set of brands reviewed by both groups c_j and $c_{j'}$. $P_{b,j}$ ($P_{b,j'}$) is the set of products under the brand b reviewed by group c_j ($c_{j'}$), and P_j ($P_{j'}$) is the set of products reviewed by group c_j ($c_{j'}$). The *max* function is used to capture the most severe behaviors of two colluder groups.

- **Common Brand Rating Deviation:** Finally, to capture the similarity between the ratings given by the overlapped colluder groups, as uncovered by the 4th observation in Section 3.2.4, we propose the Common Brand Rating Deviation to measure the rating deviation between two groups. In particular, it computes the average ratings given to the products under the same brands targeted by two groups. It is formulated based on the normalized inverted Root Mean Square Deviation (RMSD):

$$s_{\text{cb rd}}(j, j') = \frac{1}{1 + \sqrt{\frac{1}{|B|} \sum_{b \in B} (\bar{r}_{b,j} - \bar{r}_{b,j'})^2}} \quad (\text{Eq. 3.3})$$

where $\bar{r}_{b,j}$ ($\bar{r}_{b,j'}$) is the average rating given to the products under the brand b by group c_j ($c_{j'}$).

Then, given the pairwise similarity measures above, the overall pairwise similarity between two groups can be computed as the weighted average over all the above measurements:

$$s_{C_j, j'} = \frac{\sum w_k s_k(j, j')}{\sum w_k} \quad (\text{Eq. 3.4})$$

where $s_k \in \{s_{cmr}, s_{cpr}, s_{cbrd}\}$ and w_k is the non negative weight for s_k such that $\sum w_k = 1$.

Having defined the pairwise similarity between two groups, the pairwise similarity between two users u_i and $u_{i'}$ can then be obtained by taking the average over the similarities between all pairs of their respective groups as below:

$$su_{i, i'} = \frac{\sum_{k \in C_i} \sum_{l \in C_{i'}} s_{k, l}}{|C_i| \cdot |C_{i'}|} \quad (\text{Eq. 3.5})$$

where C_i and $C_{i'}$ are the set of groups that contain user u_i and $u_{i'}$ respectively.

Algorithm 1: FraudKNN

Input : training data: D ; test data: T ; the number of neighbors k

Output: the class label for each user u in T

```

1 foreach  $u_i \in T$  do
2   compute  $su_{i, i'}$  with each instance  $u_{i'} = (\mathbf{f}_{i'}, l_{i'}) \in D$ ;
3   choose  $k$  instances  $u_{i''} \in D$  with the highest non-zero  $su$  and add to  $D_k$ ;
4   if  $|D_k| < k$  then
5     | add  $KNN_{\mathbf{f}}(D, \mathbf{f}_j, k - |D_k|)$  to  $D_k$ ;
6    $l_i \leftarrow \arg \max_{l \in \{pos, neg\}} \sum_{u_{i''} \in D_k} I(l = l_{i''})$ ;
7 return  $\{l_i\}$ ;
```

We finally present the design of FRAUDKNN for colluder detection in Algorithm 1. The k nearest neighbors of user u_i are selected according to the pairwise similarity score computed with each user $u_{i'}$ in training data D (lines 1-4). However, if there are not enough users ($\leq k$) in D to achieve $su_{i, i'} > 0$, the vacancies will be filled by performing the traditional $KNN_{\mathbf{f}}(\cdot)$ algorithm that uses the raw feature vectors $\mathbf{f}$ (lines 5-6). Finally, the class label of user u_i is assigned as the one that covers most of the users in D_k , wherein $I(\cdot)$ is the identity function that takes value 1 if $l = l_{i''}$ and 0 otherwise (line 7).

3.4 The Graph-based Classification Method

Although shown to be heuristic, FRAUDKNN has made the first attempt to exploit the relational information derived from user groups to conduct detection. Unlike the review contents or reviewing behaviors, the group structures of colluders are much harder to fake because they have to review the assigned target products in order to make profits. Once a set of colluders have reviewed multiple common products together, they will be merged into a single group. Moreover, FRAUDKNN explores the correlations among colluders by measuring their similarities. This is intuitive since colluders do not work alone. They are well-organized, thus they must be correlated. However, one problem of FRAUDKNN is that it may rely too much on the group structures for detection. The major limitation is that in practice the algorithmic parameters for generating candidate group based on frequent itemset mining (FIM) would be difficult to choose when splitting users into groups. If being tightly set, false negatives would increase otherwise false positives would grow up. In addition, the group-level relational information can be too sparse to be utilized in some datasets, such that FRAUDKNN may degenerate into a traditional KNN classifier that considers only individual-wise features, which would be less robust in practice.

Therefore, more general and flexible approaches are needed for colluder detection which can also push forward the explorations made by FRAUDKNN in terms of utilization of relational information. We thus present our second method to detect colluders. The motivation behind is based on the observation that there is another type of information that may also be difficult to deceive once fraud practices have taken place, which is the *transaction correlations* among users. Given two users and their transaction histories, their transaction correlation forms once they have both reviewed at least one product within a predefined time window. We observe that colluders are more likely to review the same products with other colluders within the period of the fraud campaigns they

are involved. Then if we link two users with an edge once they have been found to be *transactionally correlated* with each other within a time window Δt , it would turn out that colluders would be more likely to appear as the neighbors of other colluders than non-colluders. *Mapping to the classification setting, we say that interlinked users are more likely to have the same class labels than remote ones.* Therefore, the determination of a user's class label can be influenced not only by his/her own attributes, but also by the class labels of the neighborhood. Then the class labels of interconnected users can be inferred collectively during classification, which is the very idea of *collective classification* [40, 47] that attempts to jointly classify instances which are implicitly or explicitly interrelated. To this end, in order to conduct classification collectively, we need a graph model to represent the interconnections among users and a framework for the class label inference.

3.4.1 Colluder Random Field (CoRF)

In this section, we study the transactional correlations between colluders using probabilistic graphical models. In particular, we propose *colluder random field* (CoRF), a probabilistic model built based on *Markov random field* (MRF) [51]. Like MRF, CoRF defines a graph of random variables to model the entities in our context, i.e., the class labels of users and the attributes of users. It also specifies the edges between these random variables, which convey the compatibility of the values the connected random variables take. Markov property is applied in our model by assuming that the class labels of users can be inferred from their attributes and the class labels of their *direct* neighbors.

Formally, all the users and their attributes in our model are formed into a *colluder graph* $\mathbf{CG} \triangleq \{\mathcal{L} \cup \mathcal{F}, \mathcal{E}, \Delta t\}$. Specifically, the (random variable) node set $\mathcal{L} = \{L_i\}_{i=1}^M$ is the set of class labels to be assigned to each user u_i . Their values $\{l_i | l_i \in \{pos, neg\}\}_{i=1}^M$ are *unobserved* and need to be inferred. The (random variable) node set $\mathcal{F} = \{\mathbf{F}_i\}_{i=1}^M$

is the set of *observed* attribute vectors associated with the users. These can be the features/indicators describing the suspiciousness of user reviewing behaviors (as introduced in Section 3.2). The dimension of the attribute vector is set to be D , and we have $\mathbf{F}_i = (F_{i1}, \dots, F_{id}, \dots, F_{iD})$. We denote by $\mathbf{f}_i = (f_{i1}, \dots, f_{id}, \dots, f_{iD})$ the value of $\mathbf{F}_i$. $\mathcal{E}$ is the set of edges connecting related random variables in the colluder graph. In particular, there are two types of edges. The first is used to connect the class label nodes. We have $(L_i, L_{i'}) \in \mathcal{E}$ if a pair of users u_i and $u_{i'}$ have *together* reviewed κ ($\kappa \geq 1$) product(s) within the time window Δt . The second type of edges connect the attributes and the class labels of the same users. In this case, we have $(L_i, F_{ik}) \in \mathcal{E}$ in which L_i and F_{ik} respectively correspond to the class label and *one* dimension of the attribute vector of the same user u_i .

Given the definitions and notations above, our goal of colluder detection is to assign each user u_i with an appropriate class label $l_i \in \{pos, neg\}$. By modeling with Markov random field, our colluder graph $\mathbf{CG}$ is associated with a global probability distribution (logarithm likelihood) as below:

$$\log(Pr(\mathbf{l}|\mathbf{CG})) = \sum_{L_i \in \mathcal{L}} \log(\phi_i(l_i)) + \sum_{(L_i, L_{i'}) \in \mathcal{E}} \log(\psi_{ii'}(l_i, l_{i'})) - \log Z \quad (\text{Eq. 3.6})$$

$$\phi_i(l_i) = \psi_i(l_i) \prod_{(L_i, F_{id}) \in \mathcal{E}} \psi_{id}(l_i) \quad (\text{Eq. 3.7})$$

where $\psi_i(l_i)$, $\psi_{id}(l_i)$, $\psi_{ii'}(l_i, l_{i'})$ are the *potential functions* defined over the three types of cliques $L_i \in \mathcal{L}$, $(L_i, F_{id}) \in \mathcal{E}$, and $(L_i, L_{i'}) \in \mathcal{E}$ in the colluder graph respectively. Z is the regularization factor.

In Eq. 3.6, ϕ_i captures the conditional distribution of the class label L_i of user u_i given his/her observed attribute vector $\mathbf{F}_i$. It can be obtained through the computation of the distribution over l_i given the attribute $\mathbf{f}_i$; while $\psi_{ii'}$ is used to capture the collusion between any pair of users who have reviewed common products (i.e., transactional correlated), and should involve the relational information between users u_i and

$u_{i'}$. At inference phase, this also allows the class labels of adjacent users to affect the final classification result. The definition of the potential functions will be presented in the next section, together with the presentation of the class label inference algorithm. Given Eq. 3.6, our goal is to find the appropriate configuration of the class labels $\hat{\mathbf{l}}$ for all users such that the following objective function is maximized:

$$\hat{\mathbf{l}} = \arg \max_{\mathbf{l}} \log(Pr(\mathbf{l}|CG)) \quad (\text{Eq. 3.8})$$

3.4.2 Collective Inference Algorithm

It is feasible to perform exact inference for a given Markov network if it has special structures such as trees. However, in our case, a colluder graph is typically comprised of thousands of nodes and loops; it thus becomes impossible to apply exact inference to the optimization function (Eq. 3.8). Approximate inference algorithms are needed to tackle this issue. Previously, [46] and [35] have evaluated different approximate inference algorithms on synthetic and real world data respectively, and found the Iterative Classification Algorithm (ICA) to be more reliable. As such, we will base our design of the collective inference algorithm on ICA [40] to infer the probable class labels for each user.

In the following, we define the potential functions used in Eq. 3.6. First, Eq. 3.9 substantiates the potential functions ϕ_i used to associate the class labels with the attribute vectors of the same users. Furthermore, given the fact that the class labels are unobserved while the attribute vectors are known, this association can be interpreted as a conditional distribution of the class labels given the attribute vectors, which can be formalized as below,

$$\phi_i(l_i) = \psi_i(l_i) \prod_{(L_i, F_{id}) \in \mathcal{E}} \psi_{id}(l_i) = Pr(l_i|\mathbf{f}_i) \quad (\text{Eq. 3.9})$$

where $\phi_i(l_i)$ is computed as the probability $Pr(l_i|\mathbf{f}_i)$ of assigning user u_i with the class label l_i given his/her attribute vector $\mathbf{f}_i$. The probability distribution $Pr(l_i|\mathbf{f}_i)$ can further

be obtained by using any classification algorithm that can output distributions over classes, such as Naive Bayes and Logistic Regression.

Second, we define the potential functions $\psi_{ii'}$ for capturing the relationships between the transactional correlated users, as shown in the following equation,

$$\psi_{ii'}(l_i, l_{i'}) = cr_{i,i'} \cdot cd_{i,i'} \cdot I(l_i = l_{i'}) \quad (\text{Eq. 3.10})$$

where $I(\cdot)$ is the identity function. $cr_{i,i'}$ and $cd_{i,i'}$ are the *collusion scoring functions* that capture different aspects of collusion. In particular, $cr_{i,i'}$ calculates the *collusion rate* of a pair of users u_i and $u_{i'}$ to capture their collaboration frequency during the time period Δt , and $cd_{i,i'}$ calculates the *collusion degree* of u_i and $u_{i'}$ to capture their collaboration intensity during the time period Δt . They are formalized as below,

$$cr_{i,i'} = \frac{\#(g_i \cap g_{i'})}{\#(g_i \cup g_{i'})} \quad (\text{Eq. 3.11})$$

$$cd_{i,i'} = \frac{1}{1 + \sqrt{\frac{1}{n} \sum_{k=1}^n (r_{ik} - r_{i'k})^2}} \quad (\text{Eq. 3.12})$$

where in Eq. 3.11, g_i ($g_{i'}$) denotes the products reviewed by u_i ($u_{i'}$) within Δt . $\#(g_i \cap g_{i'})$ denotes the number of products reviewed by both u_i and $u_{i'}$ within Δt , and $\#(g_i \cup g_{i'})$ is the number of products reviewed by either u_i or $u_{i'}$ within Δt . $cr_{i,i'}$ equals to 1 if both of the users have exactly reviewed the same set of products. The intuition behind is that the more common products two users review within a certain time period, the more likely they would be to be colluders with each other. In Eq. 3.12, r_{ik} ($r_{i'k}$) denotes the rating given to product k by u_i ($u_{i'}$). Thus $cd_{i,i'}$ is the normalized inverted RMSD over all pairs of ratings given to the n common products reviewed by both u_i and $u_{i'}$. $cd_{i,i'}$ equals to 1 if both u_i and $u_{i'}$ give exactly the same ratings to their common products. Note that Eq. 3.12 does not consider the rating scale of each user because colluders always choose the same scale that is consistent with the perception of the public (e.g., reviews with 4 or 5 stars are commonly regarded as positive reviews while 1 or 2 stars as negative ones).

Algorithm 2: Collective Inference Algorithm

Input : A colluder graph $\mathbf{CG}$
Output: The class label for each user u_i in $\mathbf{CG}$

```

// Bootstrapping
1 foreach  $L_i \in \mathcal{L}$  do
2    $p_i \leftarrow \max_l \phi_i(l); \quad l_i \leftarrow \arg \max_l \phi_i(l);$ 
// Iteration Classification
3 for  $q = 1$  to  $M$  do
4   foreach  $L_i \in \mathcal{L}$  do
5      $p_i(l_i | \mathbf{CG}, \mathbf{l}) \leftarrow \alpha \exp \{ \sum_{L_{i'} \in N(L_i)} cr_{i,i'} \cdot cd_{i,i'} \cdot p_{i'}(l_i) \cdot \phi_i(l_i) \}$  such that
         $\sum_{l \in \{pos, neg\}} p_i(l | \mathbf{CG}, \mathbf{l}) = 1;$ 
6      $p_i \leftarrow \max_l p_i(l | \mathbf{CG}, \mathbf{l});$ 
7      $l_i \leftarrow \arg \max_l p_i(l | \mathbf{CG}, \mathbf{l});$ 
8    $k \leftarrow (q/M) \times |\mathcal{L}|;$ 
9   Update  $l_i$  with top- $k$   $p_i$ ;
10 return  $\{l_i\};$ 

```

The collective inference algorithm is presented in Algorithm 2. In line 2, l_i is the current most likely assignment of the class label for user u_i , and p_i is the corresponding probability. α is a temporal normalized factor. For each user, the unobserved class labels and the corresponding distributions are initialized based on the attribute-label clique potential functions (Eq. 3.9) in the bootstrapping phase (lines 1-2). In the iteration classification phase, each iteration recomputes the distribution of the class label of each user u_i conditioned on the current class label distributions of the neighborhood $L_{i'} \in N(L_i)$ by using the potential function $\psi_{ii'}(l_i, l_{i'})$ (lines 4-7). At the end of each iteration, the top- k confident class labels are updated where k linearly increases with the iteration times (lines 8-9).

3.5 Evaluational Analysis

In this section, we evaluate the proposed methods for supervised colluder detection in online review portals. We first evaluate the performance of FRAUDKNN in Section 3.5.1. Then, in Section 3.5.2, the performance of the graph-based classification method is assessed. Finally, we conduct comparison analysis between the proposed methods and state-of-the-art colluder detection approaches in Section 3.5.3.

3.5.1 Evaluation of FraudKNN

A set of experiments have been conducted to evaluate FRAUDKNN. In Section 3.5.1.1, we evaluate the effectiveness of the pairwise similarity measures used in FRAUDKNN, and show their characterization ability in capturing the relationships between overlapped groups. Then, we show the detection performance of FRAUDKNN in Section 3.5.1.2.

3.5.1.1 Pairwise Similarity Distribution

As shown in Algorithm 1, the core of FRAUDKNN is to compute the pairwise similarity of a pair of users and choose the k most similar ones for voting. The performance will largely rely on the hypothesis that the pairwise similarity of two users within the same class is higher than that between users with different class labels.

Individual Evaluation

First of all, we separately evaluate the discriminating performance of the proposed pairwise measures for computing the similarity between any pair of groups. In particular, we show the distribution of each similarity measure over different user pairs. As we have formulated colluder detection as a binary classification problem, there are three different user pairs in our case, i.e., positive–positive pair, negative–positive pair, and negative–negative pair⁵. Figure 3.8 draws the cumulative distribution functions (CDFs)

⁵Recall that colluders and non-colluders are treated as positive and negative instances respectively.

of the three pairwise similarity measures proposed in Section 3.3, i.e., Common Member Ratio, Common Product Ratio, and Common Brand Rating Deviation, over the three users pairs. It can be seen that, in all figures, the distributions of the evaluated measures are clearly separable over different types of user pairs. In particular, the similarity values of the positive–positive pairs generally attain higher values than those of the other two pair types. This shows that the colluders are indeed involved in groups sharing more structural commonalities such as group members, targeted products, and provided ratings. In addition, we can also observe that both the pure pairs, i.e., the positive–positive and negative–negative pairs, achieve higher similarity values than the mixed one, i.e., the negative–positive pair. This is a desirable property for FRAUDKNN to successfully classify colluders and non-colluders, since in this way most of the neighbors of a colluder (or a non-colluder) would also have the same class labels, which promotes correct classification.

Figure 3.8: Distributions of the three types of pairwise similarity measures over the three user pairs, i.e., positive-positive pair (in red/dashed line), negative-positive pair (in black/dashed line), and negative-negative pair (in blue/solid line).

Overall Evaluation

Besides individual evaluation, we also examine the overall similarity between two users by combining all the pairwise measures evaluated before. Figure 3.9 shows the distributions of the overall pairwise similarity values over the three types of user pairs.

Figure 3.9: Distributions of the overall pairwise similarity values (sv) over the three user pairs, i.e., positive-positive pair (in red/dashed line), negative-positive pair (in black/dashed line), and negative-negative pair (in blue/solid line).

As before, we can see clear separations between the three CDF curves. In particular, after combination, the positive-positive pairs still attain much higher overall similarity values (sv) than the other two. It is also worth noting that the distribution of sv is quite sparse; over 97.5% of the pairwise similarity values between any user pair in our dataset are zero. This, however, would be problematic when many instances in the dataset do not have enough neighbors with non-zero sv for voting. In this case, FRAUDKNN would degenerate into a traditional KNN classifier that would only consider individuals' attributes for decision making. In our dataset, half of the users have no more than 22 neighbors with non-zero sv (as shown in Figure 3.10). Thus, our method can be seen as an extension of traditional classification approaches which exploits users' group-level relational information to improve the final detection performance.

3.5.1.2 Colluder Detection Performance

Here, we evaluate the classification performance of FRAUDKNN using the standard metrics - precision, recall and F1-score, where precision and recall are the ratios of the

Figure 3.10: Distribution of the number of the neighbors with non-zero sv for each user.

predicted true colluders to the predicted users and true colluders, respectively. In addition, as the classes in our dataset are of quite different sizes, MCC (Matthews Correlation Coefficient) is also included which is a more balanced measure for skewed binary classifications. 10-fold cross-validation is used to create dataset splits for training and testing based on the dataset. The combination of all the evaluated collusion indicators LI+IBI+CBI is used as the attributes $\mathbf{a}$ for the traditional $KNN_{\mathbf{a}}(\cdot)$ algorithm. For simplicity, the weights in Eq. 3.4 are equally set. The results are shown in Figure 3.11.

Figure 3.11: Performance (F1-score and MCC) of FRAUDKNN vs. the number of nearest neighbor k . Error bars show the standard error of the mean.

We can see that FRAUDKNN achieves encouraging results when the k is not set too

large. Specifically, at $k = 5$, the F1-score attains 0.914, which drops accordingly as k becomes larger. This again verifies the fact that as more are chosen from the training set, the ones in the test set may have greater portion of with whom the pairwise similarity scores attain zero. To this end, the traditional KNN algorithm steps in to fill up those vacancies. Due to this problem, all the subsequent experiments will choose small k to reduce the chance of degeneration.

3.5.2 Evaluation of the Graph-based Method

In this section, we evaluate the Colluder Random Field (CoRF) model for colluder detection. In Section 3.5.2.1, we will establish the experimental settings in which a series of colluder graphs are constructed from the raw review data as the input to the collective inference algorithms. Then, the classification performance of CoRF will be shown and discussed. In Section 3.5.2.2, we will discuss the impact of different time window settings to the classification performance of CoRF.

3.5.2.1 Classification Performance

The preparation of the input to the collective inference algorithm proposed in Section 3.4 demands the generation of a set of colluder graphs according to the definition introduced in Section 3.4.1. Recall that an edge is formed between any pair of user class label nodes in a colluder graph if and only if both users have reviewed κ ($\kappa \geq 1$) common product(s) within a time window Δt . In our experiments, we set κ to 1 so as to capture all potential edges among the users. The time window Δt is set from 7 to 119 days with an interval of one week long. For each parameter setting, the largest connected component of the resulting graph is taken as an instance of our colluder graphs. The statistics of the resulting colluder graphs are shown in Table 3.2.

For the training data preparation, as Algorithm 2 is fed with networked data, k -fold cross-validation like methods may not be suitable for our case, since splitting the dataset

	Size	#(pos)	#(neg)	#(neg-neg)	#(neg-pos)
$\Delta t = 7$	4496	1894	2602	6416	1473
$\Delta t = 21$	4869	1924	2945	15618	2989
$\Delta t = 35$	4931	1928	3003	24066	4348
$\Delta t = 49$	4963	1932	3031	31848	5537
$\Delta t = 63$	4978	1932	3046	39179	6674
$\Delta t = 77$	4997	1935	3062	46165	7595
$\Delta t = 91$	5010	1935	3075	52587	8571
$\Delta t = 105$	5020	1935	3085	58951	9412
$\Delta t = 119$	5023	1935	3088	65195	10176
	#(pos-pos)	%(neg-neg)	%(neg-pos)	%(pos-pos)	Density
$\Delta t = 7$	22420	0.212	0.049	0.740	0.0030
$\Delta t = 21$	40298	0.265	0.051	0.684	0.0050
$\Delta t = 35$	53815	0.293	0.053	0.654	0.0068
$\Delta t = 49$	64190	0.313	0.055	0.632	0.0082
$\Delta t = 63$	73462	0.328	0.056	0.616	0.0096
$\Delta t = 77$	82032	0.339	0.056	0.604	0.0108
$\Delta t = 91$	89575	0.349	0.057	0.594	0.0120
$\Delta t = 105$	96107	0.358	0.057	0.584	0.0131
$\Delta t = 119$	101848	0.368	0.057	0.575	0.0141

Table 3.2: Statistics of colluder graphs with different time window settings. The statistics of the colluder graphs corresponding to $\Delta t = 14, 28, 42, 56, 70, 84, 98, 112$ are not shown due to the space constraints.

randomly into k subsets and using $k - 1$ of them for training will lead to an expectation of $\frac{k-1}{k}$ of neighbors of a test node being labeled, which can bias the classification results. Instead, we use a breadth-first search (BFS) based strategy in which we construct splits for test data by randomly selecting the starting node and expanding around it till a predefined sample size has been reached⁶. The selected set of nodes Te are used for testing and the rest, denoted as Tr , are used for training. Note that, evaluation metrics are measured *only* on the subset of Te that has no neighbors in Tr . In our experiments, for a given colluder graph, we repeat this process multiple times and obtain 10 test-train pairs of splits where the class distribution of each test folder is close to that of the entire dataset (i.e., near stratified). The resulting collection of splits for a given

⁶Empirically, we set the predefined sample size as half of the size of the corresponding colluder graph.

colluder graph with time window Δt is denoted as $BS_{\Delta t}$. For the local classifier, SVM is used in the bootstrapping phase. The combination of all evaluated indicators, i.e., linguistic indicators (LI) + individual behavioral indicators (IBI) + collective behavioral indicators (CBI), serves as the attribute set. The iteration number in Algorithm 2 is set to 10. Figure 3.12 shows the classification performance of CoRF in F1-score and MCC with different time window settings.

Figure 3.12: Classification performance (F1-score and MCC) of the graph-based classification method with different time windows. Error bars show the standard error of the mean.

From this figure, it can be observed that CoRF achieves promising results in a steady manner as the colluder graph expands incrementally. This is not surprising because at the beginning when $\Delta t = 7$ the colluder graph has already possessed very high homophily: $\frac{\#(pos-pos)}{\#(pos-pos) + \#(neg-pos)} = 93.8\%$ (from Table 3.2) of the colluder-induced edges connect colluders at both ends, suggesting that a colluder will have a chance of 93.8% to choose another colluder as her neighbors; similarly, a non-colluder will have a chance of 81.3% to choose another non-colluder as as her neighbors. This can greatly benefit the class label propagation in the collective inference algorithm, since class labels are updated through seeking the neighborhood for consultant.

3.5.2.2 The Effects of Time Window for Colluder Graph Generation

The time window used to form the edges in a colluder graph is crucial for the overall classification performance. In this section, we discuss the impact of different time window settings to the detection performance. As we can see in Table 3.2, as the time window increases, the numbers of neg-neg and neg-pos edges added to the colluder graph also increase. When the time window reaches 119 days, the portion of the pos-pos edges drops to 57.5% while the portion of the neg-pos edges rises to 5.7%. One may worry that the introduced neg-pos edges could possibly affect the re-labeling of colluders in each iteration, since colluders are expected to have more non-colluder neighbors than before. However, such a case may not happen. The reason is that when determining the class label of a user, besides her neighbors' class labels being considered, the collusion scoring functions (i.e., cr and cd as shown in Eq. 3.11 and Eq. 3.12) can also take effect as the weights of the edges. To show this observation, we further plot the distribution of the weights (i.e., $cr \cdot cd$) of the colluder graph edges over different time window settings, as shown in Figure 3.13.

Figure 3.13: The distribution of the weights ($cr \cdot cd$) of the colluder graph edges vs. different time window settings with $\Delta t = 7, 42, 84, 119$.

We can see that as the time window becomes larger, the boxes of neg-pos edges and pos-pos edges shift downwards due to the addition of many new edges with lower $cr \cdot cd$ (also note that the $cr \cdot cd$ of neg-pos edges are significantly lower than that of pos-pos edges all the time). As a result, these newly added neg-pos edges would impose much weaker effects on the determination of the class labels of colluders than the “older” pos-pos edges, thus the overall classification performance would not become worse (see Figure 3.12).

Figure 3.14: Distribution of the time intervals associated with different types of edges in the colluder graph with $\Delta t = +\infty$. The median values are displayed above the corresponding boxes.

The next question then becomes how to set the time window properly to form a “good” colluder graph that is sufficient to catch most of the colluders? We argue that in practice the time window can be set based on domain knowledge from experts or experiences of website mediators. In our case here, we plot the distributions of time intervals associated with different types of edges in the colluder graph with $\Delta t = +\infty$ (Figure 3.14). We can see that the median value (50 days) of the time intervals associated with all the pos-pos edges seems to be a fair lower bound for the time window Δt . With this setting, most of pos-pos edges are retained while not many neg-pos edges are included. In this way, a

relatively high purity of the colluders' neighborhood can be achieved. Also, we can see from Table 3.2 that when $\Delta t = 49$, the corresponding colluder graph has already covered $1932/1937=99.7\%$ colluders.

3.5.3 Performance Comparison

In this section, we compare the proposed two methods (i.e., FRAUDKNN and CoRF) with various baseline approaches. Specifically, we have the following as baselines:

- Text-based spam review detector (SRD). This approach is commonly used to classify a particular review as spam or not, based on the linguistic feature of the review. However, through proper extension, it can also be used in our case for colluder detection. First, we classify each review in our candidate group dataset as spam or non-spam. Then, if one of the reviews posted by a user in our dataset is labeled as spam, the corresponding user is thus classified as a colluder, and a non-colluder otherwise⁷. For feature preparation, the Linguistic Indicator set (LI) evaluated in Section 3.2.1 is used to characterize the review text. For model training, four kinds of traditional classifiers, i.e., Random Forest, Naive Bayes (Multinomial), K-nearest neighbors, and linear SVM, are used separately to train the baseline SRDs.
- K-nearest neighbors (KNN). The traditional KNN classifier is included in our comparison, as it serves as the basis of the proposed FRAUDKNN. For feature preparation, the three indicator sets evaluated in our empirical analysis (Section 3.2) are used. They are Linguistic Indicators (LI), Individual Behavioral Indicators (IBI), Collusive Behavioral Indicators (CBI), Behavioral Indicators (BI, i.e., IBI+CBI), and the combination of all (ALL).

⁷Recall that our goal in this thesis is to detect collusive reputation fraud, and our evaluation datasets only contain colluders and non-colluders.

- Support Vector Machine (SVM). The traditional SVM with linear kernel is used as another baseline classifier. The same five indicator sets are used as features to train five independent baseline models.
- FRAUDKNN. The proposed FRAUDKNN in Section 3.3. As FRAUDKNN would invoke the traditional KNN when the distribution of the pairwise similarity is sparse, the five indicator sets are respectively used to train the traditional KNN models.
- CoRF. The proposed Colluder Random Field in Section 3.4.1. As CoRF needs a local classifier for bootstrapping, the five indicator sets are used again to train five local classifiers respectively.

For KNN and FRAUDKNN, we set $k = 5$. For CoRF, a SVM (with linear kernel) is used as the local classifier in the bootstrapping phase. We set the time window as $\Delta t = 50$ to construct the colluder graph. Thus BS_{50} is used as the evaluation dataset. The results are shown in Figure 3.15.

We have made the following observations. First, while the spam review detection (SRD) approach serves as a traditional baseline in opinion fraud detection literature, we can see that the four SRD baselines, after extension, achieve reasonably-well and comparable results in colluder detection as well, with the SVM model performing the best. Recall that in SRD, the rationale is to first identify spam reviews and then identify the spammers who post at least one spam review. Such a review-centric approach may outperform the user-centric method (e.g., KNN (LI) and SVM (LI)) where the features are derived from all reviews of a user. This can be attributed to the fact that a spammer, who is also an ordinary user, can still post genuine reviews when she is off task. It is thus not appropriate to mix all the reviews of a user together for feature extraction and model building. Instead, each review of a particular user should be treated separately. In

Figure 3.15: Performance comparison between the proposed methods (FRAUDKNN and CoRF) and the three baseline classifiers (SRD, KNN and SVM) on AmazonCn dataset. Our proposed methods improve over the baseline classifiers at the confident level of 95% based on two-tailed t-test.

the experiment results we can see that both the review-centric models (i.e., SRD (KNN) and SRD (SVM)) outperform their user-centric counterparts (i.e., KNN (LI) and SVM (LI)), with the improvements ranging from 14.5% to 18.9% in F1-score.

Second, by comparing methods using the same models but different indicator sets, we can observe the efficacy of different kinds of indicators. Specifically, across all evaluated methods (i.e., KNN, SVM, FRAUDKNN, and CoRF), we can make a consistent observation that the behavioral indicators (BI) used to characterize colluders' collective behaviors are more effective than the linguistic indicators (LI) in colluder detection, with the improvements ranging from 0.8% to 32.2% in F1-score. This suggests that it is much harder for colluders to cover their collective behavioral trails in practice when launching fraud campaigns, since no matter what actions (e.g., strategies, tricks) they take during the campaigns, these actions will together be imposed on the same targeted items, resulting in the exhibition of collective patterns to varying extent.

Finally, it can be seen that both our methods (FRAUDKNN and CoRF) outperform

the baseline classifiers (SRD, KNN, and SVM) on all indicator sets. Specifically, FRAUD-KNN achieves stably well regardless of the indicator sets. The F1-score of FRAUDKNN improves by 8.9% to 43.5%. This is reasonable since when k is set to be small (5 in the experiments), most of the users can find the neighbors based on the group-level pairwise similarity measures who are very likely to have the same class labels with them. In this case, FRAUDKNN will not degenerate into the traditional KNN model which uses indicators only, and the detection results are less sensitive to the specific indicator sets used. CoRF also works well, whose F1-score increases over traditional classification models by 9.6% to 36.2%. By comparing the two proposed methods, we find that neither one is statistically more significant than each other. Comparatively, CoRF slightly suffers from the poorer indicators (e.g., LI and IBI). However, once stronger indicators (e.g., BI and ALL) are utilized for bootstrapping, the performance of CoRF becomes better. This can be ascribed to the fact that when the local classifiers make mistakes locally in the bootstrapping, the errors would propagate to other parts of the network within just a few iterations. By using more powerful indicators such as BI and ALL for local classifier training, CoRF finally achieves the best results in both F1-score (CoRF (BI)) and MCC (CoRF (ALL)).

3.5.4 Generalization Ability

To further verify the generalization ability of the proposed approaches, we conduct further experiments on an additional dataset, which is more recent, and widely used in many review fraud studies. Note that this dataset is publicly available after the substantial work of this chapter has been done.

3.5.4.1 Dataset

The dataset we use for generalization evaluation is named YelpZip and created by the authors in [45]. It contains reviews about restaurants and hotels on Yelp.com. The

original dataset contains annotations about general fake reviews and review spammers (based on the results of Yelp’s own anti-spam system), but does not specify whether a user is a colluder or not. To obtain such information, we generate candidate groups from the original annotated datasets by following the same procedure as before, and obtain 5,980 groups in total. Then, the colluder and non-colluder labeling will proceed over all groups and the members of the groups. As with the case of AmazonCn dataset annotation, a group is labeled as a colluder group if all its members are annotated as spammers in the original dataset. Further, a user is labeled as a colluder if she is involved in any of the groups labeled as colluder group, and a non-colluder otherwise.

The statistics of the YelpZip dataset are summarized in Table 3.3.

Statistics	Value
Number of users (Number of colluders, Number of non-colluders)	3038 (474, 2564)
Number of products	1191
Group size (Min., Mean, Median, Max.)	2, 4.5, 4, 46
Number of products reviewed by a group (Min., Mean, Median, Max.)	3, 5.5, 5, 45
Number of groups a user belongs to (Min., Mean, Median, Max.)	1, 8.8, 2, 838

Table 3.3: Statistic summary of the YelpZip dataset.

We can see that compared to the AmazonCn dataset, the YelpZip dataset is smaller, containing less colluders and non-colluders. Moreover, the ratio of the number of colluders to the number of non-colluders is also less than that in AmazonCn dataset, indicating the reputation collusive fraud is relatively less active in the YelpZip dataset.

In addition, we also compare the differences between the behavioral statistics of colluders and non-colluders. Figure 3.16 examines the sizes of groups colluders or non-colluders are involved in. In particular, it plots the cumulative distributions of the average size of the groups a colluder or a non-colluder belongs to. From the figure, we can observe that, as with the AmazonCn dataset, for both classes, most of the users are involved in groups with small sizes. However, different from the AmazonCn case where colluders

are involved in groups larger than those containing non-colluders, the case of YelpZip is the opposite, with colluders generally involved in smaller groups (over 90% colluders are involved in groups with size 2). This again can suggest that the collusive fraud practice is much milder in YelpZip than in AmazonCn. On the other hand, the fact that non-colluders in YelpZip review more restaurants together might be ascribed to the inherent locality property of offline businesses; it is common that people from the same region are more likely to visit restaurants in the locality, resulting in more synchronized reviewing behaviors.

Figure 3.16: Cumulative distributions of the average size of the groups a colluder (red) or a non-colluder (blue) belongs to on YelpZip.

Finally, Figure 3.17 shows the difference between colluders and non-colluders in the number of reviewed products. We can see that in this dataset, colluders generally post less reviews than non-colluders during the span of their account lifetimes, which is consistent with the case of the AmazonCn dataset. This might imply that in reality colluders do not use their accounts so often as non-colluders do; their accounts might be used for fraud tasks for most of the time.

Figure 3.17: Distributions of the number of products reviews by a colluder (red) or a non-colluder (blue) on YelpZip.

3.5.4.2 Comparison Results

In this section, we show the generalization ability of the proposed approaches by demonstrating their performance in colluder detection on YelpZip dataset. As with the case of the AmazonCn dataset before, the same experimental analysis is conducted here as well to compare the proposed approaches with state-of-the-art baselines on YelpZip dataset. Figure 3.18 shows the results. Several observations can be made as follows.

First, we can see that in general the performance achieved by each compared method is not so remarkable as compared to the results on AmazonCn. This makes sense, since reviewing behaviors (for both colluders and non-colluders) can vary between datasets collected from different real-world scenarios. The results thus suggest that the features used here may not be so effective as used on AmazonCn dataset. But even so, all the compared methods still demonstrate their effectiveness in detecting colluders in YelpZip by achieving promising results, with the average F1-score being 0.751 for SRD, 0.738 for

Figure 3.18: Performance comparison between the proposed methods (FRAUDKNN and CoRF) and the three baseline classifiers (SRD, KNN and SVM) on YelpZip dataset. Our proposed methods improve over the baseline classifiers at the confident level of 95% based on two-tailed t-test.

KNN, 0.758 for SVM, 0.865 for FRAUDKNN, and 0.836 for CoRF.

Second, as with the case of the AmazonCn dataset, we again make the observation that when only textual features are used, the review-centric approach (SRD) performs better than the user-centric approach (i.e., KNN(LI) and SVM(LI)) by an observable margin (1~2%). Recall that SRD treats each review of a user independently while the user-centric approach considers all the reviews of a user as a whole. Such consistent results indicate that in practice colluders can also post normal reviews as the regular users do, and it can be problematic if we treat all the reviews of a user indiscriminately when a textual-based approach is adopted.

Third, for feature comparison, we compare the results from the same models but with different indicator sets used. It can be seen that the collusive behavioral indicators (CBI) are still more effective than the other two, i.e., linguistic and individual behavioral indicators (LI and IBI), implying that the collective behaviors of colluders are more reliably different from those of non-colluders across both datasets.

Finally, we can see that by exploiting relational information between colluders for detection, both of the proposed methods, i.e., FRAUDKNN and CoRF, improve the performance of each of the baselines across all indicator sets. In particular, the improvements of FRAUDKNN over the traditional KNN method range from 7.3% to 20.5% while CoRF improves over SVM by a margin ranging from 7.8% to 12.3%, demonstrating the effectiveness of both methods on YelpZip dataset. When comparing the two proposed methods with each other, we can see that FRAUDKNN dominates the results. This implies that in the YelpZip dataset, the group-level information (i.e., the similarity among the colluder groups) used by FRAUDKNN can reveal stronger relationships between colluder groups than the pairwise interactions between colluders do, which are used in CoRF.

3.6 Summary

In this chapter, we study the state-of-the-art adversarial patterns of collusive reputation fraud in online review portals, and uncover a novel attack vector where multiple tiny colluder groups cooperate with each other on targets that are partially related. We conduct a series of collusion characterization analysis and find that this new collusion strategy causes the failure of 25 state-of-the-art features proposed in previous studies for opinion fraud detection. These features are designed to capture the peculiarities of opinion fraudsters in review language usage patterns, review posting behaviors, and collective reviewing behaviors. To confront this new attack vector, two novel classification algorithms, namely FRAUDKNN and Colluder Random Field (CoRF), are proposed for colluder detection which investigate and utilize the inherent connections between colluder groups or colluders. In FRAUDKNN, three pairwise similarity measures are proposed to define a neighborhood for similar colluder groups. In particular, these measures examine the commonalities between groups in terms of the members, targets, and review sentiments. Then the classification of a user is performed by considering the class labels of

other users who come from similar groups. In CoRF, transaction correlations between users are utilized to associate potential colluders. The formation of transaction correlations occurs once two users review any common product during their life cycles. This enables the proposal of a probabilistic colluder graphical model where the attributes of users and their transaction correlations are jointly utilized for colluder classification. A novel approximate inference algorithm is also developed for collective classification based on the model. Finally, extensive experiments are performed to validate and demonstrate the effectiveness of the proposed methods in colluder classification performance.

The merits of the proposed classification methods for colluder detection are summarized as follows: (1) Different from traditional colluder classifiers that focus on individual objects, both the proposed methods exploit relational information extracted from multiple objects for detection, so as to refine the original classification results. The behavioral correlations between colluders caused by common reviewing traces are considered more robust against manipulation than the pointwise signals derived from individual objects. (2) The pairwise similarity measures proposed in FRAUDKNN are shown to effectively capture the consequent connections between tiny colluder groups working for the same fraud campaigns. (3) Compared to FRAUDKNN, CoRF provides a more principled and coherent way to characterize and utilize transactional correlations between colluders. The identification of behaviorally-related colluders can therefore benefit from each other to achieve further classification refinement.

Chapter 4

FraudInformer: An Unsupervised Framework for Collusive Reputation Fraud Detection

This chapter focuses on addressing the problem of collusive reputation fraud in unsupervised settings. As shown in the previous chapter, supervised models can achieve promising detection accuracy when annotated fake reviews are available for predictive model training. However, the major problem is that such fake review annotation can often be difficult to obtain in practice due to the lack of ground-truth in reality [24, 31, 54]. On the other hand, for existing unsupervised collusive fraud detectors that are designed to capture blatant manipulation behaviors, one problem is that these approaches are vulnerable to stealthier attacks that involve small spam campaigns. Moreover, in these approaches, the detection algorithms are strongly coupled with the proposed specialized features, resulting in the lack of flexibility to incorporate new features for model improving. To address these issues, we propose a novel detection framework FRAUDINFORMER to tackle collusive reputation fraud in unsupervised settings (Section 4.2). To handle attacks by small spam campaigns, a suite of heterogeneous pairwise features are proposed to capture the implicit connections between colluders from the same spam campaigns (Section 4.3). Compared to the group-based features used in previous studies, the proposed pairwise features are more fine-grained to directly reflect the relationships among

colluders. Moreover, a propagation-based detection algorithm is developed to cooperate with the pairwise features which is (a) intuitive; that the detection task is formulated as an autonomous process of mutual disclosure at a virtual trial where colluders are forced to give themselves away, (b) unsupervised; that no prior knowledge of fake review annotation is needed, and (c) flexible; that can incorporate any number of pairwise features into colluder detection. Theoretical analysis is conducted to evaluate the robustness of FRAUDINFORMER in defending against adversarial challenge (Section 4.5). Experimental results are shown in Section 4.4, and Section 5.5 summarizes this chapter.

4.1 Problem Formulation

First of all, we formulate the problem of unsupervised collusive opinion fraud detection. Given a typical review site (e.g., amazon.com) containing a set of products $\mathcal{P} = \{p_k\}$ with each commented by a list ℓ_k of users *in chronological order* as shown in Figure 4.1, the goal of the proposed unsupervised detection framework FRAUDINFORMER is to rank all users $\mathcal{V}$ globally so that top-ranked ones are more likely to be colluders. To produce the resulting rankings, for each user $v \in \mathcal{V}$, we associate a global spam score or we call *spamicity* s , which specifies the suspectedness of his/her engaging in spam campaigns. In particular, the spamicities of colluders are supposed to be higher than those of non-colluders.

4.2 The FraudInformer Framework

In this section, we elaborate the design and implementation of the proposed unsupervised framework for collusive reputation fraud detection.

Figure 4.1: An illustration of chronologically-ordered review lists. It shows four lists of reviewers in chronological order for products P1–P4.

4.2.1 Mutual Disclosure Modeling

The basis of FRAUDINFORMER relies on the idea of *mutual disclosure* inspired by the concept of criminal trials, where a suspect is asked to give his/her accomplices away. Similarly in our context of collusive opinion fraud, colluders participating in the same spam campaigns are also accomplices of each other. Given the behavioral clues collected from colluders’ reviewing activities as the evidence against their collusions, it is possible to expose them all at once by proceeding with proper “criminal trials” against them.

Specifically, we propose the “criminal trial” scheme as follows: for each product p_k reviewed by a list ℓ_k of n_k users in chronological order, each user v_i located at position $\ell_k(i)$ on ℓ_k is asked to accuse other surrounding users on ℓ_k within an *investigating range* ζ . The intuition behind the accusation against surrounding users is based on the temporal locality property of spammers observed in [14], which states that when a product is under opinion spam attack, multiple fake reviews may be posted within certain time periods. Thus, these fake reviews should be close to each other on the chronologically-ordered

review lists. The set of users to be accused by v_i on ℓ_k can be denoted by,

$$R_{k,i}(\zeta) = \{v_j | |\ell_k(i) - \ell_k(j)| \leq \zeta, j \neq i\} \quad (\text{Eq. 4.1})$$

For generality purpose, ζ can be any functional realization of the temporal locality. For example, ζ can be a function of n_k : $\zeta(n_k) = \sqrt{n_k}$. One merit of the presented “criminal trial” scheme is that it avoids the computation of all possible pairs of users in a site in which case most of the pairs can hardly be colluders if they have not co-reviewed any common product; the computational complexity has thus been reduced from $O(|\mathcal{V}|^2)$ to $O(2\zeta|\mathcal{V}|)$.

Apart from the accusation itself, the amount of evidence gathered during the accusation should also be considered and reflected. This is because the evidence gathered from colluder pairs should be stronger than that from non-colluder pairs. Moreover, the more intensively two colluders work with each other, the more evidence of their collusion can be gathered. Here we model the amount of collusion evidence or we call the *collusiveness* gathered from a user pair (v_i, v_j) as a real-valued symmetric function $\Phi(i, j)$. It is worth noting that the collusiveness is crucial to the success of the proposed detection framework as it serves as the only testimony during the entire judgement. To this end, the mutual disclosure based “criminal trial” regarding all users of a site can be computationally formalized as a process of collusiveness propagation. Specifically, the spamicity s_i of user $v_i \in \mathcal{V}$ is computed as the sum of the product of the amount of evidence $\Phi(i, j)$ provided by all surrounding users $v_j \in R_{k,i}(\zeta)$ and their own spamicities, over all products $k \in \mathcal{P}_i$ reviewed by v_i :

$$s_i = \sum_{k \in \mathcal{P}_i} \sum_{v_j \in R_{k,i}(\zeta)} s_j \cdot \Phi(i, j) \quad (\text{Eq. 4.2})$$

During the propagation, the spamicity of a true colluder will be constantly increasing for accumulating more collusiveness from his/her accomplices, while a non-colluder will gain less collusiveness as (s)he is not supposed to collude with neither colluders nor other regular users.

4.2.2 Confidence Weighting

Despite being derived from external collusion facts (i.e., the posted product reviews), the collusion evidence provided by a particular user about others may not be equally convincing internally. For example, a colluder may expect the reviews posted by his/her accomplices to be near his/her own postings due to their synchronous engagements in the same spam campaigns. The confidence in such accusations should thus be higher for the evidence provided about nearby users. In FRAUDINFORMER, this semantic is captured by the confidence function $\Omega_j(i, R)$, measuring to what degree the evidence provided by v_j about $v_i \in R_{\cdot, j}(\zeta)$ should be trusted from the perspective of v_j . We consider two classes of confidence functions here, i.e., symmetric and asymmetric.

Symmetric confidence functions $\Omega_j^\dagger(i, R)$ of v_j assume his/her equally distributed belief in the evidence about nearby users $v_i \in R_{\cdot, j}(\zeta)$. The simplest realization can be based on the uniform kernel function [44] which assigns identical confidence to the evidence about each nearby user:

$$\Omega_j^{\dagger U}(i, R) = \frac{1}{2} \mathbf{I}_{\{v_i \in R_{\cdot, j}(\zeta)\}} \quad (\text{Eq. 4.3})$$

where $\mathbf{I}_{\{\cdot\}}$ is the identity function. A more sophisticated case that encodes the concentration of spam reviews can be based on the Epanechnikov kernel function:

$$\Omega_j^{\dagger E}(i, R) = \frac{3}{4} \left(1 - \left(\frac{|\ell_{\cdot, i} - \ell_{\cdot, j}|}{\zeta(\cdot)} \right)^2 \right) \mathbf{I}_{\{v_i \in R_{\cdot, j}(\zeta)\}} \quad (\text{Eq. 4.4})$$

which posits that evidence about closer users on product review lists matters more than remote ones.

Asymmetric confidence functions $\Omega_j^*(i)$ on the other hand treat the confidence of v_j in each surrounding user differently. An example can be a function that only trusts the evidence about those who have the top K collusiveness $\Phi(i, j)$ with v_j :

$$\Omega_j^{*K}(i, R) = \frac{1}{K} \mathbf{I}_{\{\Phi(i, j) \geq \Phi^{(K)}, v_i \in R_{\cdot, j}(\zeta)\}} \quad (\text{Eq. 4.5})$$

where $\Phi^{(K)}$ is the K th largest collusiveness in the set. The intuition is that stronger evidence itself to some extent implies higher possibility of true collusion and thus should gain higher confidence.

Finally, to account for the impact of confidence weighting, the original expression (Eq. 4.2) for computing the spamicity of user v_i can be modified to:

$$s_i = \sum_{k \in \mathcal{P}_i} \sum_{v_j \in R_{k,i}(\zeta)} s_j \cdot \Phi(i, j) \cdot \Omega_j(i, R_{k,j}(\zeta)) \quad (\text{Eq. 4.6})$$

4.2.3 Global Ranking

To obtain a final ranking of users, a propagation-based ranking algorithm is proposed based on the mutual disclosure model (Eq. 4.6). Recall that at the end of mutual disclosure, a colluder will gain high spamicity for accumulating high collusiveness from the accomplices who also have high spamicities, while a non-colluder’s spamicity will be low for the low collusiveness received from either colluders or other non-colluders. In other words, *users who provide strong collusion evidence about other high-spamicity users tend also to have high spamicities (i.e., giving themselves away)*. Inspired by this observation, we consider to extend the Markov random walk model [43] to obtain a stabilized ranking of user spamicities, where the spamicities of users can be interpreted as the “authorities” of web pages in hyperlink structure. In the context of MRW, each directed user pair $\langle v_i, v_j \rangle$ is associated with a collusion weight $f(i \rightarrow j)$ which is equal to the sum of the product of the collusiveness between v_i and v_j , $\Phi(i, j)$ and the corresponding confidence of v_i in the evidence about v_j , $\Omega_i(j, R)$, over all products $\mathcal{P}_i$ reviewed by v_i :

$$f(i \rightarrow j) = \sum_{k \in \mathcal{P}_i} \Phi(i, j) \cdot \Omega_i(j, R_{k,i}(\zeta)) \quad (\text{Eq. 4.7})$$

The transition probability from v_i to v_j is defined by normalizing the corresponding collusion weight as:

$$p(i \rightarrow j) = \begin{cases} \frac{f(i \rightarrow j)}{\sum_{j'=1}^{|V|} f(i \rightarrow j')} & \text{if } \sum f \neq 0 \\ 0 & \text{otherwise} \end{cases} \quad (\text{Eq. 4.8})$$

Recall that $\mathcal{V}$ is the set of all users in the site. The row-normalized transition matrix is $\mathbf{M} = [\mathbf{M}_{i,j}]_{|\mathcal{V}| \times |\mathcal{V}|}$ where $\mathbf{M}_{i,j} = p(i \rightarrow j)$. To meet the conditions of $\mathbf{M}$ being a stochastic matrix and guarantee the existence of a stationary distribution, the rows with all zeros are substituted by a smoothing vector with each element set to $1/|\mathcal{V}|$. Finally, the spamicity s_i of v_i can be formulated in a recursive manner as follows:

$$s_i = \eta \sum_{j:i \rightarrow j} s_j \cdot \mathbf{M}_{i,j} + (1 - \eta) \frac{1}{|\mathcal{V}|} \quad (\text{Eq. 4.9})$$

where $\eta \in [0, 1]$ is a damping factor to control the probability of teleportation, which is set to 0.85 in our case. The ranking of users by their spamicities is obtained by running Eq. 4.9 iteratively until convergence.

4.3 Collusiveness Measure: Pairwise Features

In this section, we present multiple heterogeneous pairwise features to measure the collusiveness between user pairs, i.e., $\Phi(i, j)$ in Eq. 4.6. Multiple observable and retrievable review data such as ratings, timestamps, text, and products/brands being reviewed are considered. It is worth noting that each dimension can only cover a portion of colluding behaviors. For example, not all the pairs having similar ratings would necessarily imply the occurrence of collusion. Thus, a final combination is needed for robustness and comprehensiveness purpose.

4.3.1 Product-based Sentiment Deviation (PSD)

Having the same spamming goal of promoting/demoting the reputations of the targets, colluders from the same spam campaigns tend to express similar opinions by giving similar ratings, which can lead to lower rating deviation between them. Given a pair of users v_i and v_j , the Product-based Sentiment Deviation (PSD) computes the degree of rating deviation towards their commonly reviewed products:

$$\phi_{psd} = \frac{2}{1 + e^{d_r^P(i,j)}} \cdot \alpha_{ij} \quad (\text{Eq. 4.10})$$

where $\alpha_{ij} = \frac{|P_i \cap P_j|}{|P_i \cup P_j|}$ is a damping factor accounting for the bias induced by the absolute difference between their respectively reviewed products; $P_{i(j)}$ is the set of products reviewed by $v_{i(j)}$. $d_r^P(i, j)$ is the average rating deviation of (v_i, v_j) over their commonly reviewed products $P_i \cap P_j$,

$$d_r^P(i, j) = \text{avg}_{p_k \in P_i \cap P_j} (|r_i^{p_k} - r_j^{p_k}|) \quad (\text{Eq. 4.11})$$

where $r_{i(j)}^{p_k}$ is the rating given by $v_{i(j)}$ for the commonly reviewed product p_k .

4.3.2 Product-based Time Deviation (PTD)

Following the same predefined spam schedule, colluders prone to behave in a lockstep manner. In other words, the temporal traces of their postings tend to be close on the commonly reviewed products. Given a pair of users v_i and v_j , the Product-based Time Deviation (PTD) evaluates the reviewing time gap on their commonly reviewed products:

$$\phi_{ptd} = \frac{1}{1 + d_t^P(i, j)^\lambda} \cdot \alpha_{ij} \quad (\text{Eq. 4.12})$$

where $\lambda \geq 1$ is a parameter to accelerate the rate of decay for $d_t^P(i, j)$, which is the average reviewing time deviation between v_i and v_j over $P_i \cap P_j$,

$$d_t^P(i, j) = \text{avg}_{p_k \in P_i \cap P_j} (|t_i^{p_k} - t_j^{p_k}|) \quad (\text{Eq. 4.13})$$

where $t_{i(j)}^{p_k}$ is the reviewing timestamp for $p_k \in P_i \cap P_j$ by $v_{i(j)}$.

4.3.3 Product-based Review Text Similarity (PTS)

Previous studies [31, 39, 37] observed that review spammers are inclined to write fake reviews with similar contents not only for saving efforts but also for expressing similar opinions. To capture this signal, given a pair of users v_i and v_j , the Product-based

Review Text Similarity (PTS) is formulated as the maximum cosine similarity between their review texts over $P_i \cap P_j$:

$$\phi_{pts} = \max_{p_k \in P_i \cap P_j} (\text{cosine}(v_i^{p_k}, v_j^{p_k})) \cdot \alpha_{ij} \quad (\text{Eq. 4.14})$$

where $v_{i(j)}^{p_k}$ is the text of the review written by $v_{i(j)}$ for $p_k \in P_i \cap P_j$. The text of each review is represented by a bag of bi-grams and $\text{cosine}(v, v')$ computes the cosine similarity of the bi-gram TF-IDF vectors of v and v' .

4.3.4 Brand-based Sentiment Deviation (BSD)

Brand information has been shown to be helpful [24, 30, 61]. In [61] the author found that instead of targeting at commonly reviewed products, colluders nowadays are assigned different sets of products for reviewing. However, all the compromised products may still have the same brands. Here we consider this “higher level” information. Given a pair of users v_i and v_j , the Brand-based Sentiment Deviation (BSD) computes the degree of average rating deviation towards their commonly reviewed brands:

$$\phi_{bsd} = \frac{2}{1 + e^{d_r^B(i,j)}} \cdot \beta_{ij} \quad (\text{Eq. 4.15})$$

where $\beta_{ij} = \frac{|B_i \cap B_j|}{|B_i \cup B_j|}$ is a damping factor corresponding to brand; $B_{i(j)}$ is the set of brands reviewed by $v_{i(j)}$. $d_r^B(i, j)$ is the average rating deviation between v_i and v_j over $B_i \cap B_j$,

$$d_r^B(i, j) = \text{avg}_{b_k \in B_i \cap B_j} (|r_i^{b_k} - r_j^{b_k}|) \quad (\text{Eq. 4.16})$$

where $r_{i(j)}^{b_k}$ is the average rating given by $v_{i(j)}$ for $b_k \in B_i \cap B_j$,

$$r_{i(j)}^{b_k} = \text{avg}_{p_{k'} \in P_{i(j)}, \text{brand}(p_{k'})=b_k} (r_{i(j)}^{p_{k'}}) \quad (\text{Eq. 4.17})$$

4.3.5 Brand-based Time Deviation (BTD)

Similar to the “product” version PTD, the Brand-based Time Deviation (BTD) computes the degree of reviewing time gap between users v_i and v_j at brand level:

$$\phi_{btd} = \frac{1}{1 + d_t^{\mathcal{B}}(i, j)^\lambda} \cdot \beta_{ij} \quad (\text{Eq. 4.18})$$

where $d_t^{\mathcal{B}}(i, j)$ measures the average deviation of the reviewing time intervals for $B_i \cap B_j$, between v_i and v_j :

$$d_t^{\mathcal{B}}(i, j) = \text{avg}_{b_k \in B_i \cap B_j} (|\max(T_i^{b_k}) - \max(T_j^{b_k})| + |\min(T_i^{b_k}) - \min(T_j^{b_k})|) \quad (\text{Eq. 4.19})$$

where $T_{i(j)}^{b_k}$ is the set of reviewing timestamps for $b_k \in B_i \cap B_j$ by $v_{i(j)}$.

$$T_{i(j)}^{b_k} = \{t_{i(j)}^{p_k} | p_k \in P_{i(j)}, \text{brand}(p_k) = b_k\} \quad (\text{Eq. 4.20})$$

4.3.6 Brand-based Review Text Similarity (BTS)

Similar to the “product” version PTS, the Brand-based Review Text Similarity (BTS) measures the maximum review text similarity towards $B_i \cap B_j$:

$$\phi_{bts} = \max_{b_k \in B_i \cap B_j} \left(\max_{\substack{p \in P_i, \text{brand}(p)=b_k \\ p' \in P_j, \text{brand}(p')=b_k}} (\text{cosine}(v_i^p, v_j^{p'})) \right) \cdot \beta_{ij} \quad (\text{Eq. 4.21})$$

4.3.7 Reviewing Activity Homophily (RAH)

For a spam campaign, the schedule or budget would affect the reviewing activity patterns of involved colluders. Specifically, these colluders tend to be busy during the campaign period while idle when no task is available. Based on this intuition, the Reviewing Activity Homophily (RAH) captures the commonality of posting activities of a pair, v_i and v_j . We first split the global timeline into small time slots $\{s_1, s_2, \dots, s_k\}$ with equal width τ . Each slot corresponds to a time period of interest. ϕ^{rah} can then be defined as:

$$\phi_{rah} = \frac{1}{1 + \left[\frac{KL(q_i \| q_j) + KL(q_j \| q_i)}{2} \right]^\lambda} \quad (\text{Eq. 4.22})$$

where $KL(\cdot\|\cdot)$ is the KL divergence. $q_{i(j)}$ is the distribution of $v_{i(j)}$ over the number of reviews posted in each of their common time slots $n_{i(j)}^s$,

$$q_{i(j)} = \left\{ \frac{n_{i(j)}^s}{\sum_{s' \in S_i \cap S_j} n_{i(j)}^{s'}} \right\}_{s \in S_i \cap S_j} \quad (\text{Eq. 4.23})$$

where $S_{i(j)}$ is the set of time slots within which $v_{i(j)}$ has posted at least one review.

4.3.8 Reviewing Lifetime Homophily (RLH)

It has been reported that spammers, unlike authentic users, typically do not use their accounts for too long [39]. The usage patterns of their accounts are expected to be similar with each other since they may follow a unified arrangement in a spam campaign, leading to similar time spans of their reviewing lifetimes. The Reviewing Lifetime Homophily (RLH) captures the difference of the reviewing lifetimes of (v_i, v_j) :

$$\phi_{rlh} = \frac{1}{1 + |LT_i - LT_j|^\lambda} \quad (\text{Eq. 4.24})$$

where $LT_{i(j)}$ is the lifetime of $v_{i(j)}$,

$$LT_{i(j)} = \max(T_{i(j)}^{\mathcal{P}}) - \min(T_{i(j)}^{\mathcal{P}}) \quad (\text{Eq. 4.25})$$

and $T_{i(j)}^{\mathcal{P}}$ is the set of reviewing timestamps of $v_{i(j)}$ for the products in $P_{i(j)}$,

$$T_{i(j)}^{\mathcal{P}} = \{t_{i(j)}^p | p \in P_{i(j)}\} \quad (\text{Eq. 4.26})$$

4.3.9 Pairwise Feature Combination

Each pairwise feature has already been normalized within 0 and 1. We integrate all dimensions via a convex combination, to obtain the overall collusiveness measure for each user pair (v_i, v_j) :

$$\Phi(i, j) = \sum_k w_k \cdot \phi_k(i, j) \quad (\text{Eq. 4.27})$$

where $\sum_k w_k = 1, w_k \geq 0$, k is the index for each pairwise feature. The weighting parameter $\mathbf{w} = \{w_k\}$ specifies the importance of each feature; bigger weights should be assigned to more effective ones. As our proposed framework is unsupervised, uninformative equal emphasis will be given to each feature dimension.

4.4 Empirical Analysis

In this section, we evaluate the performance of the proposed unsupervised framework FRAUDINFORMER in detecting collusive opinion fraud. We first evaluate the proposed heterogeneous pairwise features that are used to measure the collusiveness between colluders in Section 4.4.3. The evaluation is focused on the ability of each feature to distinguish the collusiveness between different types of user pairs. Then, to understand the behaviors of FRAUDINFORMER, sensitivity analysis is performed to assess the effects of different investigating range settings (Section 4.4.4) and confidence weighting schemes (Section 4.4.5). Finally, we compare the detection performance of FRAUDINFORMER with state-of-the-art unsupervised colluder detection approaches in Section 4.4.6.

4.4.1 Dataset

Following the previous Chapter, we use the dataset created in [61] for evaluation. Recall that in this dataset, a total of 1,937 colluders (+) and 3,118 non-colluders (−) are annotated in the product reviews of Amazon.cn. The reviews for all the products reviewed by these annotated users are used in our experiments, which involves 3,987 products, 140,258 users¹, and 265,793 reviews.

¹Singletons, a special type of spammers writing only one review in their lifetimes, are not considered in our experiments since the proposed pairwise features require users to have adequate reviewing histories. Studies like [58] can handle this issue.

4.4.2 Evaluation Criteria

Two well-known ranking based metrics are used for our experimental analysis [37].

1) Precision@ k : defines the precision at cut-off k in the ranking list which corresponds to the ratio of colluders in the top k ranks.

2) Normalized Discounted Cumulative Gain (NDCG): evaluates resulting rankings with respect to an ideal ranking based on users' spamicities, which favors rankings where colluders with highest spamicities are ranked at the top. NDCG@ k is defined to be:

$$NDCG@k = \frac{DCG@k}{IDCG@k} \quad DCG@k = \sum_{i=1}^k \frac{2^{c_i} - 1}{\log_2(1 + i)} \quad (\text{Eq. 4.28})$$

where c_i is the binary class value of the user (1:colluder, 0:non-colluder) ranked at position i . IDCG@ k is the discounted cumulative gain (DCG) of the ideal ranking at position k where all colluders are ranked higher than all non-colluders.

4.4.3 Effects of Pairwise Features

As the measures of collusiveness between users, the proposed multiple pairwise features are the core to differentiate colluders from non-colluders in FRAUDINFORMER. Effective pairwise features are expected to distinguish the collusiveness of three types of user pairs, i.e., colluder to colluder pair, non-colluder to colluder pair, and non-colluder to non-colluder pair. Specifically, the collusiveness of (colluder,colluder) pairs should be higher than the other two. Figure 4.2 shows the distribution of each pairwise feature score (collusiveness) over the three user pair types. Note that to be consistent with the way of collusiveness propagation in FRAUDINFORMER, only the pairs of users whose positions in a product review list are within an investigating range (IR) are considered. Here the IR is empirically set to be 25, close to the half of the average distance between two colluders in the dataset which is 56.07. We also compute the χ^2 value [63] for each

feature; the larger the χ^2 value is, the higher discriminative power the corresponding feature has.

From Figure 4.2 we can see that (colluder,colluder) pairs dominate the area of high collusiveness in each histogram. Among these pairwise features, the Brand-based Sentiment Deviation (BSD) achieves the best performance not only in terms of the clear separation observed in the histogram but also the highest χ^2 values. Another notable observation is that brand-based pairwise features generally outperform the product-based counterparts (except PTD and BTD, despite the observed small differences), implying that colluders nowadays are more likely to target brands than specific products. By grouping different products with the same brands together, the collusive signals will be accumulated and become more prominent among colluders.

4.4.4 Effects of Investigating Range

The optimal choice for investigating range is non-trivial; a small value may lead to some distant accomplices being not covered in the accusation (low recall) while a large one may not only bring along noise by accounting for some non-colluders (low precision), but may also result in inefficient computation. Figure 4.3 shows the effects of different investigating ranges (IRs) on the overall ranking performance. Here we model IR as a function of the length n_k of a product review list ℓ_k (Section 4.2.1). We then consider four types of functions:

- Constant: $\zeta(n_k) = 5, 10, 15, 20$
- Random: $\zeta(n_k) = \text{random}(1, 20)$
- Root: $\zeta(n_k) = \sqrt{n_k}, \sqrt[3]{n_k}$
- Logarithmic: $\zeta(n_k) = \log_2(n_k + 1)$

Figure 4.2: Histograms of pairwise features over three types of pairs: colluder to colluder (C-C), non-colluder to colluder (NC-C), non-colluder to non-colluder (NC-NC). The x-axis denotes the pairwise feature scores (collusiveness) and the y-axis denotes the corresponding frequencies. χ^2 statistics is shown in the parenthesis in each sub-caption.

From the experiments, we noted two observations.

The TYPE I noise. For constant IR functions, as IR increases from 5 to 20, we find a monotonic degradation on both NDCG and Precision (Figure 4.3). Further experiment (Figure 4.4.a) is conducted with fine-grained $IR \in \{1, 2, \dots, 25\}$ and the performance is shown to clearly decline as IR increases in higher ranks ($k=100,500$) and there is a small arch around $IR=2$ when k becomes larger ($k=1000,1500$). To derive further insights into why this happens, we compute the average ratio of nearby colluders in the data which turns out to decrease when IR rises (solid line in Figure 4.4.b) while the average ratio of nearby non-colluders increases slightly (dashed line in Figure 4.4.b). This shows that as IR gets larger, in long enough product review lists, a colluder will have more

4.3.a: NDCG for top 1000 ranks

4.3.b: Precision for top 1000 ranks

Figure 4.3: Ranking performance for different investigating range (IR) settings. Uniform kernel function is used as the confidence function.

4.4.a: NDCG vs. IRs

4.4.b: Neighborhood variations for different IRs

Figure 4.4: The rise of TYPE I noise from the collusion weight allocation in FRAUDINFORMER.

non-colluders involved in the accusation who aggregately share an increasing portion of collusion weights in the collusiveness propagation, leading to higher spanicities of these

non-colluders in the final ranking (Eq. 4.7). We denote this kind of noise rising from the collusion weight allocation in FRAUDINFORMER as the TYPE I noise which may lead to performance deterioration.

The TYPE II noise. We then consider the variable IRs (Random, Root, and Logarithmic) whose performance is much poorer than that of constant IRs. This is anti-intuitive because the performance of variable IRs is expected to remain moderate between those of larger constant IRs (e.g., 25) and smaller ones (e.g., 5) rather than worse than both cases. This may be partially due to the effect of the TYPE I noise because the IRs of longer product review lists (ℓ_{Long}) are correspondingly larger, which makes it underperform small constant IRs. On the other hand, regarding the poorer performance compared to larger constant IRs, we argue that it may be attributed to two facts that in the dataset:

- 1) Most of the colluders write fake reviews for unpopular products with short user lists (ℓ_{Short}), e.g., 51.5% colluders have reviewed 43.9% products with $|\ell_{Short}| \leq 12$ and nearly 3 of them (25%) are colluders on average;
- 2) The positions of such colluders in ℓ_{Short} are quite close to each other, e.g., the average and median distances between two consecutive colluders in ℓ_{Short} with $|\ell_{Short}| \leq 12$ are 1.23 and 1 respectively.

As such, when using the proportional variable IRs, in unpopular products a colluder will have less neighbours most of whom are also colluders with similarly high collusiveness, while in popular products a non-colluder will have more neighbours most of whom are also non-colluders with similarly low collusiveness. By normalizing the collusion weights to construct a stochastic transition matrix (Eq. 4.8), the collusiveness in both cases will be smoothed out. Thus, non-colluders in ℓ_{Long} will gain higher “authority” during the global ranking due to the greater number of their non-colluder neighbours covered by larger IRs. In contrast, when using large constant IRs (e.g., 25), the colluders in ℓ_{Short}

will have a portion of non-colluder neighbours as well, making the transition probability unevenly distributed over colluder neighbours and non-colluder ones, which is just the proper reflection of the difference between the collusiveness value of C-C and C-NC pairs. We denote this kind of noise rising from the construction of transition matrix (Eq. 4.8) as the TYPE II noise.

As a summary, a desirable investigating range should meet the following conditions: 1) it should not be set too large in *popular* products for restricting the “authority” of non-colluders, and 2) it should not be set too small in *unpopular* products so as to maintain the coverage of colluders and also to boost the “authority” of colluders by providing the opportunity of intensifying the differences between positives (C-C pairs) and negatives (NC-C and NC-NC pairs) in the process of unsupervised learning.

4.4.5 Effects of Confidence Weighting

We experiment with different confidence weighting functions which encode different patterns of users’ confidence when providing evidence about each other. The results are shown in Figure 4.5. We can see that the Epanechnikov kernel with $IR=5$ (Epan_IR5) performs the best and in terms of symmetric confidence the Epanechnikov kernel outperforms the uniform kernel in all IR settings. This is reasonable since the Epanechnikov kernel assigns more belief to the evidence about closer users, which matches the temporal locality property of spam reviews [14].

On the other hand, the asymmetric function (the TopK kernel) is shown to outperform all symmetric counterparts. Recall that the TopK kernel trusts the evidence about the ones with higher collusiveness. To further analyze the nature of the TopK kernel, an additional experiment is conducted to study the impacts from different $K = \{1, 5, \dots, 50\}$ and IR settings (Figure 4.5.b). Note that $K \leq 2 \times IR$ while $K = 2 \times IR$ is equivalent to the uniform kernel with $IR=K/2$ (the red dashed lines highlight the largest K corresponding to each IR setting). Several observations are noted.

- 1) As K increases the performance gets better for all IR settings. This may be ascribed to the TYPE II noise; when K is small, for a specific colluder, the similarly high collusiveness with other nearby colluders will be smoothed out by the normalization, making it no difference with the case of non-colluders.
- 2) It shows that for a fixed K , the performance drops as IR becomes larger. This differs from the TYPE I noise as the number of neighbours does not change which is K all the time. A potential explanation is that although K is fixed, the top K selected neighbours for a particular user are changing as IR becomes larger, and the chance of obtaining neighbours with similar collusiveness scores increases as well, which would amplify the effect of the TYPE II noise that the similarly high collusiveness of C-C pairs of a colluder would be neutralized after normalization.
- 3) It shows that empirically the performance of the TopK kernel will converge to that of the uniform kernel as K goes to $2 \times \text{IR}$.

4.5.a: NDCG for top 1000 ranks

4.5.b: The performance of the TopK kernel (NDCG@1000)

Figure 4.5: Evaluation of different confidence weighting functions.

4.4.6 Comparison Analysis

We compare our framework with other competitors that exploit pointwise features for colluder detection.

(1) **GSRank** [37]: by modeling the relationships among spammers, spammer groups, and target products, GSRank, a state-of-the-art unsupervised algorithm for detecting spammer groups, uses eight well-designed group-based pointwise features to characterize colluder behaviors. We implement GSRank and the eight group-based features for the comparison.

(2) **Learning to Rank**: this is another way of integrating the aforementioned pointwise group-based features where a training ranking can be obtained by sorting users based on each feature in descending order (the group-based features assign higher scores to colluders than non-colluders). The resulting learned ranking model is in effect an optimal combination of the rankings produced by the eight group-based feature functions. In our experiments, two learning to rank algorithms - SVMRank [25] and RankBoost [17] - are adopted and parameters are tuned to obtain the optimal performance.

For FRAUDINFORMER, according to the previous experimental analysis, we choose the Epanechnikov kernel as the confidence function and the IR is set to 5. The algorithm converges after 150 iterations given that L_∞ norm is used to measure the difference between the spamicity vectors of consecutive iterations and the algorithm terminates when the difference $\leq 10^{-6}$. We compare with the learning to rank algorithms using 10-fold cross validation and the performance is averaged over all test folders (each test folder includes ≈ 500 examples). For GSRank, we use the entire dataset as one test folder since training is not needed in both GSRank and FRAUDINFORMER. All the improvements of our method over the baselines are significant at the confident level of 95% based on two-tailed t-test.

		SVMRank (gain)	RankBoost (gain)	FraudInformer
Precision@k	50	0.844 (11.6%)	0.818 (15.1%)	0.942
	100	0.804 (7.7%)	0.780 (11.0%)	0.866
	150	0.764 (2.7%)	0.738 (6.3%)	0.785
	200	0.717 (-1.2%)	0.701 (0.9%)	0.708
	250	0.672 (2.4%)	0.667 (3.1%)	0.688
NDCG@k	50	0.853 (11.9%)	0.825 (15.7%)	0.955
	100	0.819 (8.6%)	0.794 (12.1%)	0.890
	150	0.784 (6.1%)	0.759 (9.6%)	0.832
	200	0.757 (2.1%)	0.738 (4.7%)	0.773
	250	0.830 (1.3%)	0.820 (2.5%)	0.841

Table 4.1: Performance comparison between FRAUDINFORMER and the learning to rank algorithms on AmazonCn dataset. The “gain” in the parenthesis means the improved margin of our method over the baseline approaches.

Figure 4.6: Performance comparison between FRAUDINFORMER and GSRank in NDCG@k ($k \in [100, 1000]$) on AmazonCn dataset.

As shown in Table 4.1, our method in general outperforms both leaning to rank algorithms. For $k = 50$ the performance improves at least by 11.6% in Precision@k and 11.9% in NDCG@k. For the comparison with GSRank² shown in Figure 4.6, our method performs at least as well as GSRank. At higher rank positions ($k=100,200,300$), both methods can achieve promising results. However, when k exceeds 400, the performance of GSRank drops significantly. We further inspect the colluders ranked within [400,

²The result of Precision@k is similar and omitted here.

1000] by FRAUDINFORMER yet missed by the top 1000 ranks of GSRank, finding 379 colluders in total and 162 ($162/379=42.7\%$) of them are ranked at the bottom 1000 by GSRank. Among the Bottom1000 colluders, 124 ($124/162=76.5\%$) appear in ≤ 4 groups. We also find that the behaviors of some groups are not well-captured by the proposed group based features in [37], including Group Size, Group Support Count, and Group Deviation. This is reasonable since the success of GSRank hinges heavily on the mutual enhancements between related entities (i.e., groups, group members, and products); group members with low group feature scores will obtain low spamicities in GSRank. As a result, the spamicities of colluders who review many similar products with regular users may become even lower than those active users who happen to review many popular products together. Moreover, among the 38 remaining colluders ranked at the bottom 1000 by GSRank, 24 of them are involved in groups with large sizes. This may also be problematic because in large groups, certain collusion signals exhibited by a portion of group members can potentially be concealed if the characterization is conducted from a group-level perspective. In contrast, our method operates with a finer granularity by revealing connections between pairs of colluders directly. Although some colluders may not seem suspicious at the group level, their collusiveness could be captured at pairwise level by the proposed pairwise features.

4.4.7 Generalization Evaluation

As before, we conduct further evaluation to validate the generalization ability of FRAUDINFORMER on additional datasets. The YelpZip dataset is used as with the previous chapter. To align with the previous subsection, the evaluation is achieved by conducting comparison analysis on YelpZip dataset. For FRAUDINFORMER, the optimal performance is achieved by choosing the Uniform kernel as the confidence function and setting the investigating range (IR) to the length of the corresponding product review list. Table 4.2 and Figure 4.7 show the comparison results.

		SVMRank (gain)	RankBoost (gain)	FraudInformer
Precision@k	50	0.681 (15.5%)	0.617 (27.5%)	0.787
	100	0.589 (14.2%)	0.578 (16.4%)	0.673
	150	0.554 (8.3%)	0.529 (13.4%)	0.600
	200	0.502 (4.9%)	0.448 (17.6%)	0.527
	250	0.458 (6.1%)	0.442 (9.9%)	0.486
NDCG@k	50	0.687 (12.8%)	0.673 (15.1%)	0.775
	100	0.616 (13.1%)	0.623 (11.8%)	0.697
	150	0.582 (9.4%)	0.579 (10.0%)	0.637
	200	0.537 (8.2%)	0.501 (15.9%)	0.581
	250	0.498 (7.8%)	0.488 (10.0%)	0.537

Table 4.2: Performance comparison between FRAUDINFORMER and the learning to rank algorithms on YelpZip dataset. The “gain” in the parenthesis means the improved margin of our method over the baseline approaches.

Figure 4.7: Performance comparison between FRAUDINFORMER and GSRank in NDCG@k ($k \in [100, 1000]$) on YelpZip dataset.

In Table 4.2, we compare FRAUDINFORMER with the learning to rank algorithms using 10-folder cross validation and the performance is averaged over all the test folders. We can see that although the overall results of all the compared methods on YelpZip are not comparable to those on AmazonCn (Table 4.1), FRAUDINFORMER still perform better than the baselines on both metrics. The averaged improvements are 13.3% in Precision@k and 11.4% in NDCG@k. Figure 4.2 demonstrates the ranking results of FRAUDINFORMER and GSRank on YelpZip dataset. The results are obtained by per-

forming both methods on the full dataset. As with the case of AmazonCn dataset, we can again make the observation that both methods achieve comparable performance at the beginning when k is small ($k \leq 200$). However, as k increases, the gap becomes larger, which goes to 16.0% when $k = 1000$. This shows that FRAUDINFORMER has a stronger ranking ability than GSRank by ranking more suspicious users at the top. The consistent improvements obtained on both AmazonCn and YelpZip datasets can be attributed to the effectiveness of the proposed collusiveness measures capturing interconnections between colluders. Such interconnections are more indicative than the group-based pointwise features proposed in [37] which pay more attention to the exterior of spam campaigns but ignore the correlations between colluders from the same campaigns.

4.5 The Adversarial Offense

In this section, we consider the adversarial challenge by answering the question that how could an adversary do to evade detection of our method, and how much damage would be caused by such events.

By following the same rules and plans, colluders in a spam campaign are unlikely to have completely different targets or opposite opinions. However, they are able to manipulate their positions in the product review lists so as to escape detection. In general, there are two adversarial scenarios.

4.5.1 Aggressiveness Control

In the first case, colluders can evade detection by avoiding the coverage of the investigating range (IR) used by FRAUDINFORMER. Recall that in FRAUDINFORMER, the IR parameter is used to find the target set of users for subsequent collusiveness propagation. If the value of IR is discovered by colluders, they can choose an posting interval

larger than this value, so as not to appear in the same target set examined by FRAUDINFORMER. For example, if IR is set to 10, colluders can place their spam reviews at an interval of 11 to avoid being “accused” by each other in FRAUDINFORMER. In the dataset, we find that colluders are not always necessarily centralized within a specific region in the product review lists, and some may even be far away from the majority. However, blindly improving the recall (i.e. to cover more colluders) by enlarging the IR may reduce the precision of FRAUDINFORMER (as discussed in Section 4.4.4). To balance this trade-off, we have the following lemma.

Lemma 1. *Assume that in a spam campaign where each of N colluders is asked to review m out of M products that all have review lists with equal-length L . Each fake review generates one unit of impression which represents the utility of spam. Then the average impression generated by this spam campaign on each product is bounded by $\frac{\gamma^{R-1}}{\gamma^R - 1}$ if an exponential decay function $\frac{1}{\gamma^R} (\gamma > 1)$ is used to model the impression, where R is the investigating range used by FRAUDINFORMER.*

Proof: To evade the capture of other accomplices within the investigating range R , the maximum impression a spam campaign can achieve is when all the members place their reviews on the product review list one after another with an interval of R , starting from the top of each product review list. In this case, the total impressions achieved by this spam campaign is:

$$\begin{aligned}
 I_{total} &= M \sum_{i=0}^{\lfloor \frac{mN}{M} \rfloor} \frac{1}{\gamma^{R \cdot i+1}} + (Nm - M \lfloor \frac{Nm}{M} \rfloor) \frac{1}{\gamma^{\lfloor \frac{Nm}{M} \rfloor + 1}} \\
 &\leq M \sum_{i=0}^{\lfloor \frac{mN}{M} \rfloor + 1} \frac{1}{\gamma^{R \cdot i+1}} \\
 &\leq M \frac{\frac{1}{\gamma}}{1 - \frac{1}{\gamma^R}} \\
 &= M \frac{\gamma^{R-1}}{\gamma^R - 1}
 \end{aligned} \tag{Eq. 4.29}$$

Then we have $I_{ave} = \frac{I_{total}}{M} = \frac{\gamma^{R-1}}{\gamma^R - 1}$. $\square$

We can see that the average impression decreases very fast, reciprocally, as the IR increases. For example, when $\gamma = 1.1$, we have $\frac{I_{ave}(IR=1)}{I_{ave}(IR=10)} = 6.75$, which means that a moderate IR can significantly weaken and further limit the effects of spam campaigns. On the other hand, moderate IRs can effectively catch most of the colluders (Section 4.4.4), since the manipulation of review positions is not always an easy task in practice; for colluders, it is unclear how long would it take for regular users to post reviews right after their spam reviews being posted. Thus, to achieve the predefined spamming effects, colluders have to spam as quickly as possible, resulting in concentration in collusion attacks.

4.5.2 Camouflage at Rush-hour

In the second case, colluders can use camouflage to disguise themselves by only taking actions at the rush-hour when more reviews are expected to be posted (such as hours between 7pm and 11pm). In this case, colluders may hope that increasing the density in the investigating range could lower the probability of being exposed to FRAUDINFORMER. However, our method is not density aware; it works by propagating collusiveness between the set of users within the investigating range, no matter how many users in this range. If the members of a spam campaign are covered by the investigating range, they will still be captured by FRAUDINFORMER as long as they have exhibited suspiciously similar behaviors indicated by the proposed collusiveness measures (Section 4.3). Note that the collusiveness measurements on normal users would be milder than those on colluders from the same campaigns (see Figure 4.2). On the other hand, if the intervals between their reviews are larger than the investigating range, the spamming effects of this spam campaign would then become weaker, according to Lemma 1. As a result, in either case, camouflage at rush-hour cannot effectively help colluders cover their traces.

4.6 Summary

In this chapter, we study the problem of collusive reputation fraud in unsupervised settings. We have proposed a novel detection framework FRAUDINFORMER for unsupervised collusive reputation fraud detection. It advances the state-of-the-art in three significant ways. First, the mutual disclosure model in FRAUDINFORMER provides an intuitive and effective way of conducting detection in unsupervised settings. Through proper collusive evidence propagation, re-weighting, and ranking, colluders who are involved in the same campaigns can automatically be captured in a collective manner. Second, the proposed heterogeneous pairwise features can effectively and directly characterize the behavioral correlations between colluders. Different from existing group-based features, the characterization of pairwise features is more fine-grained, and can find stealthy colluders whose behaviors may not seem suspicious at group level. Finally, it decouples the processes of colluder characterization and detection, such that newly-designed features and specialized detection algorithms can easily and flexibly be integrated to achieve further performance improvements. Moreover, theoretical analysis has been conducted to study the robustness of the proposed method in defending against adversarial challenge. We prove that FRAUDINFORMER can significantly limit the damage caused by a campaign even when it attempts to compromise the system. Empirical experiments validate the effectiveness of our method and show that it significantly outperforms state-of-the-art competitors by revealing colluders with more concealed behaviors.

Chapter 5

LCM: A Latent Variable Model for Collusion Prediction and Inference in Crowdsourced Opinions

In this chapter, we address the collusive reputation fraud problem from a probabilistic perspective. As mentioned before, existing approaches to this problem rely on characterizing indicative collective behaviors of colluders for model building. Colluders from the same fraud campaigns may exhibit synchronized behaviors due to their being instructed to target the same items within the campaign period. However, most collective behaviors examined in existing solutions are focused on the external appearance of fraud campaigns, such as the campaign size and target set size, which may become invalid once colluders change their behaviors collectively. Moreover, the detection algorithms used in existing approaches are designed to only make collusion inference on the input data; predictive models that can be deployed for emerging fraud detection cannot be learned from the data. Therefore, to complement existing approaches in terms of characterization and detection, we explore more subtle behavioral trails in collusive fraud practice. In particular, a suite of homogeneity-based measures are proposed to capture the interrelationships between colluders within campaigns (Section 5.1). Moreover, a novel statistical model is proposed to further characterize, recognize, and predict collusive fraud in crowdsourced

opinions (Section 5.3). The proposed model is fully unsupervised and highly flexible to incorporate effective measures available for better modeling and prediction. Experimental analysis on real-world datasets is provided to validate the effectiveness of the proposed framework and demonstrate its superiority over state-of-the-art solutions (Section 5.4). Finally, section 5.5 concludes this chapter.

5.1 Collusive Behaviors within Campaigns

Effective capture of collective behaviors of colluders plays an important role in collusive reputation fraud detection. In this section, we propose our approach to characterizing collusive fraud in online reviews.

As discussed, colluders from the same campaigns can exhibit unique collective behavior patterns. However, existing features may not be able to tackle colluders of all kinds. Some of these features are designed mainly for finding large campaigns which involve a non-trivial number of colluders and targets. For example, *Group Size* and *Group Support Count* proposed in [37] assume groups having larger size and attacking a greater number of targets are more suspicious. However, these features are likely to miss stealthier attacks where large campaigns split into smaller ones for different subsets of targets. Also, there are features regarding colluders as anomaly and measuring how much their behaviors would deviate from those of general users. For example, *Group Deviation* proposed in [37] evaluates the difference between a group and the remaining population on a target in terms of review rating. The larger the deviation between the average rating given by a group and the remaining population, the more likely this group is a colluder group. However, due to their advantage in manpower, colluders could easily turn themselves into the majority by review-flooding the targets, so that the deviations can be eliminated.

To cope with the above issues, we develop a suite of homogeneity-based collusive behavior measures (h-CBMs) to inspect the homogeneity in the collective behaviors of colluders. In other words, our focus is on the similar behaviors of colluders exhibited during their working for the same campaigns. To facilitate the discussion of the proposed h-CBMs, we have the following definitions:

Definition 4 (Candidate Colluder Group). *To capture the collective behaviors of colluders, we borrow the concept of **candidate colluder group** from [37]. A **candidate colluder group** refers to a group of users who have reviewed multiple items together¹.*

Note that a candidate colluder group is not necessarily a true colluder group, since normal users can also review the same items due to similar preferences. However, it is a necessary condition for a user to become a colluder. *Given the notion of candidate colluder group, our goal of collusive reputation fraud detection then reduces to differentiating colluder groups from non-colluder groups.*

Definition 5 (Colluder Group). *A **colluder group** consists of colluders involved in the same fraud campaigns, while a **non-colluder group** consists of either normal users or the mix of normal users and colluders². Note that a fraud campaign can correspond to one or more colluder groups.*

The proposed h-CBMs are imposed on the candidate colluder groups, and the characterization is based on four behavioral dimensions that release strong signals related to collusive reputation fraud. They are related to the items reviewed by a group (Target-based h-CBMs), the ratings given by a group (Rating-based h-CBMs), the timestamps

¹In [37], the equivalent term used is “candidate spammer group”. The authors used frequent itemset mining (FIM) to generate groups consisting of at least two users who have reviewed at least three items together.

²Our assumption of treating the mixed groups that contain both colluders and normal users as negatives (i.e., non-colluder groups) is based on the consideration that the colluders and normal users in the mixed groups are irrelevant with their own underlying motivations behind. There should be no difference between groups that contain only normal users and the mixed ones.

of the reviews given by a group (Temporal-based h-CBMs), and the member activeness of a group (Activity-based h-CBMs).

Notation: The following notations are used to define the proposed h-CBMs below. We use g to denote a candidate colluder group and $\mathcal{M}_g$ for its members. Similar to $\mathcal{I}_v$ representing the set of items reviewed by a user v , we use $\mathcal{I}_g$ to represent the set of items reviewed by a group g .

5.1.1 Target-based h-CBMs

First and foremost, a significant characteristic of collusive reputation fraud would be related to the common items reviewed by colluders. Since no fraud campaigns will launch without specifying any targeted items, and colluders are required to review these items to get paid, the connections among colluders can be best revealed by inspecting the common items they focus on. Intuitively, colluders working for the same campaigns would be assigned to the same targets, which would inevitably leave common segments in colluders' reviewing histories. One thus can trace the collusion frequency of a group by counting the number of common items reviewed by its members. The higher the count, the more likely the members are to commit collusive fraud together. Here, we use Jaccard Similarity for such **target consistency (TC)** measurement. Specifically, we inspect the Jaccard Similarity of the set of items reviewed by the members of group g as follows,

$$h_g^{TC} = \frac{\#\{\cap_{v \in \mathcal{M}_g} \mathcal{I}_v\}}{\#\{\cup_{v \in \mathcal{M}_g} \mathcal{I}_v\}} \quad (\text{Eq. 5.1})$$

where $\#\{\cdot\}$ evaluates the cardinality of a set, and the numerator is the number of common items reviewed by all the members $\mathcal{M}_g$, and the denominator serves as a normalizer by computing the number of items reviewed by either of them. This similarity is essentially evaluating the concentration of a group of users on their common targets.

5.1.2 Rating-based h-CBMs

The review rating an item receives conveys the overall opinion from the general public, and thus becomes the target of manipulation. Since the ultimate objective of fraud campaigns is to promote or demote the targeted items, colluders are required to give highly positive or negative ratings to a particular target together so as to quickly lift up or bring down its average rating. It thus should be suspicious if a group of users often provide consistently high/low ratings to their commonly reviewed items. To capture this intuition, we inspect the **rating consistency (RC)** of a user group g by computing the variance of the ratings given by its members $\mathcal{M}_g$ to the set of commonly reviewed items $\mathcal{I}_g$; the lower the variance, the more consistent the provided ratings are, and the more suspicious the group should be. The formula for computing RC is as below,

$$h_g^{RC} = \max_{i \in \mathcal{I}_g} (\text{var}(\{r_{v,i} | v \in \mathcal{M}_g\})) \quad (\text{Eq. 5.2})$$

We can see that Eq. 5.2 first evaluates the variance of ratings given by a group g to each of the common items in $\mathcal{I}_g$. It then chooses the maximum variance due to the consideration that in general colluders' ratings would be consistent on most of their common items, such that the maximum variance would be still quite low for most of the time. In contrast, the rating variances of a non-colluder group should be randomly distributed, rendering the maximum value more likely to be relatively larger.

5.1.3 Temporal-based h-CBMs

Timing is vital to fraud campaigns since efficiency usually brings more profits. Moreover, to quickly achieve the desired manipulation effect, a fraud campaign will typically set up a schedule and the campaign members are required to finish their tasks in time. This, ironically, can be used to our advantage as the imposed time constraint on campaign activities would restrict the actions of colluders within a bounded time interval. Measures

based on time window can therefore be used to capture such synchronization of colluders' behaviors. We then employ the **temporal synchronization (TS)** measure to inspect the review posting timing of a group of users on each of the targeted items, as shown below,

$$h_g^{TS} = \max_{i \in \mathcal{I}_g} (\text{var}(\{t_{v,i} | v \in \mathcal{M}_g\})) \quad (\text{Eq. 5.3})$$

Similar to the rating consistency discussed before, Eq. 5.3 first computes the variance of the review posting timestamp of group g on each of its targeted items in $\mathcal{I}_g$. It then chooses the maximum variance due to the same reason as the case in the rating consistency measure. A lower TS value will indicate a group is more suspicious for its synchronized reviewing behavior in time.

In addition to the absolute review posting timestamps, the relative orders of review postings can also be used to reveal the orchestration of a fraud campaign. The utilization of review order comes in two flavors. First, it can consolidate the capture of review synchronization because the reviews posted at proximate times are more likely to be placed near to each other on the review pages. Second, as most of the compromised items would be the unpopular ones with few reviews received [34] (otherwise, there is no reason for reputation manipulation), even though the campaign period can be longer in some cases resulting in larger time gaps between consecutive colluder reviews, the relative orders of the colluder reviews would be still quite close to each other, as it would be less likely that many truthful reviews are injected between the consecutive colluder reviews on unpopular items. In such cases, the consistency of review orders becomes more important to indicate the occurrence of collusive fraud. We thus introduce the **review order synchronization (ROS)** measure to evaluate the synchronization of the orders of colluder reviews as below,

$$h_g^{ROS} = \max_{i \in \mathcal{I}_g} (\text{var}(\{o_{v,i} | v \in \mathcal{M}_g\})) \quad (\text{Eq. 5.4})$$

We can see that Eq. 5.4 takes a similar form with Eq. 5.3 by computing the maximum over the variances of review orders of a group on common item set $\mathcal{I}_g$.

Another time-related collusion signal is the time when a colluder posts her first review. It is reported that general spammers tend to use accounts that are auto-generated or bought in batch for their first campaign practice [36]. Thus it is possible to spot another type of synchronicity by comparing the time when colluders use their accounts for the first time. Specifically, we propose the **first-review synchronization (FS)** measure as the variance of timestamps of the first reviews posted by the members of group g as below,

$$h_g^{FS} = \text{var}(\{\min(\{t_{v,i} | i \in \mathcal{I}_v\}) | v \in \mathcal{M}_g\}) \quad (\text{Eq. 5.5})$$

where the function $\min(\{t_{v,i} | i \in \mathcal{I}_v\})$ retrieves the time when a group member posts her first review.

5.1.4 Activity-based h-CBMs

The posting activeness of colluders can also reveal the dynamics of a particular fraud campaign due to their being required to follow the same campaign schedule and also share a part of the overall posting workload, which will make their activeness patterns very similar to each other. Specifically, in terms of the campaign schedule factor, we propose the **activity consistency (AC)** measure to compute the closeness between the most active moments of the group members. This is done by evaluating the variance of the most active moments of the group members. The most active moment of a user is defined as the date on which she posts most of her reviews. The formulation goes as follows,

$$h_g^{AC} = \text{var}(\{t_v^{max} | v \in \mathcal{M}_g\}) \quad (\text{Eq. 5.6})$$

where $t_v^{max} = \arg \max_{t \in T_v} (\#\{t_{v,i} = t | i \in \mathcal{I}_v\})$ retrieves the most active moment of user v .

To account for the workload sharing factor, one way to measure the workload would be counting the total number of reviews posted by a user. To share the overall workload of a campaign, the workload assigned to each colluder may be similar. In this case, we measure the **workload similarity (WS)** as the variance of the numbers of items reviewed by the members in group g as below,

$$h_g^{WS} = \text{var}(\{|\mathcal{I}_v| | v \in \mathcal{M}_g\}) \quad (\text{Eq. 5.7})$$

where $\mathcal{I}_v$ is the set of reviews posted by user v .

Finally, we may also observe colluders' correlation in terms of their account usage pattern. As colluders may use particular accounts for particular campaigns (or multiple campaigns), the accounts used for the same campaign(s) may experience similar lifetimes. We can then compare the lifetimes of the members' accounts in a group. In particular, the lifetimes of the accounts used for the same campaigns are expected to be of similar length. We employ the **account lifetime similarity (ALS)** measure which evaluates the variance of the account lifetimes of group members. The account lifetime of a user is defined as the time span over her first and last reviews,

$$h_g^{ALS} = \text{var}(\{\max(T_v) - \min(T_v) | v \in \mathcal{M}_g\}) \quad (\text{Eq. 5.8})$$

where $T_v = \{t_{v,i} | i \in \mathcal{I}_v\}$ is the set of timestamps of user v 's all postings.

Numerically, the value of each h-CBM is standardized using 0-1 scaling and thus in the real interval $[0,1]$. The variance-based h-CBM (i.e., all except TC) are converted to similarity-based measures by using the formula,

$$s_{\text{new}} = \frac{2}{1 + s_{\text{old}}} - 1 \quad (\text{Eq. 5.9})$$

where s_{old} and s_{new} are the values of a particular h-CBM before and after the conversion. After the conversion, for all h-CBMs, a value closer to 1 will suggest the evaluated group be more likely to be a colluder group.

It is worth noting that a merit of h-CBM is that they are parameter-free. This omits the need for parameter estimation based on held-out labeled data, given the fact that high quality annotation for reputation fraud detection is hard to obtain in practice.

5.2 Problem Formulation

In this section, we formulate our target problem of collusive fraud detection in on-line reviews. In a review site, we are given a set of N candidate colluder groups $\mathcal{G} = \{g_1, \dots, g_N\}$, in which each group g_n is associated with an M -dimensional h-CBM³ vector $\phi_n = \{\phi_n^{(1)}, \dots, \phi_n^{(M)}\}$, $\phi_n^{(m)} \in [0, 1]$. Then the set of all h-CBM vectors $\Phi = \{\phi_1, \dots, \phi_N\}$ constitutes our observed data about all the groups $\mathcal{G}$. The class label of each group g_n is denoted by a binary variable $z_n \in \{0, 1\}$, specifying whether it is a colluder group or not. As our context is unsupervised, all the class labels $\mathbf{Z} = \{z_1, \dots, z_N\}$ of $\mathcal{G}$ are unknown.

Now we define the problem of detecting collusive reputation fraud from a *probabilistic modeling* perspective as follows:

Problem 1. *Given a set of candidate colluder groups $\mathcal{G}$ with their observed h-CBM vectors Φ , the problem of detecting collusive fraud in online reviews involves two subtasks: 1) **infer** the posterior distribution of the class labels $\mathbf{Z}$ of $\mathcal{G}$, $p(\mathbf{Z}|\Phi)$, and 2) **predict** the class label $\hat{z}$ for an emerging group $\hat{g}$ based on the predictive distribution $p(\hat{z}|\hat{\phi}, \Phi, \mathbf{Z})$, where $\hat{\phi}$ is the h-CBM vector of $\hat{g}$.*

5.3 The Proposed Statistical Model

In this section, we propose a novel statistical model, named Latent Collusion Model (LCM), to solve the problem of collusive reputation fraud detection from a probabilistic perspective.

³Note that other collusion-oriented features can also be used in LCM. The term “h-CBM” here is for representational purpose only.

The key idea of LCM is to treat the unknown class labels of candidate colluder groups $\mathbf{Z}$ as latent, and consider the *reciprocal* relationship between the class labels $\mathbf{Z}$ and the observed h-CBM vectors Φ from two probabilistic modeling views: generative and discriminative. In general, the generative view considers the class labels as the *hidden factor* that causes the distinctive h-CBM distributions of colluder groups and normal groups, which allows us to perform the collusion inference task by taking a clustering-based generative approach that defines a generative process for generating group h-CBM distributions from the class labels. On the other hand, the discriminative view regards the class labels as the *hidden consequence* of the observed h-CBM distributions, which enables the learning of a predictive distribution for the collusion prediction tasks.

To further take advantage of the two view modeling, two variants of LCM are proposed. The first variant is a chaining-based model which performs the two subtasks of collusion inference and prediction by chaining the generative and discriminative views in sequence. The second one is a unified model that combines both views in an integrated framework. We will show that both the variants have their own merits and can complement each other in terms of collusion inference and prediction performance. For both variants, principled optimization algorithms are implemented to conduct model learning and inference.

5.3.1 Two Views of Collusive Reputation Fraud Modeling

As discussed, the goal of modeling collusive reputation fraud is to solve Problem 1 involving the two subtasks of collusion inference and prediction. Specifically, for the collusion inference task, the objective is to compute the posterior distribution $p(\mathbf{Z}|\Phi)$, while for the collusion prediction task, we need to derive the predictive distribution $p(\hat{z}|\hat{\phi}, \Phi, \mathbf{Z})$. It can be seen that in both cases the unknown quantity $\mathbf{Z}$ plays an important role, and serves as the connection between the two subtasks. As such, we can treat the class labels

$\mathbf{Z}$ as latent in LCM, and further considers the *reciprocal* relationship between the unknown class labels $\mathbf{Z}$ and the observed h-CBM vectors Φ from two probabilistic modeling views: generative and discriminative.

5.3.1.1 The generative view: occurred collusion inference

The generative view of LCM considers the class labels as the *hidden factor* that causes the h-CBM vectors. As mentioned, colluders often possess unique collective behavior patterns in feature space. Once we know the class label of a group, we can somehow generate its h-CBM vector based on the difference between the collective behavior patterns (distributions) of colluder and non-colluder groups, which essentially yields two distinctive clusters. Then we can take a clustering-based generative approach [4] to infer the posterior distribution $p(\mathbf{Z}|\Phi)$ so as to solve the inference task of Problem 1.

Specifically, the generative approach for this clustering postulates a generative process describing how the M dimensional h-CBM vectors of a mixture of N colluder and non-colluder groups can be generated by LCM:

- 1) For each group g_n , where $n \in \{1, \dots, N\}$, generate its class label z_n based on a cluster membership distribution p_z : $z_n \sim p_z$, where $z \in \{0, 1\}$ and $p_1 + p_0 = 1$;
- 2) For each h-CBM dimension $\phi_n^{(m)}$ of group g_n , where $m \in \{1, \dots, M\}$, generate the h-CBM value from the cluster-conditional distribution parametrized as $p(\phi_n^{(m)}|\lambda_{z_n}^{(m)})$: $\phi_n^{(m)} \sim p(\phi_n^{(m)}|\lambda_{z_n}^{(m)})$.

The parameter for the cluster-conditional distribution $\lambda_{z_n}^{(m)} = \{\lambda_0^{(m)}, \lambda_1^{(m)}\}$ essentially captures the latent collective behavior patterns of colluders ($\lambda_1^{(m)}$ with $z_n = 1$) and non-colluders ($\lambda_0^{(m)}$ with $z_n = 0$) on the m th h-CBM dimension. As each h-CBM dimension $\phi_n^{(m)}$ takes values in $[0, 1]$, we substantialize the cluster-conditional distribution $p(\phi_n^{(m)}|\lambda_{z_n}^{(m)})$ as a standard Beta distribution $\mathbf{Beta}(\phi_n^{(m)}|\alpha_{z_n}^{(m)}, \beta_{z_n}^{(m)})$ with parameters $\alpha_{z_n}^{(m)}$ and $\beta_{z_n}^{(m)}$, such that $\lambda_{z_n}^{(m)} = [\alpha_{z_n}^{(m)}, \beta_{z_n}^{(m)}]^T$.

Given this generative process, the likelihood of generating the observed data Φ based on the model parameter $\lambda = \{\lambda_0^{(m)}\}_{m=1}^M \cup \{\lambda_1^{(m)}\}_{m=1}^M$ can be written as:

$$L(\lambda|\Phi, \mathbf{Z}) = \prod_{n=1}^N \left[p_1 \cdot \prod_{m=1}^M \text{Beta}(\phi_n^{(m)}|\alpha_1^{(m)}, \beta_1^{(m)}) \right]^{z_n} \left[p_0 \cdot \prod_{m=1}^M \text{Beta}(\phi_n^{(m)}|\alpha_0^{(m)}, \beta_0^{(m)}) \right]^{1-z_n} \quad (\text{Eq. 5.10})$$

Then we are able to compute the posterior distribution $p(\mathbf{Z}|\Phi, \lambda)$, which is the goal of collusion inference, by maximizing the likelihood in Eq. 5.10 with respect to the model parameter λ using an EM algorithm [11]. More details of the optimization algorithms will be presented later in Sections 5.3.2 and 5.3.3.

5.3.1.2 The discriminative view: emerging collusion prediction

To model the relationship between the colluder group class labels and h-CBMs, the discriminative view, on the other hand, regards the class labels as the *hidden consequence* of the observed h-CBM vectors. This view is considered more natural, since the very goal of h-CBMs is to discriminate between the collective behavioral patterns of colluders and non-colluders; given the h-CBM vector of a particular group, we can tell its class label by referring to the semantics of each h-CBM dimension.

To solve the collusion prediction task of Problem 1, i.e., to compute the predictive distribution $p(\hat{z}|\hat{\phi}, \Phi, \mathbf{Z})$, we take a discriminative approach [4] where $p(z_n|\phi_n)$ is directly defined. Here, we use a logistic function to model this distribution:

$$p(z_n = 1|\phi_n, \mathbf{w}) = \sigma(\mathbf{w}^T \phi_n) \quad (\text{Eq. 5.11})$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the logistic function, $\mathbf{w} = \{w_0, \dots, w_M\}$ is an $(M+1)$ -dimensional weight vector to be estimated; w_0 is the weight for an additional dummy h-CBM $\phi_n^{(0)}=1$, which is added for notational compactness. In general, other suitable models for binary responses could also apply, such as the probit regression model [5].

Then, the predictions for a future group $\hat{g}$ with $\{\hat{\phi}, \hat{z}\}$ can be made by marginalizing the predictive distribution with respect to the model parameter $\mathbf{w}$:

$$p(\hat{z} = 1 | \hat{\phi}, \Phi, \mathbf{Z}) = \int p(\hat{z} = 1 | \hat{\phi}, \mathbf{w}) p(\mathbf{w} | \Phi, \mathbf{Z}) d\mathbf{w} \quad (\text{Eq. 5.12})$$

However, this marginalization requires the knowledge of all class labels $\mathbf{Z}$, and as our context is unsupervised, we are not provided with such information. Later we will show how to obtain this information by taking advantage of the generative view discussed before. Now, given the generative and discriminative views of the collusive fraud modeling, in the subsequent sections, we will introduce two variants of LCM that build probabilistic models based on these two views.

5.3.2 The Chaining-based Model

A straightforward and intuitive way of solving Problem 1 from a probabilistic perspective would be to tackle the two subtasks, i.e., collusion inference and prediction, in sequel. The idea is that as the collusion prediction task in Problem 1 acquires class labels for predictive distribution estimation, we can thus perform the collusion inference task first, which can produce the two clusters of colluder and non-colluder groups through the clustering-based generative approach described in Section 5.3.1.1. Then, by regarding the inferred cluster labels as the *pseudo* group class labels, we are then able to estimate the predictive distribution in the collusion prediction task based on the observed group h-CBM vectors and the inferred pseudo class labels. As the operations of the two tasks of collusion inference and prediction are decoupled during the process, the advantage of such a chaining-based modeling is the inherent flexibility that specialised models suitable for the respective tasks can be optimized and applied separately. The full chaining procedure for the collusion inference and prediction is listed as follows:

- 1) Prepare the input data by computing the h-CBM vectors $\Phi = \{\phi_1, \dots, \phi_N\}$ for a set of N candidate groups $\mathcal{G} = \{g_1, \dots, g_N\}$ whose class labels $\mathbf{Z} = \{z_1, \dots, z_N\}$ need to be inferred;
- 2) Perform **collusion inference** task: to infer the posterior distribution of the class labels $\mathbf{Z}$: $q(\mathbf{Z}|\Phi)$;
- 3) Obtain the “pseudo” class labels $\mathbf{Z}$ of candidate groups according to the decision rule: for each $g_n \in \mathcal{G}$, we have:

$$z_n = 1 \quad \text{if} \quad q(z_n = 1|\phi_n) > q(z_n = 0|\phi_n) \quad \text{else} \quad 0;$$

- 4) Perform **collusion prediction** task: to estimate the parameter $\mathbf{w}$ of the predictive distribution $p(\hat{z}|\hat{\phi}, \Phi, \mathbf{Z})$ based on $\mathbf{Z}$ obtained in step 3;
- 5) Output the inferred group class labels $\mathbf{Z}$ for $\mathcal{G}$, and the parameter of the predictive distribution $\mathbf{w}$.

Now, we provide details of computing the posterior distribution $q(\mathbf{Z}|\Phi)$ in step 2 for collusion inference, and estimating the parameter $\mathbf{w}$ of the predictive distribution in step 4 for collusion prediction respectively.

First, we compute the posterior distribution $q(\mathbf{Z}|\Phi)$ according to the generative view of LCM discussed in Section 5.3.1.1. In particular, the likelihood of generating the observed data Φ based on the model parameter $\lambda = \{\lambda_0^{(m)}\}_{m=1}^M \cup \{\lambda_1^{(m)}\}_{m=1}^M$, $\lambda_{(\cdot)}^{(m)} = [\alpha_{(\cdot)}^{(m)}, \beta_{(\cdot)}^{(m)}]^T$ is given in Eq. 5.10. As $\mathbf{Z}$ is latent in the model, we develop an EM algorithm to perform parameter estimation and posterior inference. Specifically, based on Eq. 5.10, the complete data log-likelihood of this generative model can be written as,

$$\begin{aligned} \mathcal{L}_{\text{collusion_inference}}(\lambda|\Phi, \mathbf{Z}) = & \sum_{n=1}^N z_n \left[\log p_1 + \log \sum_{m=1}^M \text{Beta}(\phi_n^{(m)} | \alpha_1^{(m)}, \beta_1^{(m)}) \right] \\ & + \sum_{n=1}^N (1 - z_n) \left[\log p_0 + \log \sum_{m=1}^M \text{Beta}(\phi_n^{(m)} | \alpha_0^{(m)}, \beta_0^{(m)}) \right] \end{aligned} \quad (\text{Eq. 5.13})$$

Thus, in the E-step, we evaluate the posterior distribution of z_n given the current parameter values as,

$$\begin{aligned} q(z_n = 1) &= \mathbb{E}[z_n] = \frac{p_1 \prod_{m=1}^M \mathbf{Beta}(\phi_n^{(m)} | \alpha_1^{(m)}, \beta_1^{(m)})}{\sum_{k \in \{0,1\}} p_k \prod_{m=1}^M \mathbf{Beta}(\phi_n^{(m)} | \alpha_k^{(m)}, \beta_k^{(m)})} \\ q(z_n = 0) &= 1 - q(z_n = 1) \end{aligned} \quad (\text{Eq. 5.14})$$

In the M-step, we estimate the parameters of the model based on $\{q(z_n)\}_{n=1}^N$ by maximizing the log-likelihood. In particular, the update equation for the cluster membership distribution $p_z, z \in \{0, 1\}$ is,

$$p_1 = \frac{1}{N} \sum_{n=1}^N q(z_n = 1), \quad p_0 = \frac{1}{N} \sum_{n=1}^N q(z_n = 0) \quad (\text{Eq. 5.15})$$

For the parameter $\lambda = \{\alpha_k^{(m)}, \beta_k^{(m)}\}_{m=1}^M, k \in \{0, 1\}$, the optimal value $\{\alpha_k^{(m)*}, \beta_k^{(m)*}\}$ for each parameter pair are coupled and need to be solved simultaneously. We resort to the Newton-Raphson method where each Beta parameter pair $\lambda_k^{(m)} = \{\alpha_k^{(m)}, \beta_k^{(m)}\}$ can be optimized iteratively:

$$\lambda_k^{(m)\text{new}} = \lambda_k^{(m)\text{old}} - \mathbf{H}_{\lambda_k}^{-1} \mathbf{g}_{\lambda_k} \quad (\text{Eq. 5.16})$$

with gradients $\mathbf{g}_{\lambda_k} = [g_{\lambda_k}^\alpha, g_{\lambda_k}^\beta]$:

$$\begin{aligned} g_{\lambda_k}^\alpha &= \sum_n q(z_n = k) \{\log \phi_n^{(m)} - [\psi(\alpha_k^{(m)}) - \psi(\alpha_k^{(m)} + \beta_k^{(m)})]\} \\ g_{\lambda_k}^\beta &= \sum_n q(z_n = k) \{\log(1 - \phi_n^{(m)}) - \psi(\beta_k^{(m)}) - \psi(\alpha_k^{(m)} + \beta_k^{(m)})]\} \end{aligned} \quad (\text{Eq. 5.17})$$

and Hessian $\mathbf{H}_{\lambda_k}$:

$$\mathbf{H}_{\lambda_k} = \sum_n q(z_n = k) \cdot \begin{bmatrix} \psi'(\alpha_k^{(m)} + \beta_k^{(m)}) - \psi'(\alpha_k^{(m)}) & \psi'(\alpha_k^{(m)} + \beta_k^{(m)}) \\ \psi'(\alpha_k^{(m)} + \beta_k^{(m)}) & \psi'(\alpha_k^{(m)} + \beta_k^{(m)}) - \psi'(\beta_k^{(m)}) \end{bmatrix} \quad (\text{Eq. 5.18})$$

where $\psi(\cdot)$ is the *digamma* function; $\psi'(\cdot)$ the *trigamma* function. After the EM algorithm converges, we obtain the posterior distribution of the class labels and further the pseudo class label $\hat{\mathbf{Z}} = \{\hat{z}_n\}$ according to step 3.

Next, given the pseudo group class labels $\hat{\mathbf{Z}}$, we estimate the parameter $\mathbf{w}$ of the predictive distribution for collusion prediction. In particular, the log-likelihood of the discriminative view is given by,

$$\mathcal{L}_{\text{collusion_prediction}} = \sum_{n=1}^N \hat{z}_n \log \sigma(\mathbf{w}^T \phi_n) + (1 - \hat{z}_n) \log(1 - \sigma(\mathbf{w}^T \phi_n)) \quad (\text{Eq. 5.19})$$

Due to the nonlinearity of logistic function, the optimal solution cannot be found analytically. Again, we appeal to the Newton-Raphson method where the optimal $\mathbf{w}^*$ can be estimated iteratively:

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} - \mathbf{H}_{\mathbf{w}}^{-1} g_{\mathbf{w}} \quad (\text{Eq. 5.20})$$

with gradient and Hessian given by,

$$\begin{aligned} g_{\mathbf{w}} &= \sum_{n=1}^N (\sigma(\mathbf{w}^{\text{old}T} \phi_n) - \hat{z}_n) \phi_n \\ \mathbf{H}_{\mathbf{w}} &= \sum_{n=1}^N \sigma(\mathbf{w}^{\text{old}T} \phi_n) (1 - \sigma(\mathbf{w}^{\text{old}T} \phi_n)) \phi_n \phi_n^T \end{aligned} \quad (\text{Eq. 5.21})$$

5.3.3 The Unified Model

In this subsection, we introduce the second model variant that utilizes the two-view perspective for collusive fraud detection. In fact, a more coherent and principled approach for solving Problem 1 would be to combine the generative and discriminative views into a unified framework, such that the class labels and the predictive distribution can be obtained at the same time.

Although possessing the ability to take advantage of unlabeled data for collusion inference, the generative view of LCM suffers from the common problem of a generative probabilistic model that the performance would deteriorate if the generative model were not a perfect match for the true data generation process in reality [26]. When it comes to the chaining-based model described above, the inaccuracy of the generative

view modeling would subsequently affect the performance of the collusion prediction procedure which takes the inferred cluster labels from the generative view as input. On the other hand, by directly modeling the dependence of the class labels on the data, i.e., $p(\mathbf{Z}|\Phi)$, the discriminative view of LCM would achieve better results when a modeling mismatch exists. However, due to the fact that the discriminative view ignores the structure $p(\Phi|\mathbf{Z})$, it cannot effectively learn the colluder behavioral pattern from the data, which can be captured by the distribution of the Beta parameter λ in Eq. 5.13 introduced in the generative view of LCM.

Therefore, to benefit from the merits of both generative and discriminative views, we would expect that an optimal trade-off can be found by combining the two views in a concurrent and principled way. In particular, we design a hybrid of generative and discriminative models according to the principle of blending generative and discriminative models suggested by [29]. The graphical model representation of the unified LCM is depicted in Figure 5.1. The idea is to model the joint probability of all the h-CBM vectors Φ and the class labels $\mathbf{Z}$ given the model parameters $\{\mathbf{w}, \lambda\}$ is written as follows:

$$\begin{aligned}
 p(\Phi, \mathbf{Z}|\mathbf{w}, \lambda) &= \prod_{n=1}^N p(z_n|\phi_n, \mathbf{w}) \prod_{m=1}^M p(\phi_n^{(m)}|\lambda_{z_n}^{(m)}) \\
 &= \prod_{n=1}^N \left[\frac{1}{1 + \exp(-\mathbf{w}^T \phi_n)} \right]^{z_n} \left[\frac{\exp(-\mathbf{w}^T \phi_n)}{1 + \exp(-\mathbf{w}^T \phi_n)} \right]^{1-z_n} \\
 &\quad \cdot \left(\prod_{m=1}^M \left[\frac{\Gamma(\alpha_1^{(m)} + \beta_1^{(m)})}{\Gamma(\alpha_1^{(m)})\Gamma(\beta_1^{(m)})} (\phi_n^{(m)})^{\alpha_1^{(m)}-1} (1 - \phi_n^{(m)})^{\beta_1^{(m)}-1} \right]^{z_n} \right. \\
 &\quad \cdot \left. \left[\frac{\Gamma(\alpha_0^{(m)} + \beta_0^{(m)})}{\Gamma(\alpha_0^{(m)})\Gamma(\beta_0^{(m)})} (\phi_n^{(m)})^{\alpha_0^{(m)}-1} (1 - \phi_n^{(m)})^{\beta_0^{(m)}-1} \right]^{1-z_n} \right)
 \end{aligned} \tag{Eq. 5.22}$$

We can see that compared to the model specification in the generative view (Eq. 5.10), the cluster membership distribution p_z is replaced by the discriminative classification distribution $p(z_n|\phi_n, \mathbf{w})$ parametrized by the predictive distribution parameter $\mathbf{w}$. This

replacement thus connects the two views of modeling in a way that the estimation of the discriminative model parameter $\mathbf{w}$ and generative model parameter $\boldsymbol{\lambda}$ will affect each other through the inference about the latent class label variable $\mathbf{Z}$ during model optimization, which we will show in the subsequent section.

5.3.3.1 Learning and Inference

The model optimization involves the computation of the posterior distribution of latent variables and the estimation of the model parameters. In the unified LCM, the latent variables $\mathbf{Z} = \{z_n\}_{n=1}^N$ account for the occurrence of collusion in the input data. The

Figure 5.1: Graphical model representation of unified LCM. Light circles denote the latent random variables $\{z_n\}_{n=1}^N$ and model parameters $\{\mathbf{w}, \boldsymbol{\lambda}\}$, and dark circles denote the observed random variables $\{\phi_n\}_{n=1}^N$. The left plate indexed by N denotes the groups in $\mathcal{U}$, and the right plate indexed by “2” denotes the h-CBM distributions for colluder ($k = 1$) and non-colluder ($k = 0$) groups in $\mathcal{U}$.

model parameters are $\{\mathbf{w}, \boldsymbol{\lambda}\}$ where $\mathbf{w}$ enables the learning of a collusion predictor, and $\boldsymbol{\lambda} = \{\boldsymbol{\lambda}_0^{(m)}\}_{m=1}^M \cup \{\boldsymbol{\lambda}_1^{(m)}\}_{m=1}^M$ captures the collective behavior patterns of colluders and non-colluders. Again, as the unified LCM is a latent variable model, we develop an EM algorithm to perform model learning and inference. Specifically, we first derive the complete data log-likelihood of the model by taking the logarithm of Eq. 5.22:

$$\begin{aligned}
 \mathcal{L}_{\text{unified}}(\mathbf{w}, \boldsymbol{\lambda} | \Phi, \mathbf{Z}) &= \log p(\Phi, \mathbf{Z} | \mathbf{w}, \boldsymbol{\lambda}) \\
 &= \sum_{n=1}^N z_n \log \sigma(\mathbf{w}^T \phi_n) + (1 - z_n) \log(1 - \sigma(\mathbf{w}^T \phi_n)) \\
 &\quad + \sum_{n=1}^N z_n \log \sum_{m=1}^M \text{Beta}(\phi_n^{(m)} | \alpha_1^{(m)}, \beta_1^{(m)}) \\
 &\quad + \sum_{n=1}^N (1 - z_n) \log \sum_{m=1}^M \text{Beta}(\phi_n^{(m)} | \alpha_0^{(m)}, \beta_0^{(m)})
 \end{aligned} \tag{Eq. 5.23}$$

E-step: We derive the posterior of latent variable $\mathbf{Z}$ based on Bayes' theorem and Eq. 5.22 as follows:

$$\begin{aligned}
 q(\mathbf{Z} | \Phi, \mathbf{w}, \boldsymbol{\lambda}) &\propto p(\Phi, \mathbf{Z} | \mathbf{w}, \boldsymbol{\lambda}) \\
 &= \prod_{n=1}^N \left[\sigma(\mathbf{w}^T \phi_n) \prod_{m=1}^M \text{Beta}(\phi_n^{(m)} | \alpha_1^{(m)}, \beta_1^{(m)}) \right]^{z_n} \\
 &\quad \cdot \left[(1 - \sigma(\mathbf{w}^T \phi_n)) \prod_{m=1}^M \text{Beta}(\phi_n^{(m)} | \alpha_0^{(m)}, \beta_0^{(m)}) \right]^{1-z_n}
 \end{aligned} \tag{Eq. 5.24}$$

For simplicity, it is assumed that each group is independent with each other and so is its label. Then, by factorizing Eq. 5.24, we obtain the posterior of each z_n as:

$$q(z_n = 1 | \phi_n, \mathbf{w}, \boldsymbol{\lambda}) = \sigma(\mathbf{w}^T \phi_n + \Delta_{\boldsymbol{\lambda}}) \tag{Eq. 5.25}$$

with $\Delta_{\boldsymbol{\lambda}}$ defined as:

$$\begin{aligned}
 \Delta_{\boldsymbol{\lambda}} &= \sum_{m=1}^M \log \Gamma(\alpha_1^{(m)} + \beta_1^{(m)}) + \log \Gamma(\alpha_0^{(m)}) + \log \Gamma(\beta_0^{(m)}) \\
 &\quad - \log \Gamma(\alpha_0^{(m)} + \beta_0^{(m)}) - \log \Gamma(\alpha_1^{(m)}) - \log \Gamma(\beta_1^{(m)}) \\
 &\quad + (\alpha_1^{(m)} - \alpha_0^{(m)}) \log \phi_n^{(m)} + (\beta_1^{(m)} - \beta_0^{(m)}) \log(1 - \phi_n^{(m)})
 \end{aligned}$$

We can see that in Eq. 5.25 the inference about the class labels $\mathbf{Z} = \{z_n\}$ requires the knowledge about the generative parameter $\boldsymbol{\lambda}$ and the discriminative parameter $\mathbf{w}$.

M-step: We find the estimate for model parameters $\{\mathbf{w}, \boldsymbol{\lambda}\}$ so as to maximize the expected value of the complete data log-likelihood $\mathbb{E}_{\mathbf{Z}}[\log p(\boldsymbol{\Phi}, \mathbf{Z} | \mathbf{w}, \boldsymbol{\lambda})]$.

For estimating the discriminative parameter $\mathbf{w}$, similar to the case in the chaining-based model, we appeal to the Newton-Raphson method where the optimal $\mathbf{w}^*$ can be estimated iteratively:

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} - \mathbf{H}_{\mathbf{w}}^{-1} g_{\mathbf{w}} \quad (\text{Eq. 5.26})$$

with gradient $g_{\mathbf{w}} = \boldsymbol{\Phi}^T (\mathbf{q}^{\bar{z}} - \mathbf{p}^{\bar{z}})$ and Hessian $\mathbf{H}_{\mathbf{w}} = \boldsymbol{\Phi}^T \mathbf{Q} \boldsymbol{\Phi}$, where $\mathbf{q}^{\bar{z}}$ is a column vector $\{\mathbb{E}_q[z_n]\}_{n=1}^N$; $\mathbf{p}^{\bar{z}}$ a column vector $\{\mathbb{E}_p[z_n]\}_{n=1}^N$; $\mathbf{Q}$ an $N \times N$ diagonal matrix with elements $Q_{nn} = \mathbb{E}_q[z_n](1 - \mathbb{E}_q[z_n])$. Note that different from the estimation of $\mathbf{w}$ in the chaining-based model in Eq. 5.20, the computation of gradient and Hessian here requires the expectation of the class labels $\{z_n\}$ with respect to their posterior distributions $\{q(z_n)\}$, while in Eq. 5.20 the hard assignments of the class labels are used instead.

For estimating generative parameter $\boldsymbol{\lambda} = \{\alpha_k^{(m)}, \beta_k^{(m)}\}_{m=1}^M$, with $k \in \{0, 1\}$, the optimal value $\{\alpha_k^{(m)*}, \beta_k^{(m)*}\}$ for each parameter pair are coupled and need to be solved simultaneously. Similarly, the Newton-Raphson method is used where each Beta parameter pair $\lambda_k^{(m)} = \{\alpha_k^{(m)}, \beta_k^{(m)}\}$ can be optimized iteratively:

$$\lambda_k^{(m)\text{new}} = \lambda_k^{(m)\text{old}} - \mathbf{H}_{\lambda_k}^{-1} \mathbf{g}_{\lambda_k} \quad (\text{Eq. 5.27})$$

where gradients $\mathbf{g}_{\lambda_k} = [g_{\lambda_k}^{\alpha}, g_{\lambda_k}^{\beta}]$:

$$\begin{aligned} g_{\lambda_k}^{\alpha} &= \sum_n q(z_n = k) \{ \log \phi_n^{(m)} - [\psi(\alpha_k^{(m)}) - \psi(\alpha_k^{(m)} + \beta_k^{(m)})] \} \\ g_{\lambda_k}^{\beta} &= \sum_n q(z_n = k) \{ \log(1 - \phi_n^{(m)}) - \psi(\beta_k^{(m)}) - \psi(\alpha_k^{(m)} + \beta_k^{(m)}) \} \end{aligned} \quad (\text{Eq. 5.28})$$

and Hessian $\mathbf{H}_{\lambda_k}$:

$$\mathbf{H}_{\lambda_k} = \sum_n q(z_n = k) \cdot \begin{bmatrix} \psi'(\alpha_k^{(m)} + \beta_k^{(m)}) - \psi'(\alpha_k^{(m)}) & \psi'(\alpha_k^{(m)} + \beta_k^{(m)}) \\ \psi'(\alpha_k^{(m)} + \beta_k^{(m)}) & \psi'(\alpha_k^{(m)} + \beta_k^{(m)}) - \psi'(\beta_k^{(m)}) \end{bmatrix} \quad (\text{Eq. 5.29})$$

where $\psi(\cdot)$ is the *digamma* function; $\psi'(\cdot)$ the *trigamma* function. We can see that the forms of Eq. 5.28 and Eq. 5.29 are the same as that in the chaining-based model (Eq. 5.17 and Eq. 5.18), however, the posterior distribution of the class labels $q(z_n)$ used is different; here in the unified LCM, the class label posterior distribution $q(z_n)$ is computed based on the knowledge of the discriminative parameter $\mathbf{w}$ (Eq. 5.25).

Finally, the full EM algorithm proceeds as follows. First, we initialize $\{\boldsymbol{\lambda}_k^{(m)}\}$ with uninformative prior $[1, 1]$, and $\mathbf{w}$ randomly. Then the algorithm is performed iteratively by alternatively executing the *E-step* and *M-step* in each iteration until the log-likelihood $\log p(\Phi, \mathbf{Z} | \mathbf{w}, \boldsymbol{\lambda})$ converges.

5.3.4 Collusion Inference and Prediction

After performing model learning and inference for LCM (both chaining-based model and the unified model), we can finally solve Problem 1.

For the collusion inference task, in which the goal is to infer the class labels of the candidate groups given by the data, we can obtain the probability of each group g_n being a colluder group ($p(z_n = 1)$) or a non-colluder group ($p(z_n = 0)$) by computing the posterior distribution over its class label z_n , which is readily given by Eq. 5.14 and Eq. 5.25:

$$p(z_n = 1) = q(z_n = 1 | \phi_n, \mathbf{w}^*, \boldsymbol{\lambda}^*) = \sigma(\mathbf{w}^{*T} \phi_n + \Delta_{\boldsymbol{\lambda}^*}) \quad (\text{Eq. 5.30})$$

$$p(z_n = 0) = q(z_n = 0 | \phi_n, \mathbf{w}^*, \boldsymbol{\lambda}^*) = 1 - q(z_n = 1 | \phi_n, \mathbf{w}^*, \boldsymbol{\lambda}^*)$$

where $\mathbf{w}^*$ and $\boldsymbol{\lambda}^*$ are the (local) optimal solution obtained after the EM algorithm converges.

For the collusion prediction task, based on $\mathbf{w}^*$ and Eq. 5.12, the collusion predictive

distribution can be derived as:

$$\begin{aligned}
 p(\hat{z} = 1 | \hat{\phi}, \Phi, \mathbf{Z}) &= \int p(\hat{z} = 1 | \hat{\phi}, \mathbf{w}) p(\mathbf{w} | \Phi, \mathbf{Z}) d\mathbf{w} \\
 &\approx \int p(\hat{z} = 1 | \hat{\phi}, \mathbf{w}^*) p(\mathbf{w} | \Phi, \mathbf{Z}) d\mathbf{w} \\
 &= p(\hat{z} = 1 | \hat{\phi}, \mathbf{w}^*) = \sigma(\mathbf{w}^{*T} \phi_n)
 \end{aligned} \tag{Eq. 5.31}$$

5.4 Experimental Analysis

In this section, we conduct a series of experiments based on two real-world datasets collected from popular review sites to evaluate the predictive and inference performance of the proposed LCM, and compare with state-of-the-art collusion detection algorithms.

5.4.1 Datasets

Our experiments are conducted on the same datasets as used in the previous chapters, i.e., AmazonCn and YelpZip, which contain the golden standard of collusive reputation fraud in the wild.

5.4.2 Effectiveness of h-CBMs

In this section, we evaluate the effectiveness of the proposed h-CBMs that are designed to capture the behavioral connections between colluders. In particular, we take a classification evaluation approach to assess the discriminating ability of each h-CBM, by training a classifier for each *individual* h-CBM on the datasets. The classifier we use here is Logistic Regression. The standard classification metric - Area under ROC curve (AUC) - is employed. AUC score is commonly used to evaluate a binary classifier by inspecting the true positive rate against the false positive rate at different discrimination thresholds. 5-fold cross-validation is used to conduct the experiment. The results are shown in Figure 5.2.

Figure 5.2: Classification performance (AUC) of h-CBMs on AmazonCn and YelpZip.

In general, it can be seen that all h-CBMs are useful in characterizing the collective behaviors of colluders, as all of them attain $AUC > 0.5$ (0.629 precisely). On AmazonCn dataset, the best result is achieved by rating consistency (RC), implying that a strong agreement is reached among the colluders on the sentimental disposal of the targets in a particular fraud campaign. Temporal-based h-CBMs (i.e., temporal synchronization (TS), review order synchronization (ROS), and first-review synchronization (FS)) also work well, which explains the bursty of typical review fraud campaigns; the involved colluders are coordinated and thus synchronized as each one of them is a part of the collaboration. Finally, activity-based h-CBMs (i.e., account lifetime similarity (ALS), workload similarity (WS), and activity consistency (AC)) show their strengths, by capturing the resemblance of the account usage patterns of colluders. In particular, the promising result achieved by activity consistency (AC) suggests a new angle to quantify the connections between colluders by relating their most active moments.

On YelpZip dataset, we can observe a different behavioral distribution. Here the best performance is achieved by target consistency (TC), showing that the colluders' accounts on Yelp are used almost exclusively for the designated targets. Again, rating and temporal based h-CBMs outperform activity-based h-CBMs, suggesting that colluders' actions are

typically bounded by the campaigns they engage in, such that the similarity between their actions in terms of rating and time would become more distinctive. Finally, we see that the h-CBMs generally perform better on Amazon than on Yelp. This might indicate that the fraud campaigns on Yelp are organized in a looser manner in the sense that the desired fake reviews are spread across a longer period of time, and the usage patterns of the involved accounts are more diverse. In addition, this difference also shows that the behaviors of colluders are indeed quite different on different domains (i.e., manufacture products vs. restaurants). Nonetheless, as being the members of fraud campaigns, colluders would necessarily establish certain behavioral connections incurred due to the common campaign goals and activities, and can effectively be captured by the proposed h-CBMs.

5.4.3 Prediction Performance

In this section, we evaluate the predictive performance of LCM and make comparison with state-of-the-art *unsupervised* collusive fraud detectors. As the output of LCM is a list of probabilities of being colluders for each user in the dataset, for evaluation protocol, we employ two well-known ranking based metrics, namely Average Precision (AP) and Area Under ROC Curve (AUC). In summary, we have the following baselines for the comparison analysis:

- GSRank: an unsupervised ranking algorithm [37] proposed to detect colluder groups. It works by performing iterative computations on three related entities, i.e., users, user groups, and products. In addition, eight Group Spam Behavior Indicators (GSBIs) are proposed to capture suspicious group behaviors of colluders. In particular, GSBIs include features that investigate the *external* properties of a candidate group, such as *Group Size* and *Group Support Count* mentioned in Section 5.1 that

produce measurements of a whole group. Different from GSBIs, the proposed h-CBMs in this work focus on the *internal* properties of a candidate group. Thus, even though a colluder group can restrict its size or overwhelm the targets, the actions of its members towards the targets are inevitably homogeneous, which renders it detectable. For the experiments to be performed, as GSRank is tightly coupled with the GSBIs and cannot freely incorporate other features, we will evaluate the performance of GSRank combined with GSBIs only.

- Learning to rank: another competitor capable of unsupervised training is the learning to rank approach. This approach is traditionally used in supervised learning to learn an optimal combination of a list of training rankings. It, however, can be adapted to detect reputation fraud in unsupervised settings through proper extension [37, 39]. The insight is that when the value of a feature is semantically meaningful, for example, the larger the value, the more suspicious the sample is, a valid training ranking can be generated by sorting the samples based on their feature values in descending order. In our case, we obtain eight training rankings, each of which corresponds to one dimension of the h-CBM vectors. Two well-known learning to rank algorithms - SVMRank [25] and RankBoost [17] - are included as baselines. In the experiments, the two algorithms will be performed with GSBIs, h-CBMs, and both.
- LCM Chain (Chain for short): for the chain-based LCM model, we use Beta mixture model for collusion inference and Logistic Regression for collusion prediction respectively. As a flexible model that can work with any number of features, Chain will be performed with GSBIs, h-CBMs, and both.
- LCM Unified (Unified for short): similar to the LCM Chain model, LCM Unified will be performed with GSBIs, h-CBMs, and both.

The output of each baseline is a score assigned to each group representing the possibility of being a colluder group. In LCM, the score for each group is $p(\hat{z}|\hat{\phi}, \Phi, \mathbf{Z})$ as given by Eq. 5.31. The experiment is conducted using 5-fold cross-validation. For GSRank, as it is not a trainable model, the evaluation is performed based on one test folder each time. Our implementations of the EM algorithms for LCM converge in less than 18 iterations given that the algorithms terminate when the likelihood difference between consecutive iterations falls below 10^{-6} . The results are shown in Figure 5.3. In general, we can see that the proposed LCM (both Chain and Unified) achieves better performance than the second-place GSRank in both metrics on the two datasets. For LCM, the two variants, i.e., Chain and Unified, are observed to compliment each other when different metrics are used.

Figure 5.3: Collusion prediction performance (in Average Precision and AUC) of LCM compared with GSRank on AmazonCn and YelpZip datasets.

On AmazonCn dataset as shown in Figure 5.3.a, it turns out that Unified generally outperforms Chain, with the best result achieved by using h-CBMs, i.e., for Unified+h-CBMs, we have AP=0.857 and AUC=0.835. For the feature performance comparison, we inspect the experiments that use the same models. It shows that in all cases the performance can be improved by incorporating h-CBMs for detection (by comparing X+GSBIs

with X+ALL, with “X” representing the model used). The improvements range from 3.3% to 7.3% in AP and 3.9% to 7.6% in AUC. Also, the h-CBMs are shown to perform better than GSBIs (by comparing X+GSBIs with X+h-CBMs), by a margin of 3.5%-8.2% in AP and 4.1%-8.5% in AUC, which suggests the effectiveness of characterizing colluders’ homogeneous behaviors in practice. Then, to compare the performance of different models, we inspect the experiments with the feature set fixed (i.e., X+GSBIs). It can be seen that the proposed LCM (both Unified and Chain) exhibits superior performance over the second-place GSRank, with the improvements by 2.3%-4.1% in AP and 0.3%-1.5% in AUC.

On YelpZip dataset as shown in Figure 5.3.b, we can observe similar trends in terms of model and feature improvements. In particular, for the feature comparison, we compare the experiments X+h-CBMs and X+GSBIs with “X” being LCM Chain, LCM Unified, SVMRank, and RankBoost. It can be seen that h-CBMs generally achieve better results than GSBIs, with the improvements ranging from 7.6% to 14.4% in AP and 4.7% to 8.5% in AUC. For the model comparison, we see that the performance of GSRank+GSBIs is much worse than its performance on AmazonCn dataset. This might be ascribed to the fact that GSRank, as a power iteration algorithm, can amplify errors during propagation, which would dramatically deteriorate the performance when the used features are incapable of capturing the behaviors of colluders. In contrast, LCM is more robust to less effective features by modeling the joint probability of the class labels and feature values of candidate groups.

Finally, we compare the two LCM variants, which are observed to compliment each other in terms of ranking (AP) and classification (AUC) performance. Specifically, in terms of ranking, by fixing feature sets (i.e., comparison between X+GSBIs, X+h-CBMs, and X+ALL respectively, with “X” being Unified or Chain), we can see that Unified outperforms Chain on both datasets with a margin from 1.9%-2.5% on AmazonCn and

11.7%-14.4% on YelpZip in AP. However, in terms of AUC score, the superiority is reversed. Since AUC considers true negatives (i.e., true normal users) when computing the false positive rate and both of the datasets are slightly imbalanced with many more negatives (normal users) than positives (colluders), the superiority of Chain over Unified in AUC might indicate that Chain is more suitable for imbalance classification task in which both colluders and normal users are of interest. This can be partially due to the fact that in Unified, the probabilities of the latent group class labels and the predictive weights are jointly modeled, which would cause the data used in the model to account for the estimation of both objects at the same time. In contrast, the data in Chain are fully used for the collusion inference task, which can lead to more accuracy results since more data are exploited to estimate the group class labels alone.

5.4.4 Inference Performance

In this section, we evaluate the collusion inference performance of the proposed LCM for collusive fraud detection. Note that in LCM, the inference task is performed by computing the posterior distribution of the group class labels according to Eq. 5.30. Different from the collusion prediction task discussed before, here the experiment can be conducted and evaluated on the full datasets. Note that the learning to rank algorithms (i.e., SVMRank and RankBoost) cannot perform the collusion inference task as they are supervised models, thus they are not included in this comparison. Figure 5.4 shows the comparison results between LCM and GSRank in AP and AUC on the two evaluated datasets.

The first observation we make is that on AmazonCn dataset where a number of large colluder groups exist, by utilizing the full dataset for inference, GSRank improves its performance by attaining a slightly higher AP (0.815) than LCM+GSBIs (i.e., Chain+GSBIs and Unified+GSBIs). However, the AUC score of GSRank turns out to be significantly

Figure 5.4: Collusion inference performance (in Average Precision and AUC) of LCM compared with GSRank on AmazonCn and YelpZip datasets.

lower, suggesting that GSRank still cannot properly distinguish all colluders from non-colluders on AmazonCn dataset. This can be ascribed to the fact that apart from large groups, there are also a non-trivial number of small colluder groups whose behaviors are stealthier and cannot be effectively captured by the GSBIs that look at the external properties of candidate groups (e.g., group size and target set size) that are more vulnerable to manipulation. In contrast, the performance of LCM Chain and Unified achieve better AUC scores, with the help of the h-CBMs that capture the similarity between colluders rather than their aggregated characteristics.

Second, by comparing the inference performance of the two variants of LCM, similar trends are observed in terms of their ranking and classification abilities as those in the collusion prediction comparison. Again, on both datasets, LCM Unified achieves higher AP score than Chain when working with each of the evaluated feature sets, with the improvements ranging from 1.4% to 2.2% on AmazonCn and 1.0% to 5.6% on YelpZip. On the other hand, in terms of AUC performance, LCM Chain dominates the results on both datasets. This again, as discussed in the previous subsection, shows that LCM Chain would be more suitable for binary classification task on imbalanced datasets where both colluders and non-colluders are of interest, while LCM Unified would be appropriate for

ranking-based retrieval scenarios where the goal is to rank colluders as highly as possible in the obtained results.

5.4.5 Learned Collusive Behavioral Patterns

In addition to performing collusion prediction and inference tasks, LCM can also learn the behavior patterns of colluders and non-colluders, which are modeled as Beta distributions introduced in the generative view of LCM (Section 5.3.1.1). As mentioned, the behavioral (Beta) distributions of colluders and non-colluders are parametrized by the Beta parameters $\lambda_1 = \{\lambda_1^{(m)}\}$ and $\lambda_0 = \{\lambda_0^{(m)}\}$ respectively, where each $(\lambda_1^{(m)}, \lambda_0^{(m)})$ pair corresponds to the distributions of the m th h-CBM dimension over colluder and non-colluder clusters respectively.

Figure 5.5 shows the behavioral distributions learned by LCM Unified for colluders and non-colluders on the two datasets. We omit the results of LCM Chain as the learned distributions are similar. In Figure 5.5, each sub-plot shows the learned two Beta distributions parametrized by $\lambda_1^{(m)} = \{\alpha_1^{(m)}, \beta_1^{(m)}\}$ and $\lambda_0^{(m)} = \{\alpha_0^{(m)}, \beta_0^{(m)}\}$ on the m th observed h-CBM dimension $\phi_{(m)} \in \{\text{TC, RC, TS, ROS, FS, ALS, AC, WS}\}$. In particular, the distributions for colluders and non-colluders are shown in red (solid) lines and blue (dotted) lines. The expected values of the distributions are shown as horizontal lines. The x -axis is the domain of Beta distribution which is also the range of the corresponding h-CBMs, with values close to 1 indicating more likely to be related to collusive fraud.

In general, from Figure 5.5 we can observe clear deviations between the learned behavioral distributions of colluders and non-colluders in all sub-plots, where the densities of colluders put more mass towards the extreme right of each sub-plot, showing that the behaviors of colluders are more closely related to collusive fraud that is characterized by the proposed h-CBMs. This not only validates the hypothesis that the behaviors

5.5.a: LCM Unified on AmazonCn dataset

5.5.b: LCM Unified on YelpZip dataset

Figure 5.5: The learned collusive behavioral patterns on AmazonCn and YelpZip datasets. Each behavioral pattern is modeled as a Beta density function (PDF) and corresponds to each h-CBM dimension $\phi \in \{\text{TC, RC, TS, ROS, FS, ALS, AC, WS}\}$. The density functions for colluders and non-colluders are shown in red/solid lines and blue/dotted lines respectively. The expected values for each learned behavioral distributions are plotted as horizontal lines and also reported in the plot captions.

of colluders and non-colluders are different and can be separable in appropriate feature space, but also shows the effectiveness of LCM in learning the separated distributions. In addition, the deviation degrees of the two distributions (i.e., for colluders and non-colluders) can also to some extent reflect the relative strengths of the features used. For example, on AmazonCn dataset in Figure 5.5.a, we see that the distribution gaps are larger on temporal synchronization (TS), first-review (FS), account lifetime similarity (ALS), and activity consistency (AC) dimensions than the rest, which is generally consistent with the results of h-CBM evaluation in Section 5.4.2. Similar observations are made on target consistency (TC) and rating consistency (RC) dimensions for YelpZip dataset. It is worth noting that the results here do not perfectly match those in h-CBM evaluation because in Section 5.4.2 each h-CBM is evaluated independently, while the shown behavioral parameters λ here are learned jointly in LCM such that a balance is made over all h-CBM dimensions to obtain the maximum model likelihood and optimal group class label posteriors.

5.5 Summary

In this chapter, we identify the problem of detecting collusive reputation fraud in crowdsourced opinions from a stochastic perspective. In particular, we propose a novel statistical model, named as Latent Collusion Model (LCM), to model collusive reputation fraud from a hybrid of generative and discriminative perspectives. Not only can LCM perform collusion inference as unsupervised models, but it can also make collusion predictions as supervised models. Two variants of LCM are studied in details. The first is a chaining-based model which performs collusion inference and prediction in a sequential manner. The chaining-based model is flexible in the sense that models suitable for the respective tasks can be optimized and applied separately. Moreover, the experiment results show that it works empirically well on classifying colluder and non-colluders in

scenarios where data are imbalanced. The second is a unified model that combines the learning and inference of the generative and discriminative views, so as to achieve the optimal results via balancing between the two views. The experiment results show its superiority over the chaining-based model in colluder ranking ability. Furthermore, a suite of homogeneity-based collusive behavior measures (h-CBMs) are proposed to capture the homophily inside colluder groups. The h-CBMs are complementary to existing collusion-oriented features in handling stealthier collusive attacks. In summary, our results on two real-world datasets show the effectiveness of h-CBMs and the superiority of LCM over state-of-the-art competitors in terms of collusion inference and predictive abilities.

Chapter 6

FraudScan: An Incremental Framework for Reputation Fraud Campaign Detection

In this chapter, we address the online detection of collusive reputation fraud in crowd-sourced opinions. As mentioned, the major form of collusive reputation fraud is the reputation fraud campaigns (RFCs) which aim at distorting the reputations of evaluated items, by generating fake opinions (e.g., rating stars) through multiple spammers. Most of the existing approaches to reputation fraud campaign detection rely on characterizing their collective behaviors based on rating histories. However, these campaigns are constantly evolving and changing tactics to evade detection. For example, they can launch early attacks on the items to quickly dominate the reputations. They can also whitewash themselves through creating new accounts for subsequent attacks. It is thus challenging for existing approaches working on historical data to promptly react to such emerging fraud activities. In order to quickly respond to newly-emerging campaigns, we conduct RFC detection in online fashion. We first formulate the task of online reputation fraud campaign detection in Section 6.1. Then, a unified and scalable optimization framework has been proposed which can adapt built detection model to emerging fraud patterns over time (Section 6.2). Finally, empirical analysis on real-world datasets has been conducted to validate the effectiveness and efficiency of the proposed framework (Section 6.3).

6.1 Problem Formulation and Challenges

In this section, we introduce the notations and definitions related to the studied problem. To adapt built detection model to constantly changing campaign strategies, we formulate the task of online reputation fraud campaign detection and identify the challenges of fulfilling this task.

Assume that there is a set of n items $\mathcal{V} = \{v_j\}_{j=1}^n$ receiving ratings from a set of m users $\mathcal{U} = \{u_i\}_{i=1}^m$. A typical item rating consists of four fundamental elements: a *user id* specifying the subject giving the rating, an *item id* referring to the rated item, a numerical *star* instantiating the rating, and a *timestamp* recording the time of the rating. We use a 4-tuple $r_{ij} = \langle u_i, e_{ij}, v_j, t_{ij} \rangle$ to represent an item rating r_{ij} generated by user u_i for item v_j with star e_{ij} at time t_{ij} .

Definition 6. *Along the time dimension, we can observe a **rating sequence** comprised of an unbounded series of incoming ratings in temporal order $\{r_t\}$, with timestamp $t = \{1, 2, 3, \dots\}$. Denote $\mathcal{U}^t$ and $\mathcal{V}^t$ as the sets of users and items up to time t respectively, and a rating matrix $\mathbf{R}^t \in \mathbb{R}^{|\mathcal{U}^t| \times |\mathcal{V}^t|}$ as all the ratings from $\mathcal{U}^t$ to $\mathcal{V}^t$.*

Definition 7. *A **reputation fraud campaign (RFC)** consists of a group of **campaigned spammers** who are asked to conduct reputation fraud on a set of targeted items. A RFC can be denoted by a 3-tuple $\langle \mathcal{U}_c, \mathcal{R}_c, \mathcal{V}_c \rangle$ with $\mathcal{V}_c$ the set of targeted items, $\mathcal{R}_c$ the set of created ratings, and $\mathcal{U}_c$ the set of involved campaigned spammers.*

Definition 8. *To quantify the tendency of a user to participate in RFCs, a real-valued **spamicity** score $s_u^t \in \mathbb{R}^+$ is assigned to each user $u \in \mathcal{U}^t$ at time t . A ranked list of κ users with the largest spamicities S_κ^t will be produced as the **detection result** at time t .*

Given the definitions above, we formulate the task of **online reputation fraud campaign detection** as below: *Given the old rating matrix $\mathbf{R}^{t-1}$ at time $t - 1$, the*

model $\mathbf{M}^{t-1}$ learned from $\mathbf{R}^{t-1}$, and the detection result S_{κ}^{t-1} , upon a new rating comes in at time t , the goal is to efficiently update the new model $\mathbf{M}^t$ for reputation fraud campaign detection on the current rating matrix $\mathbf{R}^t$, and output the up-to-date detection result S_{κ}^t .

The fulfillment of the above task, however, requires two challenges to be addressed. The first is concerned with effective emerging campaign characterization. Various features have been proposed to characterize *occurred* campaigns on different dimensions. For example, deviation-based features [31, 37, 39] are commonly used to capture the differences between the ratings given by spammers and the remaining users. However, when dealing with emerging campaigns, if an item has not yet received sufficient ratings to form a general “ground-truth” rating at the beginning, the deviation of any incoming rating at this point would be no longer statistically significant to indicate manipulation. Another example is the profile-based features that focus on the external properties of campaigns, such as *Group Size* and *Group Support Count* used in [37]. These features may fail to accurately reflect the traits of emerging campaigns, since the size of a campaign is yet undetermined before the campaign ends.

The second challenge is about effective user rating behavior modeling. In online environments, the rating matrix $\mathbf{R}$ is inherently incomplete and sparse, which would cause many existing features to also be very sparse, and further hurt the performance of models built on those features. More robust representations are thus needed to encode user rating behaviors so as to effectively differentiate between campaigned spammers and normal users.

6.2 The Proposed Framework

To overcome the above challenges, we propose a novel framework, FRAUDSCAN, for on-line reputation fraud campaign detection. In light of the first challenge, we introduce the

notion of campaign context to capture the *interactions* between campaigned spammers. Such interactions are driven by ongoing collaborations between spammers from the same campaigns, as they are instructed to target the same items and give ratings with the same polarity during the campaign period (as illustrated in Figure 1.1). By monitoring such interactions across time, it is possible to spot emerging campaigns once they have launched. Second, to model the sparse rating behaviors, we take an embedding-based approach [22] where each user or item is mapped to a dense embedding formed by considering all user-rating-item relations in the data. A novel regularized matrix factorization model is proposed to substantiate the campaign embedding in the campaign context, and a campaign-aware online update scheme to further efficiently adapt the built detection model to new ratings.

6.2.1 Campaign Activity Monitoring

In this section, we introduce the campaign context built to monitor emerging campaign activities. Different from batch solutions, the campaign context is maintained incrementally upon the arrival of new ratings.

6.2.1.1 Target Conformity

The first type of campaign context is focused on the items targeted by campaigned spammers. Due to the involvement in the same campaigns, spammers would naturally co-rate the same items, and are tied by such co-rating actions. We model such co-rating behaviors as a matrix $\mathbf{C}^t \in \mathbb{R}^{|\mathcal{U}^t| \times |\mathcal{U}^t|}$, where each entry $c_{i,i'}^t$ is the normalized number of common items rated by both user u_i and $u_{i'}$ up to time t . The update rule for $\mathbf{C}^t$ is as follows: upon the arrival of a new rating r_{ij} from user u_i to item v_j , we update $c_{i,i'}^t$ as,

$$\begin{aligned} c_{i,i'}^{*t} &= c_{i,i'}^{*t-1} + 1 \quad \text{if } u_{i'} \text{ had also rated } v_j \\ c_{i,i'}^t &= c_{i,i'}^{*t} / (\sqrt{|\mathcal{V}_i^t|} \cdot \sqrt{|\mathcal{V}_{i'}^t|}) \end{aligned} \tag{Eq. 6.1}$$

where $c_{i,i'}^*$ is a helper variable for counting; $\mathcal{V}_i^t$ and $\mathcal{V}_{i'}^t$ are the items rated by u_i and $u_{i'}$ up to time t respectively.

6.2.1.2 Agreement Intensity

The ratings given by campaigned spammers should also agree on the co-rated items, as they are instructed to post either positive ratings for promotion or negative ratings for demotion. Thus, the differences between their ratings given to the co-rated items should be small. We use a matrix $\mathbf{G}^t \in \mathbb{R}^{|\mathcal{U}^t| \times |\mathcal{U}^t|}$ to capture the intensity of spammers' agreements. Each entry $g_{i,i'}^t$ keeps the up-to-date average rating difference between u_i and $u_{i'}$, and is updated as follows,

$$\begin{aligned} g_{i,i'}^{*t} &= g_{i,i'}^{*t-1} + e^{-|r_{ij} - r_{i'j}|^2} \quad \text{if } u_{i'} \text{ had also rated } v_j \\ g_{i,i'}^t &= g_{i,i'}^{*t} / c_{i,i'}^{*t} \end{aligned} \quad (\text{Eq. 6.2})$$

where $g_{i,i'}^*$ is the helper variable, and RBF (radial basis function) kernel is used to compute the rating deviation.

6.2.1.3 Operational Constraint

RFCs are often associated with time schedules specifying for how long the campaigns would last. In order to achieve the desired fraud effects, spammers are required to finish their jobs in time, such that their utilities can be aggregated. Such a constraint, however, would necessarily lead to small gaps between the timings of fraud activities. To account for this intuition, a matrix $\mathbf{O}^t \in \mathbb{R}^{|\mathcal{U}^t| \times |\mathcal{U}^t|}$ is used for tracking the proximity of users in time, with each entry $o_{i,i'}^t$ recording the average time interval between ratings given by u_i and $u_{i'}$ on the co-rated items so far, and being updated as below,

$$\begin{aligned} o_{i,i'}^{*t} &= o_{i,i'}^{*t-1} + e^{-|t_{ij} - t_{i'j}|^2} \quad \text{if } u_{i'} \text{ had also rated } v_j \\ o_{i,i'}^t &= o_{i,i'}^{*t} / c_{i,i'}^{*t} \end{aligned} \quad (\text{Eq. 6.3})$$

where $o_{i,i'}^*$ is the helper variable, and $t_{ij}(t_{i'j})$ is the timestamp of $u_i(u_{i'})$'s rating for v_j .

6.2.2 Fraud Campaign Embedding

As discussed, to address the sparsity of incoming rating data, we take an embedding-based approach for robust user rating behavior modeling, and propose a regularized matrix factorization model for RFC embedding. Concretely, we map each entity (a user or an item) in the data to a real-valued, dense vector in a low-dimensional space by taking all user-rating-item relations into account. This can be achieved by factorizing the original rating matrix $\mathbf{R}^t$ into two matrices, i.e., user matrix $\mathbf{U}^t \in \mathbb{R}^{k \times m}$ and item matrix $\mathbf{V}^t \in \mathbb{R}^{k \times n}$, where $k \ll \min(m, n)$ is the rank of the embedded space, which is commonly set to be low (e.g., within $[10, 50]$). Each column $\mathbf{u}_i^t$ of $\mathbf{U}^t$ (or $\mathbf{v}_j^t$ of $\mathbf{V}^t$) corresponds to the current embedding of user u_i (or item v_j). Meanwhile, to effectively capture campaign activities, we regularize such factorization in a proper way so that *users who have stronger interactions in the campaign context are mapped to closer regions in the embedded space*. As such, spammers from the same campaigns can be clustered together due to their strong connections in the campaign context, while normal users on the other hand would be dispersed randomly in the embedded space. Formally, we model the task of online RFC detection as a regularized embedding problem:

$$\begin{aligned} \arg \min_{\mathbf{U}^t, \mathbf{V}^t} L^t = & \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^n I_{ij} (r_{ij} - \mathbf{u}_i^{tT} \mathbf{v}_j^t)^2 \\ & + \frac{\alpha_c}{2} \sum_{i=1}^m \sum_{i' > i}^m c_{i,i'}^t \|\mathbf{u}_i^t - \mathbf{u}_{i'}^t\|_F^2 + \frac{\alpha_g}{2} \sum_{i=1}^m \sum_{i' > i}^m g_{i,i'}^t \|\mathbf{u}_i^t - \mathbf{u}_{i'}^t\|_F^2 \\ & + \frac{\alpha_o}{2} \sum_{i=1}^m \sum_{i' > i}^m o_{i,i'}^t \|\mathbf{u}_i^t - \mathbf{u}_{i'}^t\|_F^2 + \frac{\lambda_u}{2} \|\mathbf{U}^t\|_F^2 + \frac{\lambda_v}{2} \|\mathbf{V}^t\|_F^2 \end{aligned} \quad (\text{Eq. 6.4})$$

where I_{ij} is the indicator function that equals to 1 if u_i rated v_j , and 0 otherwise. $\|\cdot\|_F$ is Frobenius Norm. $\alpha_c, \alpha_g, \alpha_o, \lambda_u, \lambda_v$ are non-negative regularization parameters. The term $\sum_{i=1}^m \sum_{j=1}^n I_{ij} (r_{ij} - \mathbf{u}_i^{tT} \mathbf{v}_j^t)^2$ is the objective function that factorizes the input rating matrix $\mathbf{R}^t$ into user and item matrices $\mathbf{U}^t$ and $\mathbf{V}^t$. The terms $\sum_{i=1}^m \sum_{i'=1}^m (*)_{i,i'}^t \|\mathbf{u}_i^t - \mathbf{u}_{i'}^t\|_F^2$

are inspired by fused lasso [52], and used to constrain the former objective function, so as to bridge the campaign context and the embedded space. More specifically, they are designed to penalize large differences between the embeddings of two users if they are strongly correlated in the campaign context (i.e., with large $c_{i,i'}^t, g_{i,i'}^t, o_{i,i'}^t$). As such, campaigned spammers will obtain similar embeddings (in the sense of small Frobenius distance) due to their strong connections in the campaign context. The last two terms $\|\mathbf{U}^t\|_F^2$ and $\|\mathbf{V}^t\|_F^2$ are regularizers for avoiding overfitting.

To solve this optimization problem, stochastic gradient descent (SGD) is commonly used in the literature. In our case, for each rating $r_{ij} \in \mathbf{R}^t$, the following updating equations are used to learn the detection model $\mathcal{M}^t = \{\mathbf{U}^t, \mathbf{V}^t\}$:

$$\begin{aligned} \mathbf{u}_i^t &\leftarrow \mathbf{u}_i^t + \beta_u \left((r_{ij} - \mathbf{u}_i^{tT} \mathbf{v}_j^t) \mathbf{v}_j^t - \alpha_c \sum_{i'}^m c_{i,i'}^t (\mathbf{u}_i^t - \mathbf{u}_{i'}^t) \right. \\ &\quad \left. - \alpha_g \sum_{i'}^m g_{i,i'}^t (\mathbf{u}_i^t - \mathbf{u}_{i'}^t) - \alpha_o \sum_{i'}^m o_{i,i'}^t (\mathbf{u}_i^t - \mathbf{u}_{i'}^t) - \lambda_u \mathbf{u}_i^t \right) \\ \mathbf{v}_j^t &\leftarrow \mathbf{v}_j^t + \beta_v \left((r_{ij} - \mathbf{u}_i^{tT} \mathbf{v}_j^t) \mathbf{u}_i^t - \lambda_v \mathbf{v}_j^t \right) \end{aligned} \quad (\text{Eq. 6.5})$$

where β_u and β_v are the learning rates.

Once SGD converged, for each user pair $(u_i, u_{i'})$, we compute the (inverted) Frobenius distance $s_{i,i'}^t = \frac{1}{1 + \|\mathbf{u}_i^t - \mathbf{u}_{i'}^t\|_F^2}$ to measure the degree of their closeness in the embedded space. Then, the spamicity of a user u_i is derived as the highest closeness she has with any other user so far, i.e., $s_i^t = \max(\{s_{i,i'}^t | u_{i'} \in \mathcal{U}^t\})$. To generate the detection result S_κ^t , we rank the users by their current spamicities, and obtain the list of κ users with the largest values.

6.2.3 Online Campaign Detection

The above optimization procedure, however, is not sufficient for efficient and effective online RFC detection. First, the update scheme is in batch-mode; the model is not

updated across time. Second, it implicitly assumes that the users are independent, and does not utilize the fact that campaigned spammers are correlated in the campaign context. Here, we propose a *campaign-aware* online optimization scheme to update the built detection model. The key observation is that, whenever a new rating arrives, it is not necessary to make updates for all users and items in the data. Instead, only the parts of the model that are related to the incoming rating should be affected. More specifically, let r_{ij} be the current rating from a user u_i to an item v_j , and U_j^t the set of users who had rated v_j up to time t . Then, the arrival of r_{ij} would only change the relations between u_i and those in U_j^t in the campaign context (see Eq. (1-3)). Moreover, as rating data increase, the impact of each rating to the embedding model decreases, and new ratings would not significantly change the entire embedded space. Thus, to obtain the current user matrix $\mathbf{U}^t$ and item matrix $\mathbf{V}^t$, it is sufficient to only update the columns of $\mathbf{U}^{t-1}$ that correspond to u_i and the users in U_j^{t-1} , and the column of $\mathbf{V}^{t-1}$ that corresponds to v_j . In other words, to obtain the current detection model $\mathcal{M}^t = \{\mathbf{U}^t, \mathbf{V}^t\}$, we approximate $\mathbf{U}^t$ and $\mathbf{V}^t$ by updating the embeddings of u_i , U_j^{t-1} , and v_j that undergo changes in the campaign context, and keep the remaining part fixed as $\mathcal{M}^{t-1}$. This leads to Algorithm 3 for incremental RFC detection model update.

The algorithmic flow consists of three stages. First, we initialize the embeddings of u_i and v_j if either of them is new (line 1–6). Then, the campaign embedding is updated (line 7–13). In this stage, we re-train the embedding of u_i by first initializing $\mathbf{u}_i$ and then updating the embedding according to her up-to-date rating profile $\mathbf{R}_i^t = \mathbf{R}_i^{t-1} \cup \{r_{ij}\}$ (line 9-10). Then, we update the embeddings of users in U_j^t who are related to the current rating in the campaign context (line 11-12). Finally, we generate the detection report. For efficient update, the report S_κ is implemented as a max-heap that stores the largest κ spamicities and the corresponding users, which supports $O(\log \kappa)$ insertion. Through only updating the entities undergoing changes in campaign context, the complexity of

Algorithm 3: Incremental RFC Detection Flow

Input: $\mathbf{C}^t, \mathbf{G}^t, \mathbf{O}^t, \mathbf{U}^{t-1}, \mathbf{V}^{t-1}, S_\kappa^{t-1}, \mathbf{R}^{t-1} \cup \{r_{ij}\}$
Output: $\mathbf{U}^t, \mathbf{V}^t, S_\kappa^t$
 // (1) Initialization
 1 **if** u_i is new **then**
 2 Append a new column to the rightmost of $\mathbf{U}^{t-1}$;
 3 Initialize $\mathbf{u}_i$, i.e., the u_i 's column in $\mathbf{U}^{t-1}$;
 4 **if** v_j is new **then**
 5 Append a new column to the rightmost of $\mathbf{V}^{t-1}$;
 6 Initialize $\mathbf{v}_j$, i.e., the v_j 's column in $\mathbf{V}^{t-1}$;
 // (2) Update user and item matrix
 7 Initialize $\mathbf{u}_i$;
 8 **repeat**
 9 **foreach** $r_{i*} \in \mathbf{R}_i^{t-1} \cup \{r_{ij}\}$ **do**
 10 Update $\mathbf{u}_i$ and $\mathbf{v}_*$ according to Eq. 6.5;
 11 **foreach** $u_{i'} \in U_j^t$ **do**
 12 Update $\mathbf{u}_{i'}$ according to Eq. 6.5;
 13 **until** Convergent or maximum iteration reached;
 // (3) Update detection report
 14 **foreach** $(u_i, u_{i'})$ **do**
 15 $s_{i,i'}^t = \frac{1}{1 + \|\mathbf{u}_i^t - \mathbf{u}_{i'}^t\|_F^2}$;
 16 $s_i^t = \max(\{s_{i,i'}^t | u_{i'} \in \mathcal{U}^t\})$;
 17 Insert $\{s_i^t, u_i\}$ into S_κ^{t-1} ;
 18 **return** $\mathbf{U}^t, \mathbf{V}^t, S_\kappa^t$

online RFC detection has been considerably reduced. The time complexity of Algorithm 3 is $O((|\mathbf{R}_i^t| + |U_j^t|) \times k \times \text{Iter})$ for campaign embedding update and $O(|U_j^t| \log \kappa)$ for result reporting. Note that $\mathbf{R}_i^t$ and U_j^t are only the i th and j th columns of rating matrix $\mathbf{R}^t$, which are typically small in real applications (e.g., power-law distributed).

6.3 Experimental Analysis

In this section, we validate the proposed FRAUDSCAN for online reputation fraud campaign detection based on extensive experiments on real-world datasets. The evaluation is focused on the effectiveness of the proposed framework in RFC detection, and the efficiency of the proposed incremental scheme for detection model update.

6.3.1 Datasets

Our experiments are conducted on the same datasets as used in the previous chapters, i.e., AmazonCn and YelpZip, which contain the golden standard of collusive reputation fraud in the wild.

6.3.2 Evaluation Protocol

For effectiveness comparison, as FRAUDSCAN outputs a ranked list of users with numerical spamicity scores, we adopt two well-known metrics - Average Precision (AP) and Area Under ROC Curve (AUC) - for ranking performance evaluation. Average Precision is essentially the area under precision-recall curve, and AUC is commonly used to validate binary classification systems by varying the discrimination threshold. For efficiency evaluation, runtime of the detection algorithm is used as the performance metric.

6.3.3 Performance Evaluation

In this section, we evaluate the effectiveness of FRAUDSCAN for reputation fraud campaign detection.

6.3.3.1 Qualitative Analysis

We first conduct qualitative analysis. The success of FRAUDSCAN lies in the regularized campaign embedding that maps spammers from the same campaigns to close regions in the embedded space. Figure 6.1 shows the user projections in the embedded space on the two datasets. We can see that on both datasets, the campaigned spammers are effectively clustered together as expected, while the normal users are more scattered over the space, which validates the efficacy of the proposed embedding model. In addition, this also shows that our method can be used for visualization, which is lacking in other unsupervised methods that yield spamicity scores only.

Figure 6.1: User projections in the embedded space (the first two principal components are used). Samples consisting of 100 campaigned spammers (in red/circle) and 100 normal users (in blue/triangle) are shown.

6.3.3.2 Comparison Analysis

Next, we compare FRAUDSCAN with state-of-the-art approaches, with the following baselines used:

- GSRank [37]: a power iteration-based approach for spotting campaigned spammers based on the causal relations between users, reviews, and items. A set of Group Spam Behavior Indicators are proposed to capture suspicious collective behaviors of campaigned spammers.
- LCM [60], a statistical framework to model the collective behaviors of campaigned spammers for campaign inference and prediction.

In addition, two variants of FRAUDSCAN (FS) are also used:

- FS.B: this is a batch variant of the proposed framework. In this variant, only the

6.2.a: AmazonCn

6.2.b: YelpZip

Figure 6.2: Effectiveness comparison between FRAUDSCAN and state-of-the-art approaches on the two datasets.

matrix factorization objective is optimized, without the use of campaign context-based regularization.

- FS_BC: this is a batch variant of the proposed framework, with campaign context-based regularization used.

We empirically set optimal parameters for the hyperparameters (e.g., learning rate) of each method using a grid search in $\{0.001, 0.01, 0.1, 0.5, 1\}$. For FRAUDSCAN and its variants, four parameters α_1 , α_2 , α_3 , and k need to be tuned. We will show later the sensitivity analysis conducted on these parameters. Here, the optimal settings $(\alpha_1, \alpha_2, \alpha_3, k)$ are used, i.e., $(0.1, 0.1, 0.01, 20)$ for AmazonCn and $(0.01, 0.1, 0.01, 30)$ for YelpZip. Figure 6.2 shows the performance comparison on the two datasets. In the figure, each result is the average over 10 runs of the corresponding experiment. All the improvements in the results are significant at $p < .05$ with paired t-test. The following observations are made based on the result comparison.

First, we can see that our proposed methods FS_BC and FRAUDSCAN achieve better results than other baselines on both datasets. In particular, by comparing FS_BC with the two start-of-the-art methods GSRank and LCM, the performance is improved by a margin of [2.4%, 3.2%] in AP and [1.5%, 14.5%] in AUC on AmazonCn, and [3.2%, 33.4%] in AP and [3.8%, 11.5%] in AUC on YelpZip. This might be ascribed to the embedding-based approach used in FRAUDSCAN for user rating behavior modeling, which compactly encodes rich user-rating-item relationships in the sparse original rating matrix in a more robust way.

Second, by comparing different variants of FRAUDSCAN, we observe that FS_B performs worse than the other two on both datasets, suggesting that the factorization of the input rating matrix alone is not sufficient for campaign activity characterization. By further utilizing the campaign context proposed for capturing interactions between campaigned spammers, the performance of FS_BC is largely improved. This shows that the incorporation of campaign-level information is crucial for collusive behavior modeling.

Finally, FS_BC and FRAUDSCAN achieve comparable results on both datasets, suggesting that the approximation adopted in the proposed online detection approach does not harm the embedded model building and the detection performance.

6.3.4 Earliness of Detection

One important objective of online detection is to spot emerging campaign activities as early as possible, so as to limit the caused damage in time. To this regard, we run the compared methods across time, with a one-day interval, and measure the earliness of detection by computing the lag between the time when a spammer appeared for the first time in the system and the time when (s)he was first detected. Ideally, the lag should be zero for all spammers.

Figure 6.3: Histograms of detection time lags for spammers caught by the compared methods. The histograms are truncated at 90-day lag due to the long-tail shape.

Figure 6.3 shows the results on AmazonCn¹, by drawing histograms of detection time lags (in days) for caught spammers (true positives)². We find that all the evaluated methods can discover more than half of the caught spammers in less than 10 days after their first ratings. In particular, FRAUDSCAN achieves better results than others by catching more spammers with less delay. Surprisingly, 921 spammers are caught by FRAUDSCAN right after they have posted the first ratings (zero delay), which is 34.4% more than LCM and 45.9% more than GSRank. This shows that compared to other methods, FRAUDSCAN is more effective in quickly responding to emerging reputation fraud campaigns.

6.3.5 Sensitivity Analysis

In this section, we evaluate the influences of model parameters on the detection performance. They are α_1 , α_2 , α_3 , and k in our framework controlling the importances of the three types of campaign context and the rank of the embedded space. For each α parameter, an independent experiment is conducted by setting other parameters to zero. The results on AmazonCn are shown in Figure 6.4.

¹We omit the results on YelpZip due to less significant improvements being observed. The main reason is that the activeness of campaigns on YelpZip is relatively low, resulting in inherently large time lags between consecutive ratings.

²We regard the users ranked at the top 2000 as positives, since totally 1,937 campaigned spammers are annotated in the dataset.

Figure 6.4: The effects of parameters α_1 , α_2 , α_3 , and k on the detection performance (AUC).

We can see that all the results show similar patterns. The detection performance first rises as the parameter values increase. It then reaches the peak at particular settings and decreases afterwards. This is because when the parameters are set too small, the campaign context cannot be fully exploited to improve detection performance. However, the overuse of the campaign context can hurt the detection performance, since each type of the campaign context is just a necessary condition for characterizing RFCs. It can be seen that as long as the parameters are set within a medium range, the detection performance can always be improved compared to that without the campaign context. We can also observe that the rank of the embedded space should be set sufficiently large to achieve an optimal representation capacity.

6.3.6 Efficiency Analysis

Finally, we evaluate the runtime performance of our framework to validate its efficiency in RFC detection. In particular, the performance of the batch model FS_BC and that of the online learning model FRAUDSCAN are compared. The experiments are conducted on a machine with a single-CPU, 3.20Ghz and 16G memory.

Figure 6.5 shows the results on AmazonCn due to its larger size. We vary the size of training data, and record the execution time of each baseline. It can be observed that FRAUDSCAN requires less execution time than the batch version FS_BC on datasets with different sizes. This shows that our online learning method is more efficient than the batch counterpart in detection. Moreover, as the size of training data increases, the differences in efficiency between the two approaches become more significant. Specifically, when the full dataset is used, the proposed online learning approach achieves around 4.1x speedup in detection.

Figure 6.5: The cumulative distribution of detection latency.

6.4 Conclusions

In this chapter, we study the problem of reputation fraud campaign detection in user ratings, and propose a principled optimization framework for emerging fraud campaign detection. The proposed framework is based on the online learning scheme, and operates by monitoring campaign activities across time. To effectively characterize emerging fraud campaigns, the notion of campaign contexts is introduced to monitor immediate interactions between campaigned spammers. Moreover, to accurately model users' sparse rating behaviors, an embedding-based approach is employed to compactly map users' rating behaviors into low-dimensional space. In this approach, the campaign contexts are integrated as model regularizers that can effectively map campaigned spammers to proximate regions in the embedded space to facilitate detection. Finally, to efficiently adapt built models to emerging campaigns, an online learning scheme is proposed for model update. The main advantage of the proposed online learning scheme is that it is campaign-aware; the connections between campaigned spammers are utilized for effective model update. The experimental results on two real-world datasets show that our method achieves superior detection accuracy and efficiency over state-of-the-art batch-based detectors.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this section, we conclude the completed work in this thesis so far towards *the design of effective and efficient solutions to characterize and detect the most recent and damaging collusive reputation fraud practices in crowdsourced opinions*. To tackle the recently-uncovered stealthy attack that involves multiple colluder groups, graph-based classification algorithms are proposed in Chapter 3 for supervised collusive reputation fraud detection. In Chapter 4, a unified detection framework based on stochastic process formulation is proposed for collusive reputation fraud detection in unsupervised settings. Furthermore, to address the collusion prediction problem for handling emerging fraud practices, a statistical framework based on generative and discriminative modeling is proposed in Chapter 5. Finally, in Chapter 6, a principled and unified optimization framework is proposed for online reputation fraud campaign detection, which can adapt built detection model to newly-emerging fraud patterns.

More specifically, in Chapter 3, to cope with the latest adversarial patterns of collusive reputation fraud, two novel classification algorithms are proposed for colluder detection. Different from traditional classifiers focusing on individual objects, both of the two methods exploit relational information extracted from multiple objects for detection, so as to refine the original classification results. In FRAUDKNN, three similarity measures

are proposed to define a neighborhood for similar colluder groups, which examine the commonalities between groups in their members, targets, and review sentiments. The classification of a user is determined by the class labels of other users from similar groups. The second method CoRF is based on the more principled probabilistic graphical model, transaction correlations between users are utilized to associate potential colluders. A novel collective inference algorithm is developed for classification based on the colluder graph model. Extensive experiments have been conducted to validate and demonstrate the effectiveness of the proposed methods in colluder classification performance.

In Chapter 4, we study the problem of collusive reputation fraud in unsupervised settings. A novel detection framework FRAUDINFORMER has been proposed for unsupervised collusive reputation fraud detection. In FRAUDINFORMER, the mutual disclosure model provides an intuitive and effective way of conducting detection in unsupervised settings. Through proper evidence propagation, re-weighting, and ranking, colluders who are involved in the same campaigns can automatically be captured in a collective manner. The proposed heterogeneous pairwise features can effectively characterize the behavioral correlations between colluders. Different from existing group-based features, the characterization of pairwise features is more fine-grained, and can find stealthy colluders whose behaviors may not seem suspicious at group level. Moreover, the proposed framework decouples the processes of colluder characterization and detection, such that newly-designed features and specialized detection algorithms can easily and flexibly be integrated to achieve further performance improvements. Theoretical analysis has been conducted to confirm the robustness of the proposed method in defending against adversarial challenge. Extensive experimental results on real data validate the effectiveness of the proposed method and show that it significantly outperforms state-of-the-art competitors by revealing colluders with more concealed behaviors.

In Chapter 5, to address the collusion prediction problem, we formulate the problem of detecting collusive reputation fraud from a stochastic perspective. A novel statistical

model, named as Latent Collusion Model (LCM), has been proposed to model collusive reputation fraud from a hybrid of generative and discriminative perspectives. Two variants of LCM are studied in details. The first is a chaining-based model which performs collusion inference and prediction in a sequential manner. The chaining-based model is flexible in the sense that models suitable for the respective tasks can be optimized and applied separately. The second is a unified model that combines the learning and inference of the generative and discriminative views, so as to achieve the optimal results via balancing between the two views. Furthermore, a suite of homogeneity-based collusive behavior measures (h-CBMs) are proposed to capture the homophily inside colluder groups. The h-CBMs are complementary to existing collusion-oriented features in handling stealthier collusive attacks. Experimental results on real-world datasets show the effectiveness of h-CBMs and the superiority of LCM over state-of-the-art competitors in terms of collusion inference and predictive abilities.

In Chapter 6, the problem of online collusive reputation fraud detection is addressed. A principled optimization framework has been proposed for identifying emerging fraud campaigns. The proposed framework is based on the online learning scheme, and operates by monitoring campaign activities across time. To effectively characterize emerging fraud campaigns, the notion of campaign contexts is introduced to monitor immediate interactions between campaigned spammers. Moreover, to accurately model users' sparse rating behaviors, an embedding-based approach is employed to compactly map users' rating behaviors into low-dimensional space. In this approach, the campaign contexts are integrated as model regularizers that can map campaigned spammers to proximate regions in the embedded space to facilitate detection. Finally, to efficiently adapt built models to emerging campaigns, an online learning scheme is proposed for model update. Experimental results on real-world datasets show that the proposed method achieve superior detection accuracy and efficiency over state-of-the-art batch-based detectors.

In summary, the aim of this thesis is to address the collusive reputation fraud problem by behavioral analysis and characterization, and the design of effective and efficient detection algorithms that work in different scenarios. In terms of collusive reputation fraud characterization, various features have been invented to indicate suspicious colluding behaviors which are robust to adversarial offenses. In addition, these features are designed to be used in different scenarios (e.g., offline and online) with different forms (e.g., pairwise and group-wise). For collusive reputation fraud detection, we provide a suite of effective and efficient detection frameworks that can deal with sophisticated collusive fraud practices in various settings. The graph-based classification algorithms (Chapter 3) are suitable for supervised detection, while FRAUDINFORMER (Chapter 4), LCM (Chapter 5), and FRAUDSCAN (Chapter 6) are proposed for unsupervised detection. In addition, FRAUDSCAN is designed to cope with colluder detection in online scenarios, while the other three methods are focused on offline collusive fraud analysis and detection.

7.2 Future Work

In this section, several directions for future research are identified as follows.

7.2.1 Extension of the Graph-based Classification Algorithms

The proposed graph-based classification algorithms in Chapter 3 could be improved in several ways. First, the characterization ability of FRAUDKNN can further be enhanced. Specifically, the weights for combining the similarity measures used in FRAUDKNN (Section 3.3) are set empirically so far, which does not reflect the importance of each individual measure on the classification performance. In future work, machine learning-based approach can be applied to learn the relative contribution of each part, so as to achieve the optimal similarity combination. Second, for colluder random field (CoRF)

model, we note that the collusion scoring functions used in the collective inference algorithm in Section 3.4.2 only consider target and rating information for collusion intensity evaluation. Indeed, as we have seen in subsequent chapters, richer evidence based on temporal or other dimensions can be used and modeled to improve detection accuracy. Thus, the collective inference algorithm of CoRF can be extended to incorporate more diverse collusion evidence for colluder classification. Moreover, in supervised settings, instead of explicitly defining the pairwise potential functions associated with CoRF, we can effectively learn their formulations from the data via optimizing proper objective functions. In future work, a more principled framework can be formulated by incorporating the potential functions as a part of objective functions.

7.2.2 Extension of FraudInformer

In FRAUDINFORMER, the ranking algorithm proposed in Section 4.2.3 is crucial to the overall detection performance. The algorithm is based on Markov random walk which employs a smoothing vector with each element set to a constant in the stationary distribution computation (see Eq. 4.9). However, this formulation implicitly assumes that each user is equally important to the computation of each other’s spamicity score, which violates the fact that for colluder pairs, their mutual contributions to each other’s spamicities should be higher than those of non-colluder pairs. In further work, we shall relax such assumption by applying more sophisticated stochastic process such as personalized Markov random walk [21] that customizes the initial smoothing vector for stationary distribution computation. In addition, as the case discussed in the previous subsection, the weights for combining the proposed pairwise features for collusiveness measuring can also be learned using held-out data.

7.2.3 Extension of LCM

The proposed Latent Collusion Model in Chapter 5 can be extended in several ways.

First, in the model formulation, the parameters of the model are not regarded as random variables and are obtained based on point estimation. To further improve model fitting, it would be interesting to seek for a more principled formulation under the Bayesian framework, where all the model parameters are also modeled as random variables with imposed prior distributions. In this way, the inference of the model would yield the posterior distributions of the model parameters, which contain richer information about the collusive behavior patterns governed by those parameters.

Second, it can also be noted that, when modeling the dependency of the observed h-CBM vectors on the latent class labels from the discriminative view (Section 5.3.1.2), the logistic function is used for such relationship substantiation. It would also be interesting to see the differences in performance when alternative models are used, such as the probit regression model [5] for modeling binary random variables.

Thirdly, the class labels of the candidate groups considered in LCM only take binary values to account for the two classes, i.e., malicious and benign groups. Such a taxonomy might be too coarse for collusion analysis in real applications, where the user behavior patterns can be more diverse in both ends. In future work, more fine-grained taxonomies can be used in both characterization and detection to achieve a more robust behavior profiling.

Finally, textual features associated with the review contents are not used in h-CBMs for colluder characterization. In fact, as a spammer usually has multiple accounts at his/her disposal, the written style of the reviews posted by these accounts may share particular commonalities. For example, they may all contain specific words or phrases that are unique to the spammer. To this end, one possible future direction is to explore some stylometric features that can be extracted from the contents of reviews. With this approach, to capture campaigned spammers, we can first detect spammers who use multiple accounts for manipulation, by correlating reviews with similar written styles, and then group spammers who target the common item sets.

7.2.4 Extension of FraudScan

The proposed FRAUDSCAN in Chapter 6 can be improved in several ways.

First, the campaign contexts proposed in Section 6.2.1 are used to capture internal interactions within the reputation fraud campaigns. However, the characteristics of individual campaign members are ignored. As surveyed in Chapter 2, previous studies have proposed various features for characterizing individual review spammers on different dimensions. These features can also, to some extent, reflect the occurrence of reputation fraud, and can be incorporated into the proposed campaign embedding model in Section 6.2.2 for further performance improvements. In future work, the proposed campaign contexts can be extended to incorporate more meaningful factors concerning individual spammers.

Second, the suspect reporting procedure proposed in Section 6.2.2 can be further refined. Currently, the Frobenius distance between two users is used for user suspicion ranking. From a more general view, such distance can also be seen as a similarity measure on the collusiveness of users. As such, new ranking techniques similar to the global ranking algorithm in FRAUDINFORMER (Chapter 4) can be used to refine the detection results, by propagating collusiveness among users with Markov random walk. The advantage is that through propagation, the separation between colluders and non-colluders can be further enhanced due to the accumulation of collusiveness among colluders and the dispersion of collusiveness among normal users.

Thirdly, it is also necessary to study the robustness of the proposed framework to adversarial offenses. In particular, the proposed campaign embedding model can be extended to consider the manipulations towards the campaign contexts in online detection scenarios.

Finally, an alternative way of implementing embedding mechanism with heterogeneous information is based on deep learning. Along this line, we can model the the data

as a heterogeneous information network consisting of users and textual reviews, and then use deep neural networks to learn the embeddings of users which are expected to conceptually capture the colluders' behaviors together with their written contents and writing styles. The major benefit is that, in this way, we no longer need to explicitly specify the formulations of the campaign context to encode our observations and assumptions. Instead, it can be learned automatically from the data.

Authors Publications

Journals Articles

- 1 **Chang Xu** and Jie Zhang, “Collusive Opinion Fraud Detection in Online Reviews: A Probabilistic Modeling Approach”, *ACM Transactions on the Web (TWEB)*, 2016.

Conference Papers

- 1 **Chang Xu**, Jie Zhang, and Zhu Sun, “Online Reputation Fraud Campaign Detection in User Ratings”, *In Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI’17)*, 2017 (in submission).
- 2 Huanhuan Zhang, Jie Zhang, Carol Fung, and **Chang Xu**, “Improving Sybil Detection via Graph Pruning and Regularization Techniques”, *Proc. of the 7th Asian Conference on Machine Learning (ACML’15)*, Hong Kong, November 2015.
- 3 **Chang Xu** and Jie Zhang, “Towards Collusive Fraud Detection in Online Reviews”, *In Proceedings of the IEEE International Conference on Data Mining (IEEE ICDM’15)*, Atlantic City, NJ, USA, November 2015.
- 4 **Chang Xu** and Jie Zhang, “Combating Product Review Spam Campaigns via Multiple Heterogeneous Pairwise Features”, *In Proceedings of the 15th SIAM International Conference on Data Mining (SIAM SDM’15)*, Vancouver, BC, Canada, April 2015.

- 5 Huanhuan Zhang, **Chang Xu** and Jie Zhang, “Exploiting Trust and Distrust Information for Combating Sybil Attack in Online Social Networks”, *Proc. of the 8th International Conference on Trust management (IFIPTM’14)*, Singapore, July 2014.
- 6 **Chang Xu**, Jie Zhang, Kuiyu Chang and Chong Long, “Uncovering Collusive Spammers in Chinese Review Websites”, *In Proceedings of the 22nd ACM International Conference on Information and Knowledge Management (CIKM’13)*, San Francisco, CA, USA, October 2013.
- 7 **Chang Xu**, “Detecting Collusive Spammers in Online Review Communities”, *Proc. of the 6th Ph.D. Workshop on Information and Knowledge Management (PIKM’13)*, San Francisco, CA, USA, October 2013.

References

- [1] Leman Akoglu, Rishi Chandy, and Christos Faloutsos. Opinion fraud detection in online reviews by network effects. In *Proceedings of the 7th International AAAI Conference on Web and Social Media (ICWSM)*, volume 13, pages 2–11. Citeseer, 2013.
- [2] Luca Becchetti, Carlos Castillo, Debora Donato, Ricardo Baeza-Yates, and Stefano Leonardi. Link analysis for web spam detection. *ACM Transactions on the Web (TWEB)*, 2(1):2, 2008.
- [3] Alex Beutel, Wanhong Xu, Venkatesan Guruswami, Christopher Palow, and Christos Faloutsos. Copycatch: stopping group attacks by spotting lockstep behavior in social networks. In *Proceedings of the 22nd International Conference on World Wide Web (WWW)*, pages 119–130. ACM, 2013.
- [4] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 4. springer New York, 2006.
- [5] Chester I Bliss. The method of probits. *Science*, 79(2037):38–39, 1934.
- [6] Carlos Castillo, Debora Donato, Aristides Gionis, Vanessa Murdock, and Fabrizio Silvestri. Know your neighbors: Web spam detection using the web topology. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007.
- [7] Judith A Chevalier and Dina Mayzlin. The effect of word of mouth on sales: Online book reviews. *Journal of marketing research*, 43(3):345–354, 2006.

- [8] Hanbo Dai, Feida Zhu, Ee-Peng Lim, and HweeHwa Pang. Detecting anomalies in bipartite graphs with mutual dependency principles. In *Proceedings of the 12th IEEE International Conference on Data Mining (ICDM)*, pages 171–180. IEEE, 2012.
- [9] Hanbo Dai, Feida Zhu, Ee-Peng Lim, and HweeHwa Pang. Detecting extreme rank anomalous collections. In *Proceedings of the 12th SIAM International Conference on Data Mining (SDM)*, pages 883–894. SIAM, 2012.
- [10] Hanbo Dai, Feida Zhu, Ee-Peng Lim, and HweeHwa Pang. Mining coherent anomaly collections on web data. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1557–1561. ACM, 2012.
- [11] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.
- [12] Hui Fang, Jie Zhang, Yang Bao, and Qinghua Zhu. Towards effective online review systems in the chinese context: A cross-cultural empirical study. *Electronic Commerce Research and Applications*, 12(3):208–220, 2013.
- [13] Amir Fayazi, Kyumin Lee, James Caverlee, and Anna Squicciarini. Uncovering crowdsourced manipulation of online reviews. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 233–242. ACM, 2015.
- [14] Geli Fei, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Ridhiman Ghosh. Exploiting burstiness in reviews for review spammer detection. In *Proceedings of the 7th International AAAI Conference on Web and Social Media (ICWSM)*, volume 13, pages 175–184. Citeseer, 2013.
- [15] S. Feng, L. Xing, A. Gogar, and Y. Choi. Distributional footprints of deceptive product reviews. In *Proceedings of the 6th International AAAI Conference on Web and Social Media (ICWSM)*, volume 12, pages 98–105, 2012.

- [16] Chris Forman, Anindya Ghose, and Batia Wiesenfeld. Examining the relationship between reviews and sales: The role of reviewer identity disclosure in electronic markets. *Information Systems Research*, 19(3):291–313, 2008.
- [17] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *The Journal of machine learning research*, 4:933–969, 2003.
- [18] Hongyu Gao, Jun Hu, Christo Wilson, Zhichun Li, Yan Chen, and Ben Y Zhao. Detecting and characterizing social spam campaigns. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 35–47. ACM, 2010.
- [19] Saptarshi Ghosh, Bimal Viswanath, Farshad Kooti, Naveen Kumar Sharma, Gautam Korlam, Fabricio Benevenuto, Niloy Ganguly, and Krishna Phani Gummadi. Understanding and combating link farming in the twitter social network. In *Proceedings of the 21st International Conference on World Wide Web (WWW)*, pages 61–70. ACM, 2012.
- [20] Zoltan Gyongyi and Hector Garcia-Molina. Web spam taxonomy. In *Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2005.
- [21] Taher H Haveliwala. Topic-sensitive pagerank. In *Proceedings of the 11th International Conference on World Wide Web (WWW)*, pages 517–526. ACM, 2002.
- [22] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 50–57. ACM, 1999.
- [23] Siwei Jiang, Jie Zhang, and Yew-Soon Ong. An evolutionary model for constructing robust trust networks. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 813–820. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

REFERENCES

- [24] Nitin Jindal and Bing Liu. Opinion spam and analysis. In *Proceedings of the International Conference on Web Search and Web Data Mining (WSDM)*, pages 219–230. ACM, 2008.
- [25] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the 8th International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 133–142. ACM, 2002.
- [26] A Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, 14:841, 2002.
- [27] Christian Kreibich, Chris Kanich, Kirill Levchenko, Brandon Enright, Geoffrey M Voelker, Vern Paxson, and Stefan Savage. On the spam campaign trail. *Proceedings of the First USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 8:1–9, 2008.
- [28] Christian Kreibich, Chris Kanich, Kirill Levchenko, Brandon Enright, Geoffrey M Voelker, Vern Paxson, and Stefan Savage. Spamcraft: An inside look at spam campaign orchestration. *Proceedings of the 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2009.
- [29] Julia A Lasserre, Christopher M Bishop, and Thomas P Minka. Principled hybrids of generative and discriminative models. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 87–94. IEEE, 2006.
- [30] F. Li, M. Huang, Y. Yang, and X. Zhu. Learning to identify review spam. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2488–2493, 2011.
- [31] E.P. Lim, V.A. Nguyen, N. Jindal, B. Liu, and H.W. Lauw. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 939–948. ACM, 2010.

REFERENCES

- [32] Siyuan Liu, Jie Zhang, Chunyan Miao, Yin-Leng Theng, and Alex C Kot. An integrated clustering-based approach to filtering unfair multi-nominal testimonies. *Computational Intelligence*, 30(2):316–341, 2014.
- [33] Yong Liu. Word of mouth for movies: Its dynamics and impact on box office revenue. *Journal of marketing*, 70(3):74–89, 2006.
- [34] Michael Luca and Georgios Zervas. Fake it till you make it: Reputation, competition, and yelp review fraud. *Harvard Business School NOM Unit Working Paper*, (14-006), 2013.
- [35] Luke K McDowell, Kalyan M Gupta, and David W Aha. Cautious inference in collective classification. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2007.
- [36] Arash Molavi Kakhki, Chloe Kliman-Silver, and Alan Mislove. Iolaus: Securing online content rating systems. In *Proceedings of the 22nd international conference on World Wide Web (WWW-13)*, pages 919–930, 2013.
- [37] A. Mukherjee, B. Liu, and N. Glance. Spotting fake reviewer groups in consumer reviews. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 191–200. ACM, 2012.
- [38] A. Mukherjee, B. Liu, J. Wang, N. Glance, and N. Jindal. Detecting group review spam. In *Proceedings of the 20th International Conference Companion on World Wide Web (WWW)*, pages 93–94. ACM, 2011.
- [39] Arjun Mukherjee, Abhinav Kumar, Bing Liu, Junhui Wang, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. Spotting opinion spammers using behavioral footprints. In *Proceedings of the 19th International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 632–640. ACM, 2013.
- [40] Jennifer Neville and David Jensen. Iterative classification in relational data. In *Proceedings of the AAAI Workshop on Learning Statistical Models from Relational Data*, 2000.

REFERENCES

- [41] Myle Ott, Claire Cardie, and Jeff Hancock. Estimating the prevalence of deception in online review communities. In *Proceedings of the 21st International Conference on World Wide Web (WWW)*, pages 201–210. ACM, 2012.
- [42] Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T Hancock. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 309–319. Association for Computational Linguistics, 2011.
- [43] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. (1999-66), November 1999.
- [44] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3), 1962.
- [45] Shebuti Rayana and Leman Akoglu. Collective opinion spam detection: Bridging review networks and metadata. In *Proceedings of the 21th International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 985–994. ACM, 2015.
- [46] Prithviraj Sen and Lise Getoor. Empirical comparison of approximate inference algorithms for networked data. In *Proceedings of ICML Workshop on Statistical Relational Learning (SRL)*, 2006.
- [47] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93, 2008.
- [48] Brett Stone-Gross, Thorsten Holz, Gianluca Stringhini, and Giovanni Vigna. The underground economy of spam: A botmasters perspective of coordinating large-scale spam campaigns. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2011.
- [49] David Streitfeld. The best book reviews money can buy. <http://www.nytimes.com/2012/08/26/business/book-reviewers-for-hire-meet-a-demand-for-online-raves.html>. accessed September 8, 2013.

REFERENCES

- [50] Huan Sun, Alex Morales, and Xifeng Yan. Synthetic review spamming and defense. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-13)*, pages 1088–1096, 2013.
- [51] Ben Taskar, Pieter Abbeel, and Daphne Koller. Discriminative probabilistic models for relational data. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 485–492, 2002.
- [52] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108, 2005.
- [53] Dongxia Wang, Tim Muller, Jie Zhang, and Yang Liu. Quantifying robustness of trust systems against collusive unfair rating attacks using information theory. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.
- [54] G. Wang, S. Xie, B. Liu, and S.Y. Philip. Identify online store review spammers via social review graph. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(4), 2012.
- [55] Gang Wang, Tianyi Wang, Haitao Zheng, and Ben Y Zhao. Man vs. machine: Practical adversarial detection of malicious crowdsourcing workers. In *23rd USENIX Security Symposium, USENIX Association, CA*, 2014.
- [56] Guan Wang, Sihong Xie, Bing Liu, and Philip S Yu. Identify online store review spammers via social review graph. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(4):61, 2012.
- [57] Baoning Wu and Brian D Davison. Identifying link farm spam pages. In *Special interest tracks and posters of the 14th international conference on World Wide Web*. ACM, 2005.
- [58] Sihong Xie, Guan Wang, Shuyang Lin, and Philip S Yu. Review spam detection via temporal pattern discovery. In *Proceedings of the 18th International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 823–831. ACM, 2012.

- [59] Chang Xu and Jie Zhang. Combating product review spam campaigns via multiple heterogeneous pairwise features. *Proceedings of the 15th SIAM International Conference on Data Mining (SDM)*, 2015.
- [60] Chang Xu and Jie Zhang. Towards collusive fraud detection in online reviews. In *Proceedings of the 15th IEEE International Conference on Data Mining (ICDM)*, pages 1051–1056. IEEE, 2015.
- [61] Chang Xu, Jie Zhang, Kuiyu Chang, and Chong Long. Uncovering collusive spammers in chinese review websites. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management (CIKM)*. ACM, 2013.
- [62] Chao Yang, Robert Harkreader, Jialong Zhang, Seungwon Shin, and Guofei Gu. Analyzing spammers’ social networks for fun and profit: a case study of cyber criminal ecosystem on twitter. In *Proceedings of the 21st International Conference on World Wide Web (WWW)*, pages 71–80. ACM, 2012.
- [63] Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning (ICML)*, volume 97, pages 412–420, 1997.
- [64] Huanhuan Zhang, Chang Xu, and Jie Zhang. Exploiting trust and distrust information to combat sybil attack in online social networks. In *IFIP International Conference on Trust Management*, pages 77–92. Springer, 2014.
- [65] Huanhuan Zhang, Jie Zhang, Carol Fung, and Chang Xu. Improving sybil detection via graph pruning and regularization techniques. In *Proceedings of The 7th Asian Conference on Machine Learning*, pages 189–204, 2015.