6-2013

Anomaly Detection on Social Data

Hanbo DAI
Singapore Management University, hanbo.dai.2008@smu.edu.sg

Anomaly Detection on Social Data

by

Dai Hanbo

Submitted to School of Information Systems in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Information Systems

Dissertation Committee:

Lim Ee-Peng (Supervisor/Chair)
Professor of Information Systems
Singapore Management University

Pang Hwee Hwa
Professor of Information Systems
Singapore Management University

Zhu Feida
Assistant Professor of Information Systems
Singapore Management University

Sourav S Bhowmick
Associate Professor, School of Computer Engineering
Nanyang Technological University

Singapore Management University

2013

Abstract

Anomaly Detection on Social Data

Dai Hanbo

The advent of online social media including Facebook, Twitter, Flickr and Youtube has drawn massive attention in recent years. These online platforms generate massive data capturing the behavior of multiple types of human actors as they interact with one another and with resources such as pictures, books and videos. Unfortunately, the openness of these platforms often leaves them highly susceptible to abuse by suspicious entities such as spammers. It therefore becomes increasingly important to automatically identify these suspicious entities and eliminate their threats. We call these suspicious entities *anomalies* in social data, as they often hold different agenda comparing to normal ones and manifest anomalous behaviors.

In this dissertation, we are interested in two kinds of anomalous behaviors in social data, namely the unusual coalition among a collection of entities and the unusual conflicting opinions among entities. The two kinds of anomalous behaviors lead us to define two types of anomalies, namely, anomaly collections of the same entity type and anomalous nodes of different entity types in bipartite graphs.

This dissertation introduces two anomaly collection definitions, namely, *Extreme Rank Anomalous Collection (or ERAC)* and *Coherent Anomaly Collection (or CAC)*. An ERAC is a set of entities that cluster toward the top or bottom ranks, when all entities in the population are ranked on certain features. We propose a statistical model to quantify the anomalousness of an ERAC, and present the exact as well as heuristic algorithms for finding top-K ERACs. We then propose the follow-up problem of expanding top-K ERACs to anomalous supersets. We apply the algorithms for ERAC detection and expansion on both synthetic and real-life datasets, including a web spam, an IMDB and a Chinese online forum dataset. Results show that

our algorithms achieve higher precisions compared to existing spam and anomaly detection methods.

CAC is defined based on ERAC, emphasizing the coherence among members of an ERAC. As top-K ERACs are often overlapping with each other, for applications where disjoint anomaly collections are of interest, we propose to find top-K disjoint CACs with exact and heuristic algorithms. Experiments on both synthetic and real-life datasets, including a Twitter, a web spam, and a Chinese online forum dataset show that our approach discovers not only injected anomaly collections in synthetic datasets but also real-life coherent collections of hashtag spammer, web spammers and opinion spammers which are hard to detect by clustering-based methods.

We detect the second type of anomalies in a bipartite graph, where nodes in one partite represent human actors, nodes in the other partite represent resources, and edges carry the agreeing and disagreeing opinions from human actors to resources. The anomalousness of nodes in one partite depends on that of their connected nodes in the other partite. Previous studies have shown that this mutual dependency can be positive or negative. We integrate both mutual dependency principles to model the anomalous behavior of nodes. We formulate our principles and design an iterative algorithm to simultaneously compute the anomaly scores of nodes in both partites. Our method is applied on synthetic graphs and the results show that our algorithm outperforms existing ones with only positive or negative mutual dependency principles. Results on two real-life datasets, namely Goodreads and Buzzcity, show that our method is able to detect suspected spammed books in Goodreads and fraudulent publishers in mobile advertising networks with higher precision than existing approaches.

Contents

List of Tables

List of Figures

Acknowledgements

This dissertation would not have been possible without the help and support of many people in many ways. I consider myself extremely lucky to have them around during my Ph.D study.

I am grateful to Prof Lim Ee-Peng for imparting the quality of a good researcher, unwearied diligence and rigorous scholarship. I am thankful to Prof Pang Hwee Hwa for challenging and greatly enriching my ideas. I am grateful to Prof Zhu Feida for his kind support and timely encouragement. I thank Prof Lim, Prof Pang, and Prof Zhu for their generous help in our weekly meetings and patience in revising our papers. I would also like to thank my external committee member Prof Sourav Saha Bhowmick for his valuable suggestions.

I owe sincere and earnest thanks to my wife Wu Hui and my parents in China for the support they have given me all these years.

I would like to express my gratitude to all professors, researchers, staff and fellow students in SMU for numerous discussions, and their friendship.

Chapter 1

Introduction

1.1 Motivations

In recent years, the boom of online social media including Facebook, Twitter, Flickr and Youtube, has provided an unprecedented degree of freedom for ordinary users to engage in online social interactions and generate content. As a result, these online platforms generate enormous amount of social data every day. These social data often capture the behaviors of human actors as they interact with one another and with content resources. For example, people make friends in Facebook, follow other people in Twitter, share and tag pictures in Flickr, share and comment on videos in Youtube.

On one hand, online platforms facilitate interactions among multiple types of entities. On the other hand, the openness of platforms leaves the interactions highly susceptible to abuse, and even worse, the sheer volume of the generated data makes it infeasible to manually inspect their veracity. To find trustworthy information in online social data, it is increasingly important to automatically identify suspicious entities with unusual behavior. These entities can be either human actors e.g., spammers and fraudulent reviewers or resources e.g., spammed products or URLs.

Example 1. *Figure 1.1 shows an example of spamming reviewers in the setting of user-rating-products. Five users (represented by s_1 to s_5) and three products (represented by t_1 to t_3) with the edges carrying ratings on a scale of 5. Users in*

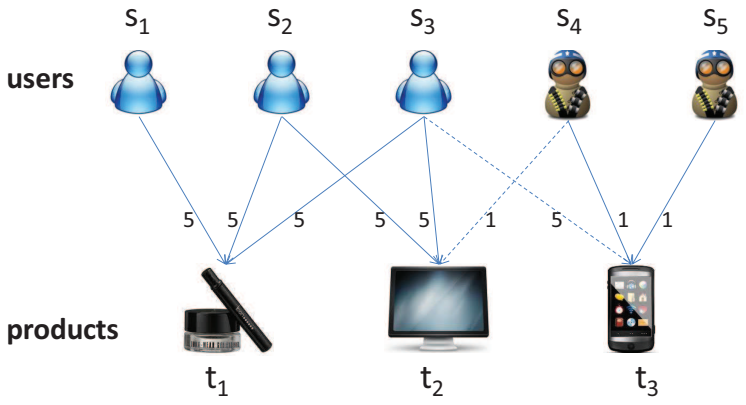

Figure 1.1: A toy example of five users rating three products.

$\{s_1, s_2, s_3\}$ are normal users who like products t_1 and t_2. s_2 also thinks that product t_3 is very good. However, products t_2 and t_3 that are given high ratings by normal users are given very low ratings by s_4 and s_5, who are possibly two spammers hired to demote t_2 and t_3.

In order to influence public opinion, spammers rarely operate with just a single account. Instead, they typically employ multiple accounts to collectively promote or demote certain targets, such as a URL or a product.

Example 2. *For example, in Twitter, a group of users may collaboratively spam on popular hashtags to promote their websites or businesses. Their strategy is to post a large number of tweets containing both their advertisement content and the popular hashtags, so that other users querying any of these hashtags would see their spamming tweets. These activities are classified as spamming according to Twitter's rules*[1]*. Figure 1.2 shows three collections S_1, S_2 and S_3 of real spammers in Twitter detected by our approach. As the unusually high usage of popular hashtags offers a clue for finding these spammers, we rank all the users in descending order by the number of times they used a hashtag. Here we show four popular hashtags (#london, #hongkong, #philipines and #mongolia), and the 34 top-ranked users out of 1899 for each hashtag. We observe that members of the three identified spammer collections are highly consistent in their heavy usage of the same set of hashtags. In contrast, for users in the blank cells in the top positions of each feature, despite their heavy hashtag usage, they are not found to be spammers because their usage*

[1] *http://support.twitter.com/articles/18311-the-twitter-rules*

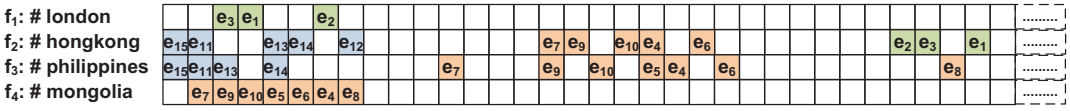

Figure 1.2: Three collections of real Twitter users ranked in descending order (left to right) by the usage frequency of four hashtags (labeled as f_1 to f_4).

patterns do not appear to be shared by other users. After reading through relevant tweets, we find that members of each collection post tweets with no real content other than a large number of hashtags appended with short URLs. It turns out that S_1 promotes travel guidance books for various cities including London and Hong Kong, corresponding to the hashtags they use for spamming. S_2 and S_3 promote the same pornography website in Brazil.

Example 3. *In an online marketplace, normal sellers would gradually accumulate their credit ratings by selling products to many normal buyers. However, a suspicious seller can create many low-price transactions with other "accomplice" buyer accounts in a short time to gain credibility, before performing fraud transactions involving large sums of money. This kind of fraudulent users with their unusual coalition with other accomplice accounts are prominent in many C2C and B2C business platforms [CW04] [PCWF07].*

These aforementioned Example 1, Example 2 and Example 3 of suspicious entities are anomalies in social data. In general, an anomaly is a data instance or subset of data instances which appears to be inconsistent with the remainder set of data [BL94]. Since anomalies can pose serious threats to the ecosystem of online social platforms, it is very important to detect them.

1.2 Research Objectives

Unlike many research topics on social data that have been well studied, including community detection, evolution analysis and link inference, the problem of anomaly detection in social data is still in its infancy [Agg11]. Nevertheless, this problem is related to fraud detection in e-commerce data [PCWF07], spammer detection in

online review data [MLG12] and anomaly detection in graphs in general [SQCF05] [AMF10].

Different from existing work, in this dissertation, we design unsupervised frameworks to mine anomalies in social data by their anomalous behaviors. As anomalous entities in social data often hold different agenda compared to normal ones, their unusual behaviors would give them away.

We focus on two types of anomalous behaviors in social data, namely, the unusual coalition among a collection of entities and the unusual conflict of opinions among human actors towards the same resource. Utilizing these anomalous behaviors, we are able to detect anomaly collections of the same entity type and node anomalies representing different types of entities in social data. Specifically, Example 2 shows unusual coalition, where user accounts of the same anomaly collection in Twitter hijack the same set of popular hashtags. Utilizing this kind of anomalous behavior, we are able to detect anomaly collections. On the other hand, Example 1 illustrates the conflicting views between $\{s_4\}$ and $\{s_1, s_2, s_3\}$ towards t_2, as well as those between $\{s_2\}$ and $\{s_4, s_5\}$ towards t_3. With this kind of anomalous behavior, we can detect node anomalies representing entities of different types in bipartite graphs.

1.2.1 Detecting Anomaly Collections

Anomalous behavior of unusual coalition is prominent in social data as shown in Example 2 and Example 3. To detect anomaly collections of unusual coalition, we propose two anomaly collection definitions *Extreme Rank Anomalous Collection(or ERAC)* and *Coherent Anomaly Collection (or CAC)*. Our novel anomaly definitions capturing the unusual coalitions is based on extreme rank, i.e., the members of an anomaly collection are consistently extremely ranked across multiple features. As the ERAC definition does not capture the coherence in the unusual behavior of members in an anomaly collection, we propose CAC to emphasize the *shared* anomalous behavior patterns among members, modeled by information theory.

For applications where entities are likely to participate in multiple anomalous collections, we propose to detect top-K ERACs. As top-K ERACs are often over-

lapping with each other, for applications where disjoint anomaly collections are of interest, we propose to find top-K disjoint CACs. For both detection problems, exact and heuristic algorithms are proposed with pruning techniques.

Existing anomaly detection approaches fall short in detecting aforementioned anomaly collections because they either focus on single point anomalies, or they are not optimized to detect collections by behaviors of unusual coalition among members of a collection[CBK09].

1.2.2 Detecting Node Anomalies in Graphs

Social data can be modeled as bipartite graphs with multiple types of entities (human actors and resources), including users-rating-products in online marketplaces, users-clicking-webpages on the World Wide Web and users-referring-users in social network platforms. Human actors in these social data often manifest anomalous behaviors such as expressing unusual conflicting views towards resources.

In Example 1, we observe that edge (s_4, t_2) and edge (s_2, t_3) do not agree with the majority opinion on target nodes t_2 and t_3 respectively. We can see that s_1, s_2 and s_3 are normal users who like products t_1 and t_2, but s_4 and s_5 are possibly two spammers hired to demote t_2 and t_3.

In general, anomalous nodes are the minority and are inconsistent with the rest of the nodes in the same partite. However, we cannot judge a node by its edges alone, instead we should also involve the linked nodes in the other partite. For instance, in Example 1 we cannot say s_2 is anomalous simply because he gives t_3 a rating of 5, which is a minority in all the ratings $\{1,1,5\}$ given to t_3. In fact, it is natural for s_2, a normal node, to give a minority rating to t_3, which is demoted by spammers.

Therefore, in bipartite graphs, the anomalousness of nodes in one partite often depends on that of their connected nodes in the other partite. Previous studies [Kle99] [WXLY11] [LLW08] have shown that this mutual dependency can be positive (the anomalousness of a node in one partite increases or decreases along with that of its connected nodes in the other partite) or negative (the anomalousness of a node in one partite rises or falls in opposite direction to that of its connected nodes

in the other partite).

In this dissertation, we unify both positive and negative mutual dependency relationships in an unsupervised framework to detect anomalous nodes in bipartite graphs. This is the first work that integrates both mutual dependency principles to model the anomalous behaviors of nodes that cannot be identified by either principle alone.

1.3 Contributions

1.3.1 Novel Anomaly Definition Based on Anomalous Behavior

Extreme Rank Anomalous Collection. We propose Extreme Rank Anomalous Collection or ERAC to describe a set of entities clustered at the top or bottom ranks of certain features. We propose a statistical model to measure the anomalousness of a collection by how extremely ranked it is with respect to any feature set.

Coherent Anomaly Collections. We propose the concept of Coherent Anomaly Collection or CAC based on ERAC with additional coherence criteria on the members of an anomaly collection. We design coherence checking method based on the idea of matrix encoding cost from information theory.

Node Anomaly in Bipartite Graph. We propose a new definition of node anomaly in bipartite graph by the anomalous behavior of unusual conflict of opinions among human actors towards resources.

1.3.2 Algorithms for Detecting Anomalies

ERAC Detection. We propose the problem of detecting top-K ERACs and develop both exact and heuristic algorithms with different pruning strategies, assuming a predefined ERAC size limit.

ERAC Expansion. We propose the problem of expanding the detected ERACs to uncover their supersets that are more anomalous. We design greedy algorithms to

solve the ERAC expansion problem without having to specify the size limit.

CAC Detection. We propose the problem of mining top-K disjoint CACs. We introduce an exact and a heuristic algorithms utilizing the properties of the p-values without the constraint on CAC size.

Node Anomaly Detection. We consider both the positive and negative mutual dependencies of anomalousness between nodes in bipartite graphs and design an iterative algorithm to define the anomaly scores of source and target nodes. We prove that with our algorithm, the rankings of source and target nodes converge.

1.3.3 Extensive Evaluation by Experiments.

On ERAC Algorithms. We design experiments to study the performance of our proposed ERAC algorithms for both ERAC detection and expansion problems in four datasets. Specifically, in synthetic datasets, both ERAC detection and expansion algorithms demonstrate high precisions. For the web spam dataset, both ERAC detection and expansion algorithms discover web spammer collections with higher precisions than existing approaches. For the IMDB dataset, both ERAC detection and expansion algorithms identify unusual actor collections that are not easily identified by clustering-based methods. For the Chinese online forum dataset, our ERAC detection algorithm identifies suspicious water army spammer collections agreed by human evaluators. ERAC expansion algorithm successfully reveals two larger spammer collections with different spamming behaviors.

On CAC Algorithms. We design synthetic data generation algorithms and study the performance of our CAC detection algorithm on synthetic data with existing baselines. The results show that our heuristic algorithm are both effective and efficient. We also apply our CAC detection algorithm on Twitter data to detect hashtag spammer collections, on a web spam dataset to detect groups of spamming websites, and on a Chinese online forum data to detect opinion spammer collections. The results on all real life datasets show that we are able to discover meaningful and informative spammer collections which are otherwise hard to find by existing

approaches.

On Node Anomaly Detection Algorithms. We generate synthetic bipartite graphs and compare our model with competitors that incorporate either positive or negative mutual dependency principle. Experiment results show that our model achieves much higher precision. We apply our model on two real life datasets: Goodreads and Buzzcity. The results show that we can identify suspected spamming users and spammed books in Goodreads. For the Buzzcity data with labeled ground truth, we also identify fraudulent IP addresses along with the fraudulent advertisement publishers in Buzzcity with higher precision than existing approaches.

1.4 Dissertation Organization

The following chapters of this dissertation are organized as follows:

- Chapter 2 reviews the existing studies on anomaly detections including point anomaly detection and collective anomaly detection, followed by existing work on detecting anomalies on graphs.
- Chapter 3 proposes the concept of Extreme Rank Anomalous Collection (or ERAC) and addresses the ERAC detection along with the follow-up ERAC expansion problems.
- Chapter 4 proposes the concept of Cohesive Anomaly Collection (or CAC) based on ERAC and addresses the problem of detecting CACs.
- Chapter 5 addresses the problem of detecting anomalous nodes in graph data with a novel iterative algorithm.
- In Chapter 6, we present our conclusions on the research results of this dissertation and an overview of future research directions.

Chapter 2

Related Work

In this chapter, we give an overview of the existing studies in the literature of anomaly detection. An anomaly is a data instance or subset of data instances which appears to be inconsistent with the remainder of that set of data [BL94]. It is also referred to as outlier, exception, peculiarity, and surprise [BL94].

An anomaly can be classified as point anomaly or collective anomaly. A point anomaly refers to an individual data instance that is inconsistent w.r.t. the rest of the data, whereas a collective anomaly or an anomaly collection refers to a collection of data instances that is inconsistent w.r.t. the rest of the data. Various techniques has been developed in various domains and on different types of datasets.

We first introduce the existing studies on detecting point anomalies, followed by those on detecting collective anomalies. We lastly focus on the work done in finding anomalies on graphs. For each type of existing studies, we compare them with our proposed approaches in this dissertation.

2.1 Point Anomaly Detection

Most of the studies on anomaly detection focus on point anomalies [CBK09], using a variety of approaches including classification-based, nearest-neighbor-based [BKNS00], statistical-based [BL94] and clustering-based approaches [EKSX96] and [GRS99]. Here we describe some representatives of each approach.

Classification-based.

The classification-based approach assumes that a classifier can be learnt from training data with ground truth of normal or anomalous instances to distinguish between normal and anomalous classes. [CDG+07] uses decision tree to predict whether a web host is a spammer based on predefined web host linkage features and content features. The ground truth of the spammers are manually labeled by Yahoo. The results show that decision tree achieves high precision. Other classification techniques including Bayesian Networks [WMCW03], Support Vector Machines [HLV03] and Rule Based [Agg05] are also applied to detecting disease outbreaks, and system call intrusions.

The biggest disadvantage of classification-based approach is that they require accurate labels for normal or abnormal classes, which are often not available.

Nearest-neighbor-based.

The nearest-neighbor-based approaches assume normal data instances occur in dense neighborhoods, while anomalies occur far from their closest neighbors. [KN98] calculates the anomaly score of a data instance by counting the number of nearest neighbors that are not more than d distance apart from the given data instance. [BKNS00] assigns the Local Outlier Factor to data instances, which is the ratio of average local density of the k nearest neighbors of the instance and the local density of the data instance itself.

These approaches are unsupervised and do not assume any generative distribution of the data. However, their assumption of anomalies may not be true for anomaly collections, which are locally dense.

Statistical-based.

Statistical techniques have also been applied to discover point anomalies on the assumption that normal entities occur in high probability regions of a statistical model, while anomalies occur in the low probability regions. This technique utilizes statistical inference tests to determine if an entity is generated by the statistical model [BL94]. For example, Grubb's test and the student's t-test have been applied under the assumption that the data is generated by a Gaussian distribution. If the assumptions regarding the underlying data distribution hold true, statistical techniques pro-

vide a statistically justifiable solution for anomaly detection. Unfortunately, these assumptions often do not hold, especially for high dimensional data.

Clustering-based.

The assumption taken by the clustering-based approach to detect point anomalies is, normal data instances belong to a cluster in the data, while anomalies either do not belong to any cluster. Based on this assumption, several clustering algorithms that do not require every data to belong to a cluster are developed including DBSCAN [SEKX98] and Findout [YSZ02]. However, they are not optimized for detecting anomalies but treating anomaly detection as a by product of clustering.

2.2 Collective Anomaly Detection

Since all these point anomaly detection approaches assume that anomalies appear in sparse regions or are far away from the normal entities, true anomalous collections that are dense or are close to the normal entities may escape detection. There are a few existing studies on detecting anomaly collections. They are mostly classification-based or statistical-based or clustering-based. Our proposed approaches for detecting ERAC and CAC belong to the category of collective anomaly detection.

Classification-based.

[DSN08] detects anomaly collections of relational records in categorical datasets with a supervised approach, assuming non-anomalous data points are labeled. Models are learned from the training data using Bayesian networks with no anomalies, and a set of records are flagged as an anomaly collection if they are individually anomalous, sharing the same value for a subset of features, and the number of records in the set is significantly larger than expected. However, due to the large amount of records in real datasets, it is not easy to get the ground truth of normal records. Furthermore, in many applications, members in an anomaly collection are not individually anomalous in the first place, and they may only have similar, instead of the same, values for a subset of features.

The work in [DSN08] is extended to detect a group of anomalous objects in

[KJD09]. A bayesian network is learned from the labeled normal data and another bayesian network of the same structure but different parameters is also learned and assumed to generate outliers. However, since anomaly collections are often of small sizes, their outlier generation model trained would suffer from over-fitting.

Statistical-based.

[ACCD11] uses a statistical model to detect in a network an anomaly collection in which the node features have a different distribution than the rest of the nodes. The work assumes there is only one anomalous cluster in the network, and each node is assumed to have one feature.

Clustering-based.

[LTZ10] detects anomaly collections by a data structure called isolation forest. The work assumes that after data points are projected to some hyperplane, the anomalous points follow one distribution while the normal points follow a different distribution. The two distributions are further assumed to be separable by minimizing their dispersions. However, the validity of these assumptions is arguable especially for datasets with multiple similar extreme patterns. Furthermore, the anomaly score is defined on point level instead of on collection level, which is not suitable for finding collections with shared extreme behavior.

Other work on detecting anomalous clusters includes [DXLL09], [HXD03] and [LTS04]. They assume that normal entities belong to large and dense clusters, whereas outliers form small or sparse clusters. Thus they apply a clustering algorithm on the data set, and declare small clusters that are below some size threshold as anomalous clusters. In [DXLL09] and [HXD03], small clusters are the smaller ones that together constitute less than 10% of the population. In [LTS04], the threshold is half of the average cluster size. These approaches also require a threshold for the size of anomalous clusters, which is not trivial to set. Furthermore, the small clusters identified do not necessarily exhibit extreme behavior like the spammer clusters.

Other approaches.

[MLW+11] and [MLG12] detect collections of product review spammers in Amazon. This work assumes each group of spam reviewers must have worked together

on at least certain number of products and uses frequent pattern mining to find candidate spam collections. However, it does not model the collective extreme behaviors, which in fact is a very common nature of spammer collections. [BXG+13] finds collections of "like" spammers in Facebook by searching for near bipartite-cores. However, they do not model coherent extreme behaviors in general.

Other work models the problem using burst detection [Kle02] and change-point detection [TY06], where formal models are proposed to study the normal behavior of a collections of data points on data streams or time series data. The collection of data points that are not necessarily generated by these models are declared as anomalies.

Comparison with our proposed approaches.
All above mentioned existing approaches are in general not specifically designed for detecting anomalous collections with abnormal behaviors such as extreme collective behaviors, which are prominent in online social networks. Our proposed ERAC and CAC detection approaches are able to identify collections of anomalous coalition behaviors with much higher precision.

2.3 Anomaly Detection on Graph data

Few anomaly detection studies focus on graph data. A graph anomaly is a data instance which appears to be inconsistent with the remainder of the data. One unique characteristics of graph data is that it contains graph structure, of which anomaly detection approaches should take good advantage.

Existing studies on graph anomalies mostly adopt two paradigms for factoring in the graph structure. One is extracting graph structure and then applying general anomaly detection method, whereas the other is directly exploiting the mutual dependencies among different types of nodes through links in graphs.

Structural feature based.
In [AMF10], a node is considered as anomalous if its neighborhood significantly differs from those of others. They focus on four types of node anomalies characterized by the 1-hop neighborhood (Egonet) in weighted graphs. The neighborhood

of a node is summarized by pairs of neighborhood features including the number of edges vs. the number of nodes in an egonet, which are assumed to follow power law distribution. The nodes whose neighborhoods deviate from the fitted power law curve are flagged as node anomalies.

[LC08] detects node anomalies in semantic graphs. The neighborhood of a node is summarized by paths of various length. A node is an anomaly if it carries abnormal semantic paths, which is discovered by the standard distance based anomaly detection techniques.

Anomalous subgraph patterns are studied in [NC03]. The basic assumption is that anomalies induce irregularities in the information content of the graph. They further assume regularities or normal patterns are the best substructures pattern in terms of Minimum Description Length (MDL) from information theory. Subgraph outliers are the ones experiencing less compressions by best substructure patterns.

In [SQCF05], node anomaly in bipartite graph is studied. A source/target node is anomalous if the average similarity of its 1 hop neighbors are low. For example, authors publish in conferences of different research communities (clusters) are anomalous. The similarity of nodes are then computed using random walk.

Mutual dependency based.

There are also studies investigating mutual dependency of different types of nodes in graphs, to discover anomalous nodes in graph data.

[Kle99] proposes HITS with the concepts of authority and hub and uses the mutual reinforcement principle to rank webpages. The assumption is that a good authority is linked by many good hubs and a good hub links to many good authorities. [YHY07] also uses mutual reinforcement principle to study the veracity of information on the web. This work assumes a web site is trustworthy if it provides many pieces of true information, and a piece of information is likely to be true if it is provided by many trustworthy web sites.

Salient terms and sentences are identified using positive mutual dependency principle in [Zha02]. They assume a term has high saliency score if it appears in many sentences with high saliency scores and a sentence has high saliency score if it

contains many terms with high saliency scores. [WLLH08] extends the model in [Zha02] and proposes mutual reinforcement chain of document, sentence and terms.

In [BLZ+09], users, questions and answers in the community question answering setting are modeled as three types of nodes. Coupled multiple mutual reinforcing relationships among the three types of nodes are utilized to detect high-quality answers, questions, and users. [WXLY11] finds suspicious reviewers in constructed review graph with three types of nodes including reviewers, reviews and stores by mutual dependency principle. Specifically, a reviewer is more trustworthy if (s)he has written more honest 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. The trustworthiness of reviewers, the honesty of reviews, and the reliability of stores are iteratively computed and thus the spam reviews are detected.

[LLW08] uses negative mutual dependency principle to detect biased users and controversial products in evaluation systems. A reviewer is more biased if he deviates more on less controversial objects and an object is more controversial if there is greater deviation by less biased reviewers.

Comparison with our proposed approaches.
We differ from the aforementioned structural feature based studies as we do not extract graph structure to numerical values, but utilize directly the graph structure to propagate the anomaly scores of both partites. Furthermore, in a bipartite graph, we detect anomalies in both partites simultaneously.

Moreover, the mutual dependency principles of the existing work can be summarized as positive [Zha02] [WLLH08] [BLZ+09] (the anomalousness of a node in one partite increases or decreases along with that of its connected nodes in the other partite) or negative [LLW08] (the anomalousness of a node in one partite rises or falls in opposite direction to that of its connected nodes in the other partite). Our framework is different from existing work as it incorporates both positive and negative mutual dependency principles and manages to identify node anomalies that are undetectable by either mutual dependency principle alone.

2.4 Summary

In this chapter, we describe existing studies on point anomaly detection, collective anomaly detection and node anomaly detection on graphs, which are related to our approaches proposed in this dissertation. For each type of existing work, we briefly analyze their ideas, techniques and applications. Furthermore, we describe the advantages of our approaches proposed in the following chapters compared to these existing ones.

Chapter 3

Detecting Extreme Rank Anomaly Collections

3.1 Introduction

When entities are ranked by a set of features, a small subset of the entities could cluster at the top or bottom ranks of some of these features. We call such an entity subset an *extreme rank anomaly collection* or *ERAC*.

ERACs are prominent in many real-life applications. For example, in online marketplaces, a fraudulent user may create many low-price transactions with other "accomplice" accounts in a short time to gain credibility, before performing fraud transactions involving large sums of money [CW04] [PCWF07]. Consequently, they are likely to rank at extreme positions with respect to features such as average number of transactions and transaction rate.

Web spammers are good examples of ERACs as well. As reported in [FMN04], [CDG+07] and [GGMP04], web spammers adopt many spamming strategies to boost rankings of their pages in search queries. One strategy used is to stuff the pages with popular keywords and anchor texts that are unrelated to one another. Web spammers may also generate pages from similar templates on the fly in order to perform "link spam". As a result, when measured by these features such as the number of popular keywords or the number of anchor texts in webpages, spam-

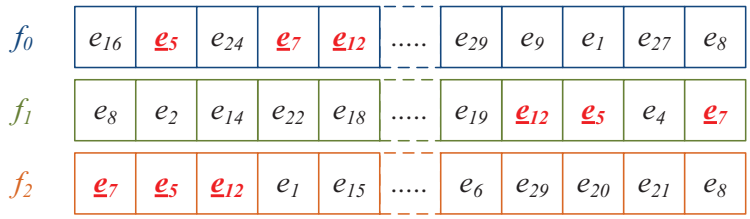

Figure 3.1: An example of ERAC. 30 entities $\{e_0,...,e_{29}\}$ are ranked according to each 3 features $\{f_0,f_1,f_2\}$. In this example, $\{e_5,e_7,e_{12}\}$ is an ERAC

mer hosts consistently demonstrate very extreme behaviors and form an identifiable anomalous collection, in contrast to normal web hosts.

To illustrate, suppose we have 30 web hosts $\{e_0,...,e_{29}\}$ as shown in Figure 3.1. The three host features $\{f_0,f_1,f_2\}$ reflect the aforementioned spamming strategies: f_0 represents the average number of popular keywords, f_1 is the variance of the word count, and f_2 captures the average ratio of anchor text to normal text. For each feature, we rank all the web hosts in descending order by their feature values. We can then identify $\{e_5,e_7,e_{12}\}$ as an ERAC because all of its entities appear at the top positions on features f_0 and f_2, and at the bottom positions on feature f_1. The fact that e_5, e_7 and e_{12} collectively display extreme behaviors across the three features is strong evidence that they are likely to be web spammers.

Existing anomaly detection approaches fall short in the above applications because they either focus on single point anomalies, or they are not optimized to detect collections with extreme characteristics. Note that a set of single point anomalies does not always form an extreme rank anomaly collection, because not every entity in an ERAC appears at extreme positions. For example, in Figure 3.1, e_{12} is not very extreme by itself although it is part of an extreme rank anomaly collection $\{e_5,e_7,e_{12}\}$. In contrast, e_8 appears at extreme positions on all three features, but it does not form an extreme cluster with any other entities. Our problem therefore cannot be solved by simply grouping the single point anomalies found by existing approaches.

We propose to model ERAC by the hypergeometric distribution. The anomalousness of an ERAC is quantified by a statistical p-value that captures the following general principles — A collection is more anomalous if: (I) It contains a larger number of entities that are ranked at extreme positions on some feature(s); and (II)

Its entities are consistently ranked at extreme positions across more features.

Due to the large number of ERACs of various sizes, we first tackle the **ERAC Detection problem** of discovering top-K ERACs with a predefined size limit, which is set to small values for efficiency reasons.

Nevertheless, after being offered with the top ERACs of a predefined size, users may want to see the most anomalous supersets of some detected top-K ERACs. For example, in the web spam case, users may find the detected ERAC $\{e_5, e_7, e_{12}\}$ of interest as they have the common spamming strategy of using lots of popular keywords, with very little variance on the word count, and high fraction of anchor text. It is natural to ask, can we detect the superset of this ERAC that are even more anomalous with similar sets of spamming strategies? Therefore, we also propose the **ERAC Expansion problem** to uncover the supersets of the detected ERACs that are more anomalous than the original top-K ones. Unlike the ERAC detection problem, ERAC expansion is done without predefined size constraints.

The rest of the chapter is organized as follows. After introducing the problem formulation in Section 3.2, Section 3.3 presents our ERAC detection algorithms for both independent and dependent feature sets. Section 3.4 describes our ERAC expansion algorithms. Section 3.6, Section 3.7, Section 3.8 and Section 3.9 report on experiments on both ERAC detection and expansion. Finally, Section 3.10 concludes the chapter.

This chapter is based on our publication in SIAM International Conference on Data Mining (SDM 2012) [DZLP12b].

3.2 Extreme Rank Anomaly Collection

In this section, we start by introducing related concepts regarding the anomalousness or extremity of a collection on a single feature. We then describe the anomalousness definition for multiple features. We also formally define ERAC and ERAC detection problem here in this section. The frequently used notations are summarized in Table 3.2.1.

Let E denote the universal entity set, and F a set of features. For an entity

Notation	Meaning	Notation	Meaning
E	the universal entity set	F	a feature set $\{f\}$
S	an entity collection	r	an extremity index
$rank_f(e)$	the ranking of e w.r.t. f	$S_f(r)$	entities in S within r on f
$p_f(S,r)$	p-value of S w.r.t. r on f	$\widehat{p}_f(S)$	S' representative p-value on f
$r_f(S)$	representative r of S on f		

Table 3.1: Notations

$e \in E$, $e.f$ denotes the value of e for feature $f \in F$. Entities can be ranked with respect to any feature. $rank_f(e)$ denotes the rank of entity e in E w.r.t. feature f, which is simplified to $rank(e)$ when the context is clear. For example, in Figure 3.1, $rank(e_8) = 1$ and $rank(e_7) = 30$ on feature f_1, assuming all feature values are distinct.

An entity may lie in top or bottom rank positions on different features. For the purpose of exposition, the following discussion focuses on the top rank position case. Our analysis extends readily to the bottom rank position case.

3.2.1 Measuring Anomalousness for A Single Feature

We begin by measuring how anomalous an entity collection is w.r.t. a single feature. Our approach is based on the following principle: *on a given feature* f, an entity collection S is more anomalous, or more extremely ranked, if more entities in S appear in extreme regions of the ranked list of the universal entity set E w.r.t. f.

Given a set of entities S, a feature f, we use an **extremity index** r to refer to an extreme region, and use $S_f(r)$ to denote the set of entities in S which appear in the top r rank positions w.r.t. feature f,

$$S_f(r) = \{e \mid \forall e \in S, rank_f(e) \leq r\}$$

Continuing the example in Figure 3.1, for $S = \{e_5, e_7\}$ and $r = 2$, we have $S_{f_0}(r) = \{e_5\}$. Similarly, for $S = E$ and extremity index $r = 3$, $E_{f_1}(r) = \{e_8, e_2, e_{14}\}$. Note that $|E_f(r)| = r$ when the entities have distinct feature values. Where there is a tie, $|E_f(r)|$ may be larger than r.

The extremity of any given entity collection $S \subset E$ can be derived based on $|S_f(r)|$, which is in fact $|S \cap E_f(r)|$. In general, if we randomly pick $|S|$ entities

from $|E|$ entities, the number of entities in S that also belong to top or bottom r of the entity list, (i.e., $E_f(r)$) follows the hypergeometric distribution. Intuitively, we can imagine there are totally $|E|$ balls with r balls of red and $|E|-r$ balls of black. We draw $|S|$ number of balls from this $|E|$ balls and the number of times we get red balls follows the hypergeometric distribution. Thus the probability of observing $|S_f(r)|$ common entities shared by S and $E_f(r)$ can be computed as in the hypergeometric distribution as follows:

$$prob(|S_f(r)|,|E|,|E_f(r)|,|S|) = \frac{\binom{|E_f(r)|}{|S_f(r)|} \cdot \binom{|E|-|E_f(r)|}{|S|-|S_f(r)|}}{\binom{|E|}{|S|}}$$

We now define the **p-value**[1] of S w.r.t. extremity index r and feature f, denoted as $p_f(S,r)$, as the probability of observing at least $|S_f(r)|$ common entities between a random collection of size $|S|$ and $E_f(r)$.

$$p_f(S,r) = \sum_{i=|S_f(r)|}^{\min(|E_f(r)|,|S|)} prob(i,|E|,|E_f(r)|,|S|)$$

Thus, a given S leads to different p-values with different r. For any given r and f, the smaller the p-value of S, the more extremely ranked S is w.r.t. r. Therefore, among all the choices of r, we pick the one which gives the smallest p-value. This particular r measures the maximum extremity of ranking that S could possibly have, which is by our definition also the maximum anomalousness of S w.r.t. f. Formally, we call this r the **representative extremity index** of S w.r.t. f, which is denoted as $r_f(S)$ and defined as follows,

$$r_f(S) = argmin_{0<r<|E|/2}\, p_f(S,r)$$

The choice of r ranges from 1 to less than half of the population of E, because we now focus on the top rank position case. The bottom rank position case can be derived similarly.

Correspondingly, the **representative p-value** of S w.r.t. f is defined as $\widehat{p}_f(S) = p_f(S, r_f(S))$.

[1]The p-value defined here is the right ended p-value of the hypergeometric distribution. The right ended p-value is used instead of the left ended one because in our case the more (instead of less for the left ended case) S overlaps with $E_f(r)$, the more extremely ranked S is w.r.t. f.

Referring to Figure 3.1, for $S = \{e_5, e_7\}$, $r = 2$ and $r' = 4$, we have $|S_{f_0}(r)| = 1$ and $|S_{f_0}(r')| = 2$. Moreover, $p_{f_0}(S,r) = \sum_{i=1}^{\min(2,2)} prob(i,30,2,2) = 0.131$ and $p_{f_0}(S,r') = \sum_{i=2}^{\min(4,2)} prob(i,30,4,2) = 0.013$. As $p_{f_0}(S,\ r') < p_{f_0}(S,r)$, r' better represents how extremely ranked S is than r does for f_0. Suppose $r_{f_0}(S) = r' = 4$, we have the representative p-value of S w.r.t. f_0 as $\widehat{p}_{f_0}(\{e_5, e_7\})) = 0.013$.

Statistically, the representative p-value reflects how anomalous (i.e. extremely ranked) a collection S is w.r.t. a particular feature f. The smaller the representative p-value, the more anomalous the collection is. For example, in Figure 3.1, we compare $S = \{e_5, e_7\}$ with $S' = \{e_7, e_{12}\}$ by their representative p-values for f_0. $\widehat{p}_{f_0}(\{e_7, e_{12}\}) = p_{f_0}(\{e_7, e_{12}\}) = 0.023$. Since $\widehat{p}_{f_0}(\{e_5, e_7\}) = 0.013$ is smaller than $\widehat{p}_{f_0}(\{e_7, e_{12}\})$, we conclude that $\{e_5, e_7\}$ is more anomalous than $\{e_7, e_{12}\}$ w.r.t. f_0; this is intuitive as $\{e_5, e_7\}$ sits more towards the top positions than $\{e_7, e_{12}\}$ on f_0. Note that the p-value measures directly at the collection level, which is different from measuring on the entity level followed by aggregating across the individual measures.

We are now ready to give our definition for an *Extreme Rank Anomaly Collection*.

Definition 1. ***[Extreme Rank Anomaly Collection (ERAC)]*** *Given a universal entity set E and an entity set S s.t. $S \subset E, 1 < |S| < |E|/2$, a set of independent features F and a threshold α, we say S is an* Extreme Rank Anomaly Collection *w.r.t. F if (I) $\exists F^S \subseteq F$ such that $|F^S| > 1$ and $\forall f \in F^S, \widehat{p}_f(S) \leq \alpha$; (II) $1 < |S| < |E|/2$.*

The definition of ERAC is based on a global null hypothesis of multiple hypothesis tests, where each test is associated with one feature. We say S is anomalous w.r.t. F, if the derived p-values are smaller than a predefined significance level[2] α in at least two tests.

Note that when we reject the null hypothesis and say that S is anomalous, S may not be significant for every feature. We call F^S the significant features of S.

We impose the condition $1 < |S| < |E|/2$, as an anomalous collection should

[2]In order to control the type 1 error (false positive), the significance level for each individual test should be adjusted. Our model can accommodate all existing adjustment techniques including Bonferroni Correction, Holm-Bonferroni and Westfall-Young step-down. We adopt the Bonferroni Correction [Dun55] to adjust the significance level to $\alpha/|F|$ to be conservative. For example, assuming 0.05 is the intended significance level for each single test, α would be set to $0.05/|F|$.

contain more than one entity and yet remain the minority of the population.

The definition also requires a set of independent features F. We define the **dependency** of any two features $f, f' \in F$ based on the statistic of Kendall Tau rank correlation coefficient [Ken48].

$$dep(f, f') = \frac{|3(n_c - n_d)|}{\sqrt{|E| \cdot (|E|-1) \cdot (2|E|+5)/2}}$$

Where n_c is the number of concordant pairs of entities and n_d is the number of discordant pairs of entities in the entity lists of f and f'. [3]

3.2.2 Measuring Anomalousness for Multiple Features

To measure how extreme an entity collection is ranked on a set of features, we generalize our principle as follows: *For a given independent feature set F, a collection is more anomalous if it is extremely ranked for more features in the given feature set.*

As the representative p-value measures how anomalous an ERAC is for a single feature, we define the **anomaly score** of an ERAC S for F as the product of the representative p-values for all the significant features in F. As the probability value tends to be very small, we take the log form:

$$\Omega(S, F) = -\sum_{f \in F^S} \log \widehat{p}_f(S)$$

This definition is consistent with the principle that the more features S is extremely ranked against, the more anomalous it is.

With the anomaly score defined, we can now formulate our *ERAC detection problem* as follows:

Definition 2. ***[ERAC Detection Problem]*** *Given an entity universe set E, an independent feature set F, a target collection size N ($N < |E|/2$) and K, find the top-K most anomalous Extreme Rank Anomalous Collections of size at most N.*

[3] As the dependency statistic follows the standard normal distribution, we just need to compute the z-score value λ corresponding to any given significance level of a two tailed test. For example, if the significance level is 0.05, the z-score is 1.96. Therefore, given f and f', if the $dep(f, f')$ score is not greater than λ, then f and f' are independent of each other.

As there are potentially large number of ERACs in a large population E, we focus on the top-K ERACs. We require a predefined size limit N of small values, as the anomalous collections are by definition the minority in the population anyway. We do not require the elements in different extreme rank anomaly collections to be mutually exclusive, as many entities may join different anomalous collections with different extreme behaviors. Therefore, a larger K may reveal additional extreme patterns that are less anomalous.

3.3 ERAC Detection Algorithms

To find the top-K ERACs of size at most N, a naive way is to enumerate all collections of size up to N, sort them by anomaly score in decreasing order, and return those with the top-K ranks. This approach is infeasible as the search space is exponential in the size of E, i.e., $\binom{|E|}{2} + ... + \binom{|E|}{N}$. Therefore we propose a bottom-up approach that successively generates larger ERAC candidates from smaller ones, maintaining a current top-K list and pruning unpromising growth paths whenever possible.

Out of this approach arise two main challenges. *(I) How to generate candidates from the current set of ERACs of smaller sizes?* We adopt a classic approach used in the Apriori Algorithm [AS94] to generate candidates for frequent item-sets. Candidates are generated level by level from single entities to collections of size N. To generate a collection of size $n+1$, we only combine two collections of size n sharing the same first $n-1$ entities, assuming that entities in each collection are sorted alphabetically; this has been shown to ensure unique generation of each candidate collection. *(II) How to prune unnecessary growth paths and generate fewer candidates?* A crucial technique used in Apriori is to prune any current candidate from future consideration whenever it is found to fall below the threshold set by the last collection in the current top-K list, thus avoiding potentially traversing of the entire exponential search space.

Unfortunately, in general, the anomaly score of ERAC does not enjoy this downward closure property to support the standard pruning strategy in top-K computa-

tion. Specifically, for any collection S, even when the anomaly score $\Omega(S,F)$ is less than the least one in the current top-K list, we cannot conclude that for all super-sets S' of S, the anomaly score $\Omega(S',F) \leq \Omega(S,F)$ and therefore safely prune S. The absence of this monotonicity on the anomaly score poses a difficulty to our bottom-up search approach. It seems that we would actually have to keep all collections of size n to generate candidates of size $n+1$ to guarantee the completeness of the mining result.

To tackle this problem, we develop an exact algorithm in Section 3.3.1 and two heuristic algorithms in Section 3.3.2 and 3.3.3. The more general case of dependent features is discussed in Section 3.3.4.

3.3.1 An Exact Algorithm ERAC_E

Given a collection S, despite the absence of monotonicity on the anomaly score which precludes setting an upper-bound on $\Omega(S')$ for all super-sets S' of S, it is possible to derive an upper-bound on $\Omega(S')$ for those super-sets of S of a given size n. We denote this **upper-bound with size-constraint** n as $\hat{\Omega}(S,F,n)$, which will be formally defined shortly. Intuitively, the most anomalous super-set S' of S can be formed by adding to S exactly $|S'|-|S|$ entities which are ranked the most extreme positions.

Formally, given a collection S, size n, $(|S| < n \leq |E|/2)$, and a feature f, the most anomalous super-set of S is denoted as $\hat{S}_n$, and defined as $\hat{S}_n(f) = S \cup S^*$, where $|S^*| = n - |S|$ and $\forall e' \in S^*, \forall e \in E \setminus \hat{S}_n, rank_f(e') \leq rank_f(e)$.

To illustrate with the example in Figure 3.1, suppose $S = \{e_5, e_{12}\}$. Assuming no tie in ranking, we have $\hat{S}_4(f_0) = \{e_{16}, e_5, e_{24}, e_{12}\}$ because, on f_0, e_{16} and e_{24} are the two entities ranked the most extreme in the top positions, excluding the entities already in S.

Accordingly, given F and S, the upper-bound with size-constraint n for S is defined as:

$$\widehat{\Omega}(S,F,n) = -\sum_{f \in F} \log \widehat{p}_f(\hat{S}_n(f))$$

Before we present the theorem to show that $\widehat{\Omega}(S,F,n)$ thus defined indeed repre-

sents the upper-bound on the anomaly score of all super-sets of S of size n, we first introduce a property of p-value that is needed to establish the theorem.

Property 1. *Given any feature f, collections S and S' and extremity indices r and r', if $|S| = |S'|$, $|E_f(r)| = |E_f(r')|$ and $|S_f(r)| > |S'_f(r')|$, then $p_f(S,r) < p_f(S',r')$.*

Proof. $p_f(S,r) = \sum_{i=|S_f(r)|}^{\min(|E_f(r)|,|S|)} prob(i,|E|,|E_f(r)|,|S|)$ and $p_f(S',r') = \sum_{i=|S'_f(r')|}^{\min(|E_f(r')|,|S'|)} prob(i,|E|,|E_f(r')|,|S'|)$. Since $|S| = |S'|$, $|E_f(r)| = |E_f(r')|$ and $|S_f(r)| > |S'_f(r')|$, $p_f(S',r') = p_f(S,r) + \sum_{i=|S'_f(r')|}^{|S_f(r)|-1} prob(i,|E|,|E_f(r)|,|S|)$. As $prob(i, |E|,|E_f(r)|,|S|) > 0$, we have $p_f(S,r) < p_f(S',r')$. □

Property 1 can be explained as, with all other parameters kept constant, the larger the number of entities in S that fall into the extreme positions, the smaller the p-value is. Now we have Theorem 1.

Theorem 1. *Given S, $0 < |S| < |E|/2$, $\forall S'$ such that $S \subset S'$ and $|S'| < |E|/2$, we have $\Omega(S',F) \leq \widehat{\Omega}(S,F,|S'|)$*

Proof. Suppose S and S' are two collections s.t. $S \subset S'$. To prove $\Omega(S',F) = -\sum_{f\in F} \log \widehat{p}_f(S') \leq \widehat{\Omega}(S,F,|S'|) = -\sum_{f\in F} \log \widehat{p}_f(\hat{S}_{|S'|}(f))$, we need to show that for any f, $\widehat{p}_f(S') \geq \widehat{p}_f(\hat{S}_{|S'|}(f))$. Let $\widehat{S}$ denote $\hat{S}_{|S'|}(f)$.

Since $|\widehat{S}| = |S'|$, $E_f(r_f(S')) = E_f(r_f(S'))$ and $|\widehat{S} \cap E_f(r_f(S'))| \geq |S' \cap E_f(r_f(S'))|$. According to Property 1, we have $p_f(S',r_f(S')) \geq p_f(\widehat{S},r_f(S'))$. By definition, $\forall 0 < r < |E|/2, p_f(\widehat{S},r_f(\widehat{S})) \leq p_f(\widehat{S},r)$. So we have $p_f(\widehat{S},r_f(\widehat{S})) \leq p_f(\widehat{S},r_f(S'))$. Thus $p_f(\widehat{S},r_f(S')) \geq p_f(\widehat{S},r_f(\widehat{S}))$. Therefore, $p_f(S',r_f(S')) \geq p_f(\hat{S}_{|S'|}(f), r_f(\hat{S}_{|S'|}(f)))$, and $\Omega(S',F) \leq \widehat{\Omega}(S,F,|S'|)$. □

Based on the upper-bound with size-constraint, we propose an exact algorithm that incorporates the following pruning strategy. At any time in the bottom-up search process, there are three data structures in the system: (I) the current top-K list; (II) the set of ERACs of size n, which are used to generate candidates of size $n+1$; and (III) the set of single entities which have yet to grow into any collection. Denote Ω_t as the anomaly score of the least anomalous collection in the current top-K list.

Pruning Technique 1. *In generating candidates of size $n+1$, only those collections S with anomaly score upper-bound for size-constraint $n+1$, i.e., $\hat{\Omega}(S,F,n+1)$, larger than Ω_t are grown.*

Why is this pruning strategy sound? Observe that due to the absence of downward closure property, a standard Apriori-style bottom-up search cannot drop any collection S even if its anomaly score falls below Ω_t. On the other hand, the upper-bound derived directly from the size N, i.e., $\hat{\Omega}(S,F,N)$, would almost always prevail over Ω_t, leaving us with little pruning power. In contrast, with our pruning strategy, the upper-bound is computed with a size-constraint which increases in tandem with the size of candidates being generated. Consequently, (I) when the size is small, an upper-bound with the same small size-constraint is more likely to fall below Ω_t; (II) as the size grows, Ω_t increases monotonically as well, continually pushing the bar higher for the upper-bound to beat. This accounts for the greater pruning power of our method. The ERAC_E algorithm for computing top-K ERACs of size up to N is shown in Algorithm 1, followed by a running example in Table 3.2 to illustrate the pruning techniques applied on the example in Figure 3.2.

Algorithm 1 ERAC_E for independent features

Input: E, F, K, N
Output: Top-K ERACs: $\mathbb{S}^*$
{$\mathbb{S}^*$ is implemented by a priority queue of max length K. Ω_t is the smallest anomaly score of collections in $\mathbb{S}^*$, and is set to zero if $\mathbb{S}^* = \emptyset$. $\mathbb{S}(i)$ is the current selected collections of size i, implemented by a hash tree.}

1: $n = 1$; $\mathbb{S}^* = \emptyset$; $\mathbb{S}(i) = \emptyset$, for $i = 1..N$; $\mathbb{S} = \{\{e\} \mid e \in E\}$
2: **while** $\mathbb{S} \neq \emptyset$ && $n < N$ **do**
3: update $\mathbb{S}^*$ by elements in $\mathbb{S}$ {Ω_t is updated accordingly}
4: $\mathbb{S} = \{\{e\} \mid e \in E\} - \mathbb{S}(1)$
5: **for** $i = 1$ to n **do**
6: $\mathbb{S} = \{S \in \mathbb{S} \mid \widehat{\Omega}(S,F,n+1) > \Omega_t\}$ {$\mathbb{S}$ now keeps the set of eligible collections}
7: $\mathbb{S}(i) = \mathbb{S} \cup \mathbb{S}(i)$
8: $\mathbb{S} = join(\mathbb{S}(i), \mathbb{S}(i))$ {generate new candidates of size $i+1$, and update $\mathbb{S}^*$ accordingly by elements in $\mathbb{S}$}
9: $n{+}{+}$
10: **return** $\mathbb{S}^*$

Step 6 of Algorithm 1 applies Theorem 1 in excluding from $\mathbb{S}$ those collections with upper-bounds smaller than the threshold. Moreover, through $join()$, the supersets of collections with anomaly score upper-bounds smaller than the threshold are

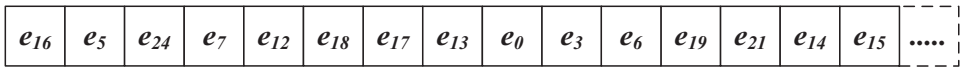

Figure 3.2: Top 15 entities ranked according to f_0 in Figure 3.1

n	$\mathbb{S}(i)$	$\mathbb{S}^*$ (Ω_t)
1	$\mathbb{S}(1) = \{\{e_{16}\},\{e_5\},\{e_{24}\},\{e_7\},\{e_{12}\}\}$	$\{\{e_{16}\}\}$ (3.4)
2	$\mathbb{S}(1)$: same as above $\mathbb{S}(2) = \{\{e_{16},e_5\},\{e_{16},e_{24}\},\{e_{16},e_7\},$ $\{e_{16},e_{12}\},\{e_5,e_{24}\},\{e_5,e_7\},\{e_5,e_{12}\},$ $\{e_{24},e_7\},\{e_{24},e_{12}\},\{e_7,e_{12}\}\}$	$\{\{e_{16},e_5\}\}$ (6.08)
3	$\mathbb{S}(1)$: same as above $\mathbb{S}(2)$: same as above $\mathbb{S}(3) = \{\{e_{16},e_5,e_{24}\},\{e_{16},e_5,e_7\},$ $\{e_5,e_{24},e_7\}\}$	$\{\{e_{16},e_5,e_{24}\}\}$ (8.31)

Table 3.2: Algorithm 1 running on the example shown in Figure 3.2

excluded from $\mathbb{S}$. The $join(\mathbb{S}(i),\mathbb{S}(i))$ function compute $S_1 \bigotimes S_2$ for each (S_1,S_2) pair derived from $\mathbb{S}(i)$, where the operation $\bigotimes$ combines two size-i collections with identical $i-1$ elements to a size-$(i+1)$ collection (and is implemented similarly as in the Apriori Algorithm [AS94]).

Running Example. Figure 3.2 shows the top 15 entities ranked by f_0 out of a universe of 30 entities. Suppose K = 1 and N= 3. Table 3.2 shows the execution of Algorithm 1 with the changes in $\mathbb{S}(i)$, $\mathbb{S}^*$ and Ω_t.

When $n = 1$, $\mathbb{S}^*$ is updated to keep the current most anomalous collection after Step 3: $\{e_{16}\}$. Ω_t is updated to $\Omega(\{e_{16}\},\{f_0\}) = -\log \widehat{p}(\{e_{16}\}) = 3.40$. At Step 4, since $\mathbb{S}(1) = \emptyset$, $\mathbb{S}$ contains all the singular sets. At Step 6, the set of selected collections of size 1 is $\mathbb{S} = \{\{e_{16}\},\{e_5\},\{e_{24}\},\{e_7\},\{e_{12}\}\}$ by comparing their upper-bounds and Ω_t. $\{e_{18}\}$ is not eligible as $\widehat{\Omega}(\{e_{18}\},\{f_0\},2) = -\log \widehat{p}(\{e_{16},e_{18}\}) = 3.37 < \Omega_t$. In Step 8, we get $\mathbb{S}$ containing 10 collections. When $n = 2$, $\mathbb{S}^*$ is updated to $\{\{e_{16},e_5\}\}$ and $\Omega_t = \Omega(\{e_{16},e_5\},\{f_0\})$ =6.08. $\mathbb{S}$ contains the remaining singular collections from $\{e_{18}\}$ to $\{e_{15}\}$ in the ranked list. This time i goes from 1 to 2. When $i = 1$, at Step 6, the algorithm tries to pick out the previous leftover singular entities that may be selected to generate collections of size 3. However, even $\widehat{\Omega}(\{e_{18}\},\{f_0\},3) = -\log \widehat{p}(\{e_{16},e_5,e_{18}\}) = 5.31 < \Omega_t$, meaning none of them is selected for now. When $i = 2$, in Step 6, we get $\mathbb{S} = \{\{e_{16},e_5\},\{e_{16},e_{24}\},\{e_{16},e_7\},\{e_5,e_{24}\},\{e_5,e_7\},\{e_{24},e_7\}\}$. After Step 8, $\mathbb{S} = \{\{e_{16},e_5,e_{24}\},\{e_{16},e_5,e_7\},\{e_5,e_{24},e_7\}\}$.

Finally, we get the top-1 ERAC of size no greater than 3 on feature f_0: $\{e_{16}, e_5, e_{24}\}$ with anomaly score 8.31. Note that if we had set $N = 6$, previous leftover singular entities e_{18} would be selected and the corresponding nodes in the lattice tree would be grown from level 1 to level 6. For simplicity, we only show the results for $N = 3$.

In this example, a standard Apriori algorithm without Pruning Strategy 1 would have to traverse the entire search space of all the candidates up to size 3, visiting altogether $\binom{15}{1} + \binom{15}{2} + \binom{15}{3}$=575 nodes in the search lattice, due to the absence of downward closure property. In comparison, we only visit 5+10+3=18 nodes in total, saving the visit to 96% of the nodes even in this small example.

Efficient Anomaly Score Computation. By the definition of $\Omega(S,F)$, for a given S and each feature $f \in F$, we need to compute $\widehat{p}_f(S)$, which is decided by the representative extremity index of S w.r.t f. The naive approach to find this representative extremity index would examine all $|E|/2$ possible indices. However, the following property of the p-value allows us to avoid checking all the extremity indices.

Property 2. *Given any feature f, collections S and S', and extremity indices r and r', if $|S| = |S'|$, $|S_f(r)| = |S'_f(r')|$ and $|E_f(r)| > |E_f(r')|$, then $p_f(S,r) > p_f(S',r')$.*

Proof. Suppose $|E_f(r)| = |E_f(r')| + 1 = r_0 + 1$ and $|S_f(r)| = |S'_f(r')| = i_0$, then we have $1 - p_f(S,r) = \sum_{i=0}^{i_0-1} \frac{\binom{r_0+1}{i} \cdot \binom{|E|-r_0-1}{|S|-i}}{\binom{|E|}{|S|}}$. Thus $(1-p_f(S,r)) \cdot \binom{|E|}{|S|} = \sum_{i=0}^{i_0-1} \binom{r_0+1}{i} \cdot \binom{|E|-r_0-1}{|S|-i}$. According to Pascal's triangle, $(1-p_f(S,r)) \cdot \binom{|E|}{|S|} = \sum_{i=0}^{i_0-1} (\binom{r_0}{i} + \binom{r_0}{i-1}) \cdot \binom{|E|-r_0-1}{|S|-i} = \sum_{i=0}^{i_0-1} \binom{r_0}{i} \cdot \binom{|E|-r_0-1}{|S|-i} + \sum_{i=0}^{i_0-1-1} \binom{r_0}{i} \cdot \binom{|E|-r_0-1}{|S|-i-1}$. Similarly, $(1-p_f(S',r')) \cdot \binom{|E|}{|S|} = \sum_{i=0}^{i_0-1} \binom{r_0}{i} \cdot \binom{|E|-r_0}{|S|-i}$. According to Pascal's triangle, $(1-p_f(S',r')) \cdot \binom{|E|}{|S|} = \sum_{i=0}^{i_0-1} \binom{r_0}{i} \cdot (\binom{|E|-r_0-1}{|S|-i} + \binom{|E|-r_0-1}{|S|-i-1}) = \sum_{i=0}^{i_0-1} \binom{r_0}{i} \cdot \binom{|E|-r_0-1}{|S|-i} + \sum_{i=0}^{i_0-1} \binom{r_0}{i} \cdot \binom{|E|-r_0-1}{|S|-i-1}$ $= (1-p_f(S,r)) \cdot \binom{|E|}{|S|} + \binom{r_0}{i_0-1} \cdot \binom{|E|-r_0-1}{|S|-(i_0-1)-1}$. Thus, $p_f(S,r) = p_f(S',r') + \frac{\binom{r_0}{i_0-1} \cdot \binom{|E|-r_0-1}{|S|-(i_0-1)-1}}{\binom{|E|}{|S|}} = p_f(S',r') + \frac{\binom{|E_f(r)|-1}{|S_f(r)|-1} \cdot \binom{|E|-|E_f(r)|}{|S|-|S_f(r)|}}{\binom{|E|}{|S|}}$. Generally, for any given $E_f(r)$ and $|E_f(r')$ such that $|E_f(r)| > |E_f(r')|$, $p_f(S,r) = p_f(S',r') + \sum_{j=0}^{|E_f(r)|-|E_f(r')|-1} \frac{\binom{|E_f(r)|-1-j}{|S_f(r)|-1} \cdot \binom{|E|-|E_f(r)|+j}{|S|-|S_f(r)|}}{\binom{|E|}{|S|}}$. As $\sum_{j=0}^{|E_f(r)|-|E_f(r')|-1} \frac{\binom{|E_f(r)|-1-j}{|S_f(r)|-1} \cdot \binom{|E|-|E_f(r)|+j}{|S|-|S_f(r)|}}{\binom{|E|}{|S|}} > 0$, we hence have $p_f(S,r) > p_f(S',r')$. □

Property 2 suggests that with all the other parameters kept constant, the smaller the extremity index, the smaller the p-value is.

With Property 2, we need to check only those extremity indices corresponding to the ranking of each entity in S. For example, in Figure 3.1, for $S = \{e_5, e_7\}$ and f_0, we just check $r = 2$ and $r' = 4$; the other extremity indices can be skipped. Take $r'' = 3$ for example, as $|S_{f_0}(r'')| = |S_{f_0}(r)|$ but $|E_f(r'')| = 3 > |E_f(r)| = 2$, according to Property 2, we have $p_{f_0}(S,r) < p_{f_0}(S,r'')$. Thus, we do not need to consider r''. Similarly, all other extremity indices that do not correspond to the rank of any entity in S can be proven to have larger p-values than the extremity index corresponding to the rank of some entity in S. Therefore, to compute the representative p-value $\widehat{p}_f(S)$ for each $f \in F$, we examine only $O(|S|)$ extremity indices instead of $O(|E|)$.

Time Complexity. As proposed in [Wu93], the p-value $p_f(S,r)$ can be calculated in $O(\min(|E_f(r)|, |S|))$ steps by recursion and factorial acceleration. Thus, the total time complexity of computing $\Omega(S,F)$ is $O(|S|^2 \cdot |F|)$, assuming $\min(|E_f(r)|, |S|) = |S|$. Similarly, by definition the time complexity of $\widehat{\Omega}()$ is on the same order as $\Omega(S,F)$. Step 6 takes $O(|\mathbb{S}| \cdot n^2 \cdot |F|)$. Step 8 can be implemented by a hash tree and thus is of $O(|\mathbb{S}(i)|^2)$. The size of $\mathbb{S}(i)$ is data dependent, which in the worse case is $|E|^i$. Let $|\overline{\mathbb{S}}|$ denote the average size of $\mathbb{S}(i)$ for all i and all n. The running time of the "for" loop is of $O(|\overline{\mathbb{S}}|^2 + |\overline{\mathbb{S}}| \cdot n^3 \cdot |F|)$. Therefore, the total running time of Algorithm 1 is of $O(|\overline{\mathbb{S}}|^2 + |\overline{\mathbb{S}}| \cdot N^4 \cdot |F|)$.

3.3.2 A Naive Heuristic Algorithm: ERAC_N

The number of seeds selected in Step 6 of the ERAC_E algorithm may be large and it may grow exponentially as the collection size increases. To further prune the potential candidates, we take the naive heuristic that only the top-m most anomalous potential candidates are selected as seeds to generate the collections of larger sizes.

With this heuristic, we add one more step: "$\mathbb{S}$= top $m \cdot |E|$ most anomalous collections in $\mathbb{S}$ " after Step 6 in Algorithm 1 with $0 < m < 1$. We thus obtain ERAC_N, a naive heuristic algorithm to detect top-K ERACs with independent feature set with m as an additional parameter.

Since the naive heuristic fixes the number of seeds for collections of all sizes, it is expected to be much faster than the exact algorithm. Suppose on average we

have $m \cdot |E|$ number of collections producing at most $(m \cdot |E|)^2$ potential candidates in the *join*() step, the running time of the "for" loop is of $O(m^2 \cdot |E|^2 + |E| \cdot n^3 \cdot |F|)$. Therefore, the time complexity of the heuristic algorithm is $O(m^2 \cdot |E|^2 + |E| \cdot N^4 \cdot |F|)$. The choice of m, which directly affects running time and the anomalousness of the top-K results, will be studied in the experiment sections.

3.3.3 A More Sophisticated Heuristic Algorithm: ERAC_H

Since the bound given in Section 3.3.1 could be rather loose and gives little pruning power, and the assumption taken in the naive heuristic may be too strong and miss many anomalous ERACs, we now show a more sophisticated pruning technique that exploits our Apriori-style candidate generation.

Suppose we are at the stage of generating candidates of size n, and we know these candidates have anomaly scores that are no smaller than x. Now if we are examining two size-$(n-1)$ collections S_1 and S_2, and conclude from our computation that the upper-bound on the anomaly score of the resultant collection from combining S_1 and S_2 is still less than x, then it is unnecessary to combine them.

As stated in Section 3.2, the p-value of a given collection S is determined by the extremity index r ($r < |E|/2$) and the number of entities in S that appear in $E_f(r)$, denoted as i. For any given p-value and size n, we can express the underlying i and r according to the p-value formula by denoting the p-value as $p_f(i,r,n)$, or just $p(i,r,n)$ when the context is clear.

Let S_x be the ERAC realizing x (i.e., $\Omega(S_x,F) = x$). Since the anomaly score is the sum of the negative logarithm of the representative p-values for every feature, we want to derive for each feature $f \in F$, the corresponding representative p-value $p_f(i^x,r^x,n)$ for S_x. From there, we check whether, for feature f, combining S_1 and S_2 will achieve a representative p-value that is even smaller than $p_f(i^x,r^x,n)$. If so, combining S_1 and S_2 will achieve a higher anomaly score than x. We will discuss how to decompose x into a set of p-values later in this section. For now, we assume $p_f(i^x,r^x,n)$ is known for any f.

Our task is to compute, for any given feature f, the bound on the representative

p-value of the resultant collection from combining S_1 and S_2, and compare against $p_f(i^x, r^x, n)$.

We first examine the following p-value Property 3.

Property 3. *Given any feature f, collections S and S' and extremity indexes r and r', if $|S_f(r)| = |S'_f(r')|$, $|E_f(r)| = |E_f(r')|$ and $|S| > |S'|$, then $p_f(S,r) > p_f(S',r')$.*

Since as far as the hypergeometric distribution is concerned, the role of $|S_f(r)|$ and $|E_f(r)|$ are interchangeable [Har65], Property 3 can be proven similarly to Property 2. Property 3 suggests that with all the other parameters kept constant, the smaller the collection size, the smaller the p-value is.

We now show the following Lemma stating that the combination of S_1 and S_2 is warranted, i.e., the resultant collection's p-value is smaller than $p_f(i^x, r^x, n)$, only when each of them has a p-value that is at most $p(i^x - 1, r^x, n-1)$.

Lemma 1. *For any feature f, given any representative p-value $p(i,r,n)$ of some collection S of size n, the representative p-values of S's subsets of size $n-1$ that generate S is at most $p(i-1,r,n-1)$.*

Proof. Let S_1 and S_2 denote the generating subsets of S. We have $|S_1| = |S_2| = n-1$, $|S_1 \cap S_2| = |S_1| - 1$, $|S_1 - S_2| = 1$ and $S_1 \cup S_2 = S$. According to the definition of $p(i,r,n)$, S has i entities in $E(r)$. Hence S_1 and S_2 must have either i or $i-1$ entities in $E(r)$. Therefore, S_1 and S_2 must have p-value of either $p(i-1,r,n-1)$ or $p(i,r,n-1)$. According to Property 3, we know that $p(i-1,r,n-1) \geq p(i,r,n-1)$. Hence, $p(i-1,r,n-1)$ is the largest possible p-value among S_1 and S_2, since the representative p-value is smaller than or equal to $p(i-1,r,n-1)$ by definition. We therefore prove that the upper-bound on the representative p-values of S's subsets of size $n-1$ that generate S (i.e. S_1 and S_2) is $p(i-1,r,n-1)$. □

An illustration of Lemma 1 is as follows. Suppose $|E| = 30$, there exists a collection S of size $n = 4$ whose representative p-value indicates that it has 3 members (i.e., $i = 3$) ranked among the top 8 (i.e., $r = 8$). Thus we have $p(3,8,4) = 0.048$. A collection (e.g., S_1) that can generate S must have at least 2 entities ranked among

the top 8. This means $p(i-1,r,n-1) = p(2,8,3)$ =0.166 must be one of the p-values of S_1. By definition, S_1's representative p-value is therefore no greater than $p(2,8,3)$, which we conclude to be the upper-bound.

Investigating deeper into the relationship between $p(i,r,n)$ and $p(i-1,r,n-1)$, we arrive at the following:

Property 4. $p(i_1-1,r_1,n-1) - p(i_1,r_1,n) = \frac{n-i_1+1}{n} \cdot prob(i_1-1,|E|,r_1,n)$.

Proof. According to the definition of p-value, $p(i,r,n) = p(i-1,r,n) - prob(i-1,|E|,r,n)$. According to Property 3, $p(i-1,r,n) = p(i-1,r,n-1) + \frac{\binom{n-1}{i-1-1}\cdot\binom{|E|-n}{r-(i-1)}}{\binom{|E|}{r}}$. Thus, $p(i,r,n) = p(i-1,r,n-1) + \frac{\binom{n-1}{i-1-1}\cdot\binom{|E|-n}{r-(i-1)}}{\binom{|E|}{r}} - prob(i-1,|E|,r,n)$. We have

$p(i_1-1,r_1,n-1) - p(i_1,r_1,n) = -\frac{\binom{n-1}{i_1-1-1}\cdot\binom{|E|-n}{r_1-(i_1-1)}}{\binom{|E|}{r_1}} + \frac{\binom{n}{i_1-1}\cdot\binom{|E|-n}{r_1-(i_1-1)}}{\binom{|E|}{r_1}}$

$= -\frac{\binom{n-1}{i_1-1-1}\cdot\binom{|E|-n}{r_1-(i_1-1)}}{\binom{|E|}{r_1}} + \frac{(\binom{n-1}{i_1-1}+\binom{n-1}{i_1-1-1})\cdot\binom{|E|-n}{r_1-(i_1-1)}}{\binom{|E|}{r_1}} = \frac{\binom{n-1}{i_1-1}\cdot\binom{|E|-n}{r_1-(i_1-1)}}{\binom{|E|}{r_1}}$

$= \frac{n-i_1+1}{n} \cdot prob(i_1-1,|E|,r_1,n)$. □

It is important to note that the bound we derived is for the particular size-n collection S generated from combining its size-$n-1$ subsets S_1 and S_2. It does not generalize to the case where S_1 and S_2 are arbitrary collections of size $n-1$. For example, $p(1,2,3) = 0.284 > p(3,10,3) = 0.1053$, but the collections of size 4 generated from $p(1,2,3)$ is of $p(2,2,4) = 0.0315$, which is smaller than $p(4,10,4) = 0.043$. This suggests although $p(i-1,r,n-1)$ is the upper-bound on collections of size $n-1$ for generating the particular collection of $p(i,r,n)$, it may not be the upper-bound for generating any collections that have a p-value smaller than $p(i,r,n)$. We need to carefully examine all possible relationships between different r and i.

For any i_1, i_2, r_1 and r_2, we have 9 possible cases chosen from $\{i_1 < i_2, i_1 = i_2, i_1 > i_2\} \times \{r_1 < r_2, r_1 = r_2, r_1 > r_2\}$. Given two representative p-values $p(i_1-1,r_1,n-1)$ and $p(i_2-1,r_2,n-1)$ of collections of size $n-1$, and the condition $p(i_1-1,r_1,n-1) > p(i_2-1,r_2,n-1)$, there are 4 cases that do not satisfy the condition. $\{i_1 = i_2\} \times \{r_1 = r_2, r_1 < r_2\}$ violates the condition according to Property 2. $\{i_1 > i_2\} \times \{r_1 = r_2, r_1 < r_2\}$ is not possible according to Property 1 and Property 2. The rest of the cases are addressed by the following two lemmas:

Lemma 2. *Given two representative p-values $p(i_1-1,r_1,n-1)$ and $p(i_2-1,r_2,n-1)$ of collections of size $n-1$, and $p(i_1-1,r_1,n-1) > p(i_2-1,r_2,n-1)$, if ($i_1 = i_2$ and $r_1 > r_2$) or ($i_1 < i_2$ and $r_1 = r_2$) or ($i_1 < i_2$ and $r_1 > r_2$), then $p(i_1,r_1,n) > p(i_2,r_2,n)$.*

Proof. For $i_1 = i_2$ and $r_1 > r_2$, we have $p(i_1,r_1,n) > p(i_2,r_2,n)$, according to Property 2. For $i_1 < i_2$ and $r_1 = r_2$, we have $p(i_1,r_1,n) > p(i_2,r_2,n)$, according to Property 1. For $i_1 < i_2$ and $r_1 > r_2$, we have $p(i_1,r_1,n) > p(i_2,r_2,n)$, according to Property 1 and Property 2. □

Lemma 3. *Given two representative p-values $p(i_1-1,r_1,n-1)$ and $p(i_2-1,r_2,n-1)$ of collections of size $n-1$, and $p(i_1-1,r_1,n-1) > p(i_2-1,r_2,n-1)$, if ($i_1 < i_2$ and $r_1 < r_2$) or ($i_1 > i_2$ and $r_1 > r_2$) and $(n-i_1+1)\cdot prob(i_1-1,|E|,r_1,n) < (n-i_2+1)\cdot prob(i_2-1,|E|,r_2,n)$, then $p(i_1,r_1,n) > p(i_2,r_2,n)$.*

Proof. According to Property 4, $p(i_1,r_1,n) - p(i_1-1,r_1,n-1) = -\frac{n-i_1+1}{n}\cdot prob(i_1-1,|E|,r_1,n)$. Similarly, $p(i_2,r_2,n) - p(i_2-1,r_2,n-1) = -\frac{n-i_2+1}{n}\cdot prob(i_2-1,|E|,r_2,n)$. Since $(n-i_1+1)\cdot prob(i_1-1,|E|,r_1,n) < (n-i_2+1)\cdot prob(i_2-1,|E|,r_2,n)$, we have $p(i_1,r_1,n) - p(i_1-1,r_1,n-1) > p(i_2,r_2,n) - p(i_2-1,r_2,n-1)$. With $p(i_1-1,r_1,n-1) > p(i_2-1,r_2,n-1)$, we therefore have $p(i_1,r_1,n) > p(i_2,r_2,n)$. □

In Lemma 3, if we impose on i_2 such that $\forall r', \nexists i' < i_2$ with $p(i',r',n) < p(i_2,r_2,n)$, then for the case of $i_1 < i_2$ and $r_1 < r_2$, we always have $p(i_1,r_1,n) > p(i_2,r_2,n)$ by definition.

Therefore, according to Lemma 2 and Lemma 3, for a given representative p-value $x = p(i^x,r^x,n)$, such that $\nexists i' < i^x$ with $p(i',r',n) < p(i^x,r^x,n)$ for any r', we can compute the lower-bound on the p-value of collections of size $n-1$ by $p(i^x-1,r^x,n-1)$. Except for the case of $i_1 > i^x$, $r_1 > r^x$ and $p(i^x-1,r^x,n-1) < p(i_1-1,r_1,n-1)$, we need to further check whether $p(i^x,r^x,n) < p(i_1-1,r_1,n-1)$ still holds. If it no longer holds, we need to increase our upper-bound from $p(i^x-1,r^x,n-1)$ to $p(i_1-1,r_1,n-1)$. This guarantees the true upper-bound.

However, this additional checking is costly. We therefore heuristically take $p(i^x-1,r^x,n-1)$ as the upper-bound on p-value of collections of size $n-1$. With

$p(i^x-1, r^x, n-1)$, we take $p(i^x-2, r^x, n-2)$ as the upper-bound on p-value of collections of size $n-2$, and so on.

Thus, given an anomaly score x of any size-n collection and a collection size $n', n' < n$, we define the **lower-cut with score-size constraint** (x, n) as

$\Omega^*(x, n, n') = -\sum_{f \in F} \log p_f(i^x - (n-n'), r^x, n')$.

After computing these lower-cuts for collections of size n' along each feature, we sum the negative log values of the bounds to get a lower-cut on the anomaly score of the collections; any collections of size n' with anomaly score below this lower-cut can be pruned as shown in the following pruning strategy.

Pruning Technique 2. *Given an anomaly score threshold x for collections of size n, and its correspondent representative p-value $p_f(i^x, r^x, n)$ for each feature $f \in F$, we use $p_f(i^x - (n-n'), r^x, n')$ to derive the lower-cut $\Omega^*(x, n, n')$ and prune away any collection of size $n', n' < n$, such that its anomaly score is smaller than $\Omega^*(x, n, n')$.*

Overall our heuristic algorithm works as follows. Given a size N, we first estimate a threshold Ω_N for collections of size N by the anomaly score of the collection comprising the top-N anomalous singular entities, which is a valid candidate collection to start with. It is possible that this estimated initial bound Ω_N is too aggressive and is even higher than the true Ω_t of the final top-K result. As remedy, we first run with this initial Ω_N to obtain a preliminary top-K result, then set the smaller one between the initial Ω_N and this preliminary Ω_t as the new threshold Ω_N to reboot the algorithm.

With this new threshold Ω_N and its correspondent set of $p_f(i^x, r^x, n)$, we compute the sequence of lower-cuts $\Omega^*(\Omega_N, N, i)$ for all levels $1 \leq i < N$. Another trick is that in generating candidates of size n, it could be that the anomaly score of the least one in the current top-K list (i.e., Ω_t) can provide a better cut, i.e., $\Omega^*(\Omega_t, n+1, n) > \Omega^*(\Omega_N, N, n)$. Using the better cut, we can prune away current ERACs of size n before trying to combine any two of them to generate candidates of size $n+1$. The details are shown in Algorithm 2.

Running Example. Setting K = 1 and N= 3 again, we show how Algorithm 2 executes on the example in Figure 3.2. The algorithm estimates Ω_N as $\Omega(\{e_{16}, e_5,$

Algorithm 2 ERAC_H for independent features

Input: E, F, K, N

Output: Top-K ERACs: $\mathbb{S}^*$

$\{\mathbb{S}^*$ is implemented by a priority queue of max length K. Ω_t is the smallest anomaly score of collections in $\mathbb{S}^*$, and is set to zero if $\mathbb{S}^* = \emptyset$ $\}$

1: $n = 1$; $\mathbb{S}^* = \emptyset$; $\mathbb{S} = \{\{e\} \mid e \in E\}$
2: $\Omega_N = \Omega(S_N, F)$, where S_N is the union of the top-N anomalous elements in $\mathbb{S}$.
3: **repeat**
4: $\quad \Omega'_N = \Omega_N$
5: $\quad$ **for** $n = 1$ to $N-1$ **do**
6: $\quad\quad \mathbb{S} = \{S \in \mathbb{S} \mid \Omega(S,F) > \max(\Omega^*(\Omega_N, N, n), \Omega^*(\Omega_t, n+1, n))\}$
7: $\quad\quad \mathbb{S} = join(\mathbb{S}, \mathbb{S})$ $\{\mathbb{S}^*$ and Ω_t are updated by elements in $\mathbb{S}$ whenever necessary$\}$
8: $\quad \Omega_N = \Omega_t$
9: **until** $\Omega_N \geq \Omega'_N$
10: **return** $\mathbb{S}^*$

$e_{24}\}, \{f_0\}) = 8.31$, since these three entities are the top-3 anomalous singular entities. When $n = 1$, the algorithm checks whether each collection of size 1 can beat the current Ω_t by Pruning Technique 2. As $\Omega^*(8.31, 3, 1) = -\log p(1,3,1)$, only $\{e_{16}\}$,$\{e_5\}$ and $\{e_{24}\}$ are selected to generate collections of size 2. After the join step, $\mathbb{S} = \{\{e_{16}, e_5\}, \{e_{16}, e_{24}\}, \{e_5, e_{24}\}\}$. The algorithm goes on to find $\{e_{16}, e_5\}$ as the current top-1 ERAC. When $n = 2$, Step 7 finds all elements in the current $\mathbb{S}$ are eligible to generate collections of size 3. The join() generates the collection $\{e_{16}, e_5, e_{24}\}$ and keeps it as the top-1 ERAC. Since the current Ω_t is equal to the estimated threshold $\Omega(\{e_{16}, e_5, e_{24}\}, \{f_0\})$, the algorithm stops and returns $\{e_{16}, e_5, e_{24}\}$ as the final result.

For the same example, we have shown that the exact algorithm ERAC_E visits 18 nodes in total, whereas the heuristic algorithm ERAC_H only visits 3+3+1=7 nodes, saving the visit to 61% of nodes over the exact algorithm.

Time Complexity. We now analyze the time complexity of Algorithm 2. Let $|\overline{\mathbb{S}'}|$ denote the average size of $\mathbb{S}$ for all n. Since Step 6 is of $O(|\overline{\mathbb{S}'}| \cdot n^2 \cdot |F|)$ and Step 7 is of $O(|\overline{\mathbb{S}'}|^2)$, the time complexity of Algorithm 2 is $O(|\overline{\mathbb{S}'}|^2 + |\overline{\mathbb{S}'}| \cdot N^3 \cdot |F|)$, lower than that of Algorithm 1 as $N^3 < N^4$ and as we expect $|\overline{\mathbb{S}'}|$ in Algorithm 2 to be much smaller than the counterpart $|\overline{\mathbb{S}}|$ in Algorithm 1.

Anomaly Score Decomposition. We now discuss how to derive representative

value $p_f(i^x, r^x, n)$ for each feature $f \in F$ from a given anomaly score x. Ideally, we aim to obtain the set of p-values that minimize $\Omega^*(x,n,n')$, subject to (I) $x \leq -\sum_{f\in F} \log x_f$; (II) $\nexists i' < i^x$ with $p(i',r',n) < p(i^x,r^x,n)$.

For efficiency purpose, we approximate the minimum value of $\Omega^*(x,n,n')$ by assuming the anomaly score x is evenly divided across features. We first compute $e^{-\frac{x}{|F|}}$, then find $p_f(i^x,r^x,n)$ such that (I) $p_f(i^x,r^x,\ n) \leq e^{-\frac{x}{|F|}}$; (II) $\nexists i' < i^x$ with $p(i',r',n) < p(i^x,r^x,n)$; (III) $\nexists r' < r^x$ with $p(i^x,r',n) < e^{-\frac{x}{|F|}}$. Condition (I) guarantees that $x \leq -\sum_{f\in F} \log x_f$. Condition (II) is required in the heuristic. Condition (III) is based on Property 2. As i^x is fixed, we choose the larger r^x so that the corresponding $p(i^x-(n-n'),r^x,n')$ is larger, which in turn leads to a smaller anomaly score.

3.3.4 Handling Dependent Features

For the simplicity of discussion, we have so far assumed that the features in F are independent of one other. Now we handle the more general case of dependant features. A feature set F is said to be an Independent Feature Set or IFS, if $|F| > 1$ and $\forall f_i, f_j \in F$, f_i is independent of f_j. The dependency of any two features is defined by the Kendall Tau rank correlation coefficient [Ken48]. As there are potentially many IFS, we focus on the maximal IFSs.

An IFS F is a **maximal IFS** if $\nexists$ an IFS F' s.t. $F \subset F'$. Given a feature set F, the set of all maximal independent feature sets derived from F is denoted as $\mathbb{F}$.

In Section 3.2, the anomaly score is measured on one independent feature set. Given $\mathbb{F}$, the overall anomaly score of S is aggregated by taking the largest anomaly score across all maximal IFSs in $\mathbb{F}$:

$$\Omega(S) = \max_{F'\in\mathbb{F}} \Omega(S,F')$$

We now derive two algorithms shown in Algorithm 3 to find ERACs involving a dependent feature set F based on the exact and heuristic algorithms for independent features. We denote the algorithm using ERAC_E as ERACD_E and the one using ERAC_H as ERACD_H. In both ERACD_E and ERACD_H, we first generate all the maximal IFS from F. For each maximal IFS, we call ERAC_E or ERAC_H. Finally,

we get the top-K collections of size up to N by aggregating the top-K collections across all maximal IFSs. Since we are presented with multiple maximal IFSs, we return not only the ERACs but also their associated maximal IFSs. We still use a priority queue of maximum length of K as before, but with composite elements $\langle S, F' \rangle$, where S is an ERAC and F' is the maximal IFSs that make S most anomalous. We constrain the priority queue to have distinct collections only. Also note that it is NP-hard to compute all maximal IFSs $\mathbb{F}$ [Rob86]. We adopt the algorithm in [Epp05] for an approximation for computing the maximal IFS.

Algorithm 3 ERACD_E and ERACD_H

Input: E, F, K and N
Output: top-K ERACs and maximal IFSs: $\mathbb{C} = \{\langle S, F' \rangle\}$
{$\mathbb{C}.\mathbb{S}^*$ denotes the top-K ERACs. $\mathbb{C}.F(S)$ returns the corresponding maximal IFS of S in $\mathbb{C}$}

```
1: ℂ = ∅
2: 𝔽 = generateMIFS(F)
3: for all F' ∈ 𝔽 do
4:     𝕊 = ERAC_E(F', N, K) {for ERACD_H, 𝕊 = ERAC_H(F', K, N) }
5:     for all S ∈ 𝕊 do
6:         if S ∈ ℂ.𝕊* then
7:             if Ω(S, ℂ.F(S)) < Ω(S, F') then
8:                 delete ⟨S, ℂ.F(S)⟩ from ℂ
9:                 add ⟨S, F'⟩ to ℂ
10:        else
11:            add ⟨S, F'⟩ to ℂ
12: return ℂ
```

3.4 ERAC Expansion

3.4.1 ERAC Expansion Problem

With the aforementioned algorithms, users are able to find top-K ERACs of size no greater than a predefined N. It is natural to wonder if the identified ERACs can be expanded to larger, more anomalous ERACs. For example, after seeing some identified top-K spammer collections, users may be interested in supersets of these spammer collections, which are even more anomalous.

One way to derive the larger ERACs is to rerun the ERAC detection algorithms

with a larger N. However, users often have little idea about the exact size of the larger ERACs to set, hence a trial and error strategy is non-ideal. Moreover, it is possible that some interesting ERAC in the top-K list for N may disappear in the top-K list of a larger N with the current K setting. This makes it even harder to track down the superset of the ERAC of interest.

In this section, we propose the problem of expanding an ERAC to the most anomalous superset of any size. We then propose algorithms to solve the problem.

Definition 3. ***[ERAC Expansion Problem]*** *Given an ERAC S, find $S' \subset E - S$ such that (I) $S \cup S'$ is an ERAC; (II) $\Omega(S \cup S', F) > \Omega(S, F)$; (III) $\forall S'' \subset E - S$, we have $\Omega(S \cup S', F) \geq \Omega(S \cup S'', F)$*

The definition states that expanding an ERAC S will produce the superset of S having the largest anomaly score among all of S's supersets.

3.4.2 ERAC Expansion Algorithms

With little modification, the algorithms in Section 3.3 for detecting top-K ERACs of size no greater than N can be applied to expand the ERACs. Specifically, given an ERAC S to be expanded, for the exact algorithm ERACD_E, we first set N=$|E|/2$ and K=1. Then in the process of computing the upper-bound of the supersets of any candidate collection S', we compute the upper-bound of $S \cup S'$ instead. According to the analysis in Section 3.3.1, the complexity is $O(|\overline{\mathbb{S}}|^2 + |\overline{\mathbb{S}}| \cdot E^4 \cdot |F|)$.

For the sophisticated heuristic algorithm ERACD_H, we also set N=$|E|/2$ and K=1. When computing the lower-cut using S_x, we use $S \cup S_x$ instead. According to the analysis in Section 3.3.3, the complexity is of $O(|\overline{\mathbb{S}'}|^2 + |\overline{\mathbb{S}'}| \cdot E^3 \cdot |F|)$.

As the modified ERACD_E and ERACD_H are still costly, we propose below more efficient algorithms for ERAC expansion.

A Greedy Algorithm.

We design a greedy algorithm to add one entity at a time until no further expansion can produce a more anomalous ERAC. In each step, we add one entity such that the

resultant ERAC gives the highest anomaly score.

This greedy scheme is based on the observation that members of an ERAC behave like each other and most likely occupy the extreme positions of a similar set of features. Thus, given a subset of an ERAC, the rest of the members of this ERAC are expected to have similar extreme behavior as the subset.

A naive way of choosing an entity to join S is to try out every entity in E and select the one that gives the largest anomaly score after joining with S. This is costly, as each insertion of an entity to S entails scanning through all the entities in E. A more efficient approach only needs to try out the entities within the extremity index at each insertion step, which can be easily derived by the existing p-values of S on each feature.

Specifically, for any feature f, we know the corresponding $p(i,r,n)$ for S on f. To insert into S an entity that gives a larger anomaly score, we want $p(i+1,r',n+1)$ to be smaller than $p(i,r,n)$. Since there can be many r' that satisfy the condition, we define the upper-bound on r', $r(S,f)$, to be the largest extremity index such that adding any entity within $r(S,f)$ produces a smaller p-value than $p(i,r,n)$ on feature f. Note that these p-values can be pre-computed and sorted for efficient retrieval. Thus finding the right $r(S,f)$ can be done in constant time.

Given S and F, we define the **candidate pool** $A(S,F)$ as the set of entities within $r(S,f)$ for all $f \in F$. That is, $A(S,F) = \bigcup_{f\in F} E_f(r(S,f))$, where $E_f(r)$ is the set of entities within $r(S,f)$ defined in Section 3.2.

Lemma 4. *$\forall e \in E - A(S,F)$, we have $\Omega(S\cup\{e\},F) < \Omega(S,F)$.*

This lemma states that we only need to check the entities in the candidate pool instead of all entities in E to guarantee we find the entity e that gives the largest anomaly score after joining S.

The lemma can be easily proven. Since $r(S,f)$ is the upper-bound on the extremity index for feature f, each entity in $A(S,F)$ when joining S will give a smaller p-value in at least one feature than that of S alone. Thus, every entity in $A(S,F)$ has a chance to produce a more anomalous superset after joining S. On the other hand, every entity in $E - A(S,F)$ has no chance to produce a more anomalous superset,

as any resultant superset is less anomalous than S w.r.t every feature in F. Since the final anomaly score is the sum of the anomaly scores w.r.t. all significant features, each entity in $E - A(S,F)$ when joining with S will give a smaller anomaly score than that of S.

With this lemma, we propose the following greedy algorithm for expanding S.

Algorithm 4 ERAC Expansion ERAC_exp

Input: E, F, an ERAC S^0 to be expanded
Output: Expanded ERAC S from S^0

1: $S = S^0$
2: **for** $i = |S^0|to|E|/2$ **do**
3: $A = \emptyset$ {A keeps the candidate pool $A(S,F)$}
4: **for all** $f \in F$ **do**
5: find $r(S,f)$ in the pre-computed and sorted p-value list
6: $A = A \cup E_f(r(S,f))$
7: Find entity e in A s.t. $\Omega(S \cup \{e\},F)$ is the largest.
8: **if** $\Omega(S \cup \{e\},F) > \Omega(S,F)$ **then**
9: $S = S \cup \{e\}$
10: **else**
11: break
12: **return** S

The algorithm is given an initial collection S^0 to expand and outputs the resultant S, which is the most anomalous superset of S^0 of any size. In each loop of i in Algorithm 4, a new A (i.e., $A(S,F)$) is computed from Step 4 to Step 6 and entities in A are visited to find the one that boosts the anomaly score the most. If the resultant anomaly score still increases, the size of S is incremented by one.

We now analyze the complexity of Algorithm 4. In Step 7, let us suppose $|\bigcup_{f \in F} E_f(r(S,f))| = \bar{r} \cdot |F|$, where $\bar{r}$ is the average number of entities within the extremity index across all features. Given that computing $\Omega(S \cup \{e\},F)$ is of $O((|S|+1)^2 \cdot |F|)$, Step 7 takes $O(\bar{r} \cdot |F| \cdot (|S|+1)^2 \cdot |F|)$.

Therefore, the complexity of the whole algorithm is $O(|E| \cdot |F| + \bar{r} \cdot |F| \cdot |E|^3 \cdot |F| + |E|^3 \cdot |F|)$. Recall the complexity of the modified sophisticated heuristic algorithm for detecting ERACs is of $O(|\overline{\mathbb{S}'}|^2 + |\overline{\mathbb{S}'}| \cdot E^3 \cdot |F|)$. Since $|\overline{\mathbb{S}'}|$ is $O(|E|^N)$, the expansion algorithm here is much faster.

A Heuristic-based Greedy Algorithm.

Note that the time consuming part of the expansion algorithm is in each loop of i computing the anomaly score for all entities in the updated candidate pool $A(S,F)$ in Step 7, which has complexity $O(\bar{r} \cdot |F| \cdot |E|^2 \cdot |F|)$. If we can expedite this step, we can achieve an even more efficient algorithm. This is possible if we avoid computing the value of $\Omega(S \cup \{e\}, F)$ for each entity in $A(S,F)$. Instead we choose from $A(S,F)$ the entity that gives the largest anomaly score when joining S^0, which is pre-computed for each entity e in E.

We introduce the following **expansion heuristic**: entity e in $A(S,F)$ that gives the largest $\Omega(S^0 \cup \{e\}, F)$ value will likely give the largest $\Omega(S \cup \{e\}, F)$.

Since our algorithm greedily adds entities to S^0 one after another, this heuristic ignores all entities added to S^0 in the expansion process and always selects the entity in $A(S,F)$ that contributes the most to S^0 instead of S. Suppose S^0 is in fact a subset of a larger ERAC S, the heuristic has a better chance of retrieving S from S^0, if S^0 already has a similar extreme pattern as S, i.e., both of them occupy extreme positions on similar features.

However, S^0 may not always have a similar extreme pattern as S, i.e., S^0 may occupy extreme positions of a few more or less features than S does, which is more likely to happen when the size of S^0 is much smaller than S. When this happens, the expansion heuristic may fail as it favors only the entities that are similar to all members in S^0, which are not necessarily similar to all members in S.

For example, Figure 3.3 shows the top-12 positions of four features with seven entities e_1 to e_7. The empty positions are occupied by other entities, which are not relevant and are not shown for simplicity. Suppose the target ERAC is $S = \{e_1, e_2, e_3, e_4, e_5, e_6\}$. If we expand $S^0 = \{e_1, e_2\}$ with the expansion heuristic, the first entity joining S^0 is e_7, which produces the highest $\Omega(S^0 \cup \{e\}, F)$ value compare to other entities. This will lead us to a different superset than S. However, if we start with $S^0 = \{e_1, e_2, e_3\}$, the following entities to join S^0 are then from among $\{e_4, e_5, e_6, e_7\}$, as they all produce high $\Omega(S^0 \cup \{e\}, F)$ values. This way, the heuristic successfully leads us to the more anomalous superset S.

To incorporate this heuristic, we change Step 7 in Algorithm 4 to "find the entity

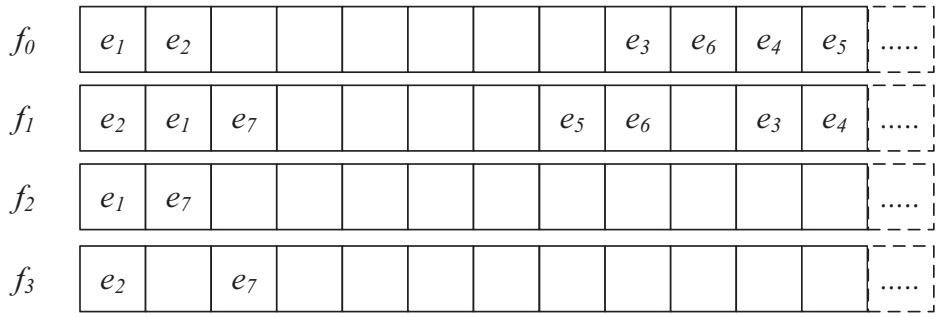

Figure 3.3: An illustration of the expansion heuristic. $S = \{e_1, e_2, e_3, e_4, e_5, e_6\}$

e in A such that $\Omega(S^0 \cup \{e\}, F)$ is the largest". We also assume that the corresponding anomaly score $\Omega(S^0 \cup \{e\}, F)$ for each entity e in E is pre-computed. As a result, the new Step 7 has complexity $O(\bar{r} \cdot |F|)$.

Therefore, the complexity of the whole heuristic-based greedy expansion algorithm is $O(|E| \cdot |F| + |E| \cdot \bar{r} \cdot |F| + |E|^3 \cdot |F|)$, faster than without the heuristic. We apply the more efficient heuristic based greedy algorithm, denoted ERAC_exp, in the experiments to demonstrate the effectiveness of expanding ERACs.

3.5 Discussion on Exploratory ERAC Search

In this section, we discuss an exploratory search strategy for more anomalous ERACs, making use of the algorithms developed for both ERAC detection and expansion.

Ideally, users want to search for more anomalous ERACs regardless of collection size. Since the search space grows exponentially with the population size, for efficiency reasons, we propose to first investigate the ERAC detection problem of finding top-K ERACs of size less than a predefined N of small values. The detected ERACs are treated as "seeds" in the follow-up ERAC expansion problem to retrieve the larger and more anomalous ERACs. In this strategy, the ability of discovering more anomalous ERACs largely depends on whether the seeds contains any subset of the more anomalous ERACs.

A direct way of incorporating seeds that lead to more anomalous ERACs is to set a large K and N. However, larger K and N will increase the execution time of the

ERAC detection algorithms. Unfortunately, setting the right K and N is not trivial. Due to the lack of downward closure property, it is possible that the subsets of a less anomalous ERAC are even more anomalous than the subsets of a more anomalous ERAC. Thus, one cannot easily derive the proper K and N by any monotonic property.

In this section, we therefore propose an exploratory search strategy that gradually increases K and N based on a gain function. Since the gain function is derived based on the proposed ERAC detection and expansion algorithms, we first describe the combined ERAC algorithm ERAC_comb in Algorithm 5. $\mathbb{C}$ denotes the resultant set of ERACs after expanding the top-K ERACs of size no greater than N.

Algorithm 5 Combined ERAC algorithm ERAC_comb

Input: E, F, K and N
Output: $\mathbb{C}$

1: $\mathbb{C} = \emptyset$
2: $\mathbb{C}'$ = $ERACD\_H(K,N)$ {run the ERAC detection algorithm}
3: **for all** $S \in \mathbb{C}'$ **do**
4: S=ERAC_exp(E,F,S) {run the ERAC expansion algorithm}
5: add S to $\mathbb{C}$
6: **return** $\mathbb{C}$

We use $T(K,N)$ to denote the time required to run ERAC_comb for a given K and N. The goal function, denoted by $G(K, N)$, can be defined according to users' search goal. For example, if users want to get the most anomalous ERACs, the goal function is $\max_{S\in\mathbb{C}}(\Omega(S,F))$. If users want as many anomalous ERACs as possible, the goal function is $\sum_{S\in\mathbb{C}}(\Omega(S,F))$.

Given (K,N) and (K',N') with $K' \geq K$ and $N' \geq N$, corresponding to two runs of the ERAC_comb algorithm, the **gain function** $I(K',N',K,N)$ is defined as:

$$\frac{G(K',N') - G(K,N)}{G(K,N)} - \frac{T(K',N') - T(K,N)}{T(K,N)}$$

The first part of the gain function captures the change ratio in gain function and the other capture the change ratio in execution time. This gain function has larger value if the second run of ERAC_comb with (K',N') have larger gain $G(K',N')$ and smaller execution time $T(K', N')$ than the first run with (K,N). We use the ratio to normalize the changes so that the change on gain and time are on the same scale.

Now, we describe our high-level search strategy in Algorithm 6. The increase of K or N is decided by the gain function.

Algorithm 6 Exploratory ERAC search

Input: E, F, initial K^0 and N^0 selected by user for K and N
Output: $\mathbb{C}$, the resultant set of ERACs

Set $K = K^0$ and $N = N^0$
Run ERAC_comb(E,F,K,N); set $t = T(K,N)$ and $g = G(K,N)$
while exploratory search continues **do**
    Run $\mathbb{C}_1$ =ERAC_comb$(E,F,K+1,N)$; set $t_1 = T(K+1,N)$ and $g_1 = G(K+1,N)$
    Run $\mathbb{C}_2$ =ERAC_comb$(E,F,K,N+1)$; set $t_2 = T(K,N+1)$ and $g_2 = G(K,N+1)$
    if $\frac{g_1-g}{g} - \frac{t_1-t}{t} > \frac{g_2-g}{g} - \frac{t_2-t}{t}$ **then**
        $\mathbb{C} = \mathbb{C}_1$
        $K = K+1$; $t = t_1$; $g = g_1$;
    else
        $\mathbb{C} = \mathbb{C}_2$
        $N = N+1$; $t = t_2$; $g = g_2$;
return $\mathbb{C}$

3.6 Experiments on Synthetic Data

In this section, we examine the performance of our ERAC detection and expansion algorithms on synthetic datasets. We are interested to know (I)how well can our algorithms retrieve the injected ERACs in synthetic data of various population sizes; (II)can our algorithms retrieve more anomalous collections compared to existing approaches.

Synthetic Data Generation.

The input to our synthetic data generation algorithm includes population size and number of features. In addition, we assume the size of any injected ERAC and the total number of injected ERACs are given. The output is the set of entity lists $\{EL_f\}$ and the set of injected ERACs. We begin with randomized entity lists and inject ERACs of various sizes until the entity lists are no longer independent of each other. The dependency between entity lists is computed as in [Ken48]. Algorithm 7 shows the detail.

Algorithm 7 *Generate synthetic data with injected ERACs*

Input: E, F, size of injected ERACs n, number of injected ERACs n_s
Output: a set of entity lists $\{EL_f\}$ and a set of injected ERACs $\mathbb{S}$

1: Generate for each feature f, a random entity list over the same population E as $\{EL_f\}$.
2: Let $\mathbb{S} \leftarrow \emptyset$
3: **while** $\{EL_f\}$ are independent of each other and $|\mathbb{S}| < n_s$ **do**
4: Randomly select $F' \subseteq F$ to be the significant features for a new injected ERAC.
5: Estimate the upper bound z on the extremity index of this ERAC, such that this ERAC is more anomalous than the top ERAC found in the original randomized entity lists in step 1.
6: For each feature $f \in F'$, randomly generate extremity index r_f, such that $r_f < z$.
7: Randomly select a feature $\widehat{f} \in F'$, and randomly select a collection S whose representative p-value is $p(n, r_{\widehat{f}}, n)$ w.r.t. $\widehat{f}$ and S is disjoint with all previously injected ERACs in $\mathbb{S}$.
8: Add S to $\mathbb{S}$. $\{S$ is one of the injected ERACs$\}$
9: **for** each $f \in F'$ and $f \neq \widehat{f}$ **do**
10: Randomly select a collection S' whose representative p-value is $p(n, r_f, n)$ w.r.t. r_f and is disjoint with all previously injected ERACs in $\mathbb{S}$.
11: Switch the positions of the elements in S to those of elements in S' in EL_f.
12: **return** $\{EL_f\}$ and $\mathbb{S}$

3.6.1 ERAC Detection in Synthetic Data

We compare the exact and heuristic algorithms in terms of effectiveness and efficiency by varying the parameter settings and generating synthetic datasets with different ground truths.

When generating synthetic datasets, we fix the number of features at 10, the size of any injected ERAC at 10, and the number of injected ERACs at 5. Different synthetic datasets are generated by varying the population size $|E|$ from 100 to 300. For each population size, we generate three datasets for measuring the average performance of our exact ERAC_E, naive heuristic ERAC_N and sophisticated heuristic ERAC_H algorithms.

For all our ERAC detection algorithms, we fix N=10 and K=5 according to the synthetic data generation parameters. As for ERAC_N, we vary m that controls the number of candidate collections, i.e., $m \in \{0.1, 0.5, 1.0\}$. A larger m requires more

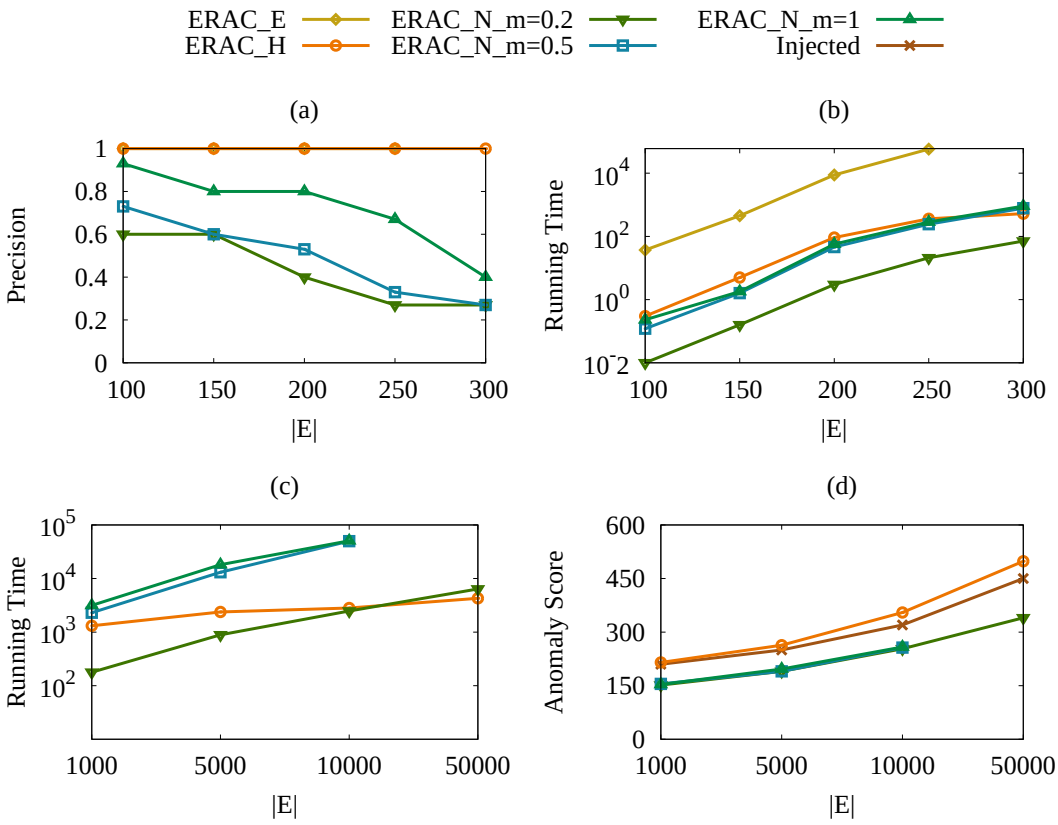

Figure 3.4: Results on synthetic data

candidate collections to be processed.

Figure 3.4(a) and Figure 3.4(b) report the precision and running time of each algorithm respectively, averaged over 3 generated datasets for each population size. When $|E| = 300$, the exact algorithm takes more than 24 hours, thus its results are excluded.

We see from Figure 3.4(a) that both ERAC_E and ERAC_H are able to retrieve all injected ERACs and achieve perfect precision. On the other hand, the precision of ERAC_N drops as m decreases and the population size $|E|$ increases. These precisions are in general lower than those of ERAC_E and ERAC_H. Figure 3.4(b) confirms that ERAC_H and ERAC_N require much lower execution time than the exact one as expected. ERAC_N needs more execution time when m decreases. The figures show that our sophisticated algorithm ERAC_H achieves comparable precisions with much less execution time than ERAC_E. It also achieves higher precisions although it incurs a longer running time than ERAC_N.

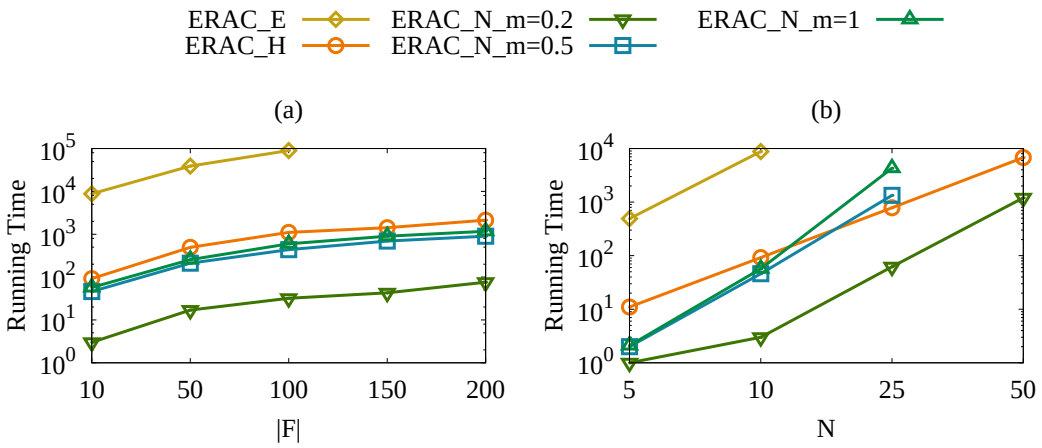

Figure 3.5: The impact of $|F|$ and N on running time of proposed algorithms

To test the scalability of ERAC_N and ERAC_H, while keeping $N = 10$, $K = 5$ and $|F| = 10$, we gradually increase the population size up to 500,000, where ERAC_N with $m = 0.5$ and $m = 1.0$ cannot complete within 24 hours. For each population size, we again generate three datasets. Besides running time, we also report the anomaly score of the K-th most anomalous ERAC identified by each algorithm and the anomaly score of the K-th most anomalous injected ERAC, denoted as injected. We plot the results in Figure 3.4(c) and Figure 3.4(d).

We see that ERAC_H scales well with data size. In all the settings, ERAC_H is able to retrieve either all the injected ERACs or the ones that are even more anomalous than the injected ERACs. Note that as the population size goes larger, if any randomly injected ERAC is not very anomalous, its subsets are more likely to form even more anomalous ERACs with other entities. However, ERAC_N cannot even retrieve collections that are as anomalous as the injected ones. This shows that the pruning technique used in ERAC_H takes good advantage of the properties of the ERACs and render much better results than the naive heuristics.

Next, we investigate the impact of the size of the feature set F and the collection size N on the execution time of ERAC_N (with $m \in \{0.1, 0.5, 1.0\}$ as before) and ERAC_H. According to the result of above experiments, we fix the population size $|E|$ to 200, so that ERAC_E can finish in a reasonable period. We first set $N = 10$, K=5 and vary $|F|$ from 10 to 200. For each choice of $|F|$, we generate three datasets as before. The running time is averaged across datasets and shown in Figure 3.5(a).

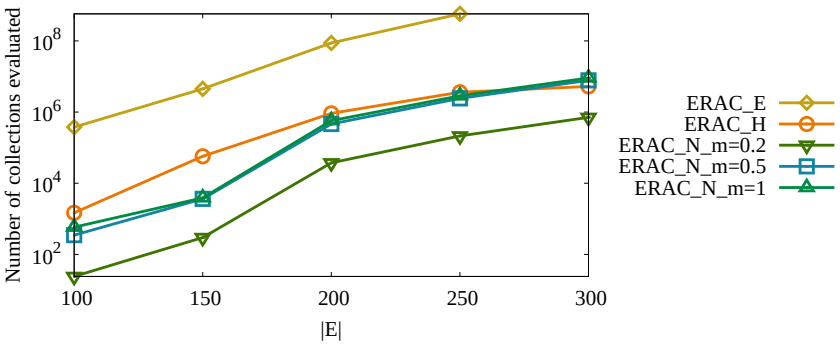

Figure 3.6: The pruning power of proposed algorithms

We can see that the number of features has less impact on the running time of all algorithm than population size $|E|$ does as shown in Figure 3.4. We then set $|F| = 10$, K=5 and vary N from 5 to 50. For each choice of N, we also generate three datasets and plot the averaged running time in Figure 3.5(b). We can observe from this figure that N has much larger impact on the running time than $|F|$ does, which is in line with the time complexity analysis. Note that in Figure 3.5, if any run takes more than 24 hours, its results are excluded.

We also study the pruning power of the proposed algorithms by measuring the number of collections evaluated. The fewer the number of collections being evaluated, the large pruning power an algorithm has. We fix N=10, K=5, $|F| = 10$ and vary the population size $|E|$ from 100 to 300 as before for the purpose of cross checking. We generate three datasets for each setting of $|E|$, the results are plotted in Figure 3.6, which shows our heuristic algorithms ERAC_H and ERAC_N has larger pruning power than the exact one as expected.

3.6.2 ERAC Expansion in Synthetic Data

Here we demonstrate the effectiveness of expanding ERACs on synthetic data using the algorithm proposed in Section 3.4.2. We are interested to know whether the expanding algorithm ERAC_exp can retrieve the injected ERACs from the top-K ERACs identified by the detection algorithm ERAC_H.

For the synthetic data generation, we set $|E|$=200, $|F|$=10 and the size of the injected ERAC to 10. For simplicity, we inject one ERAC into each synthetic data.

N	3	5	7	9
success rate	35/50	50/50	50/50	50/50

Table 3.3: The results of ERAC expansion in five synthetic datasets. Success rate reflects the ratio of successfully retrieving the injected ERAC by exp in five synthetic datasets

We run the synthetic generation algorithm five times so that we have 5 synthetic datasets, each with one injected ERAC.

Next we apply the ERACD_H with N=3, 5, 7, 9 and K=10 on each of the five synthetic datasets. It turns out that for each setting of N, all top-10 ERACs returned are subsets of the corresponding injected ERAC.

For each of the five synthetic datasets, we expand top-10 ERACs for each setting of N=3,5,7,9 and run our expanding algorithm ERAC_exp. Thus, we run ERAC_exp 200 (5*4*10=200) times altogether. To measure the performance of the algorithm, we measure the "success rate" of the algorithm in retrieving exactly the injected ERAC over the five synthetic datasets. Otherwise, we give a zero score.

We observe in Table 3.3 that when N gets larger, ERAC_exp successfully retrieves the injected ERAC for all the five synthetic datasets. When N=3, there are 15 cases out of 200 where the algorithm fails to expand some top-K ERACs to the original injected ERAC of size 10. This is because, when N is small, the detected ERAC of size N is likely to have extreme patterns dissimilar with the injected ERAC. Thus, in the greedy expansion process, the heuristic in ERAC_exp does not favor the other members in the injected ERACs and ends up with a different expanded ERAC than the injected one. When N is larger, it is more likely for the detected ERACs to have similar extreme pattern as the injected ERAC, leading to a very high success rate.

3.7 Experiments on Web Spam Data

As reported in [BCD+08], spammers often try to game the search result ranking by fabricating incoming links from link farms, which are usually also spammers deploying the same spamming strategies. Moreover, these incoming spammer pages are often created from the same web page template at a very low cost.

As a result, the incoming neighborhoods of spammers are extremely homogeneous or heterogeneous compared to those of normal ones that are gradually built up. In other words, web spammers are expected to be ranked at the top or bottom as measured by the homogeneity of their incoming neighborhood.

Given a node e, we define the **Incoming Neighborhood Feature** of a feature f in such a way that a node with homogeneous incoming neighborhood has a small value, while a node with heterogeneous incoming neighborhood has a large value. This value is defined as $median_{e',e'' \in \textit{1-hop incoming neighborhood of } e,\ e' \neq e'' \neq e} |e'.f - e''.f|$.

We extract the web host graph from the WEBSPAM-UK2006 dataset[4] published by Yahoo! Research. We adopt the 96 content features provided by [CDG+07] [BCD+08], where the features of a host are represented by its home page as well as the page with the highest PageRank score on the host. We compute 6 structural features at the host level, including the number of 1-hop and 2-hop incoming neighbors. The incoming neighborhood features are derived from these content and structural features and are used to rank web hosts in the following experiments.

We iteratively remove entities with less than 2 incoming neighbors, assuming they are not spammers. This is because spammers are more likely to have many incoming neighbors and those spammers having few incoming neighbors are of little spamming power anyway. This leaves one big connected web host graph, with 5634 nodes (1709 spammers) associated with 102 features.

3.7.1 Effectiveness of ERAC Detection Algorithm

Since some of the 102 features in the web spam dataset are dependent on each other, we apply the ERAC detection algorithm designed for dependent feature set to compute the top-K ERACs of size no greater than N. As the heuristic algorithm ERACD_H is the most efficient one among all proposed ones, we apply it to detect ERACs in real-life datasets. We set K=1000 and N=12, so that ERACD_H can finish in a few hours.

[4] http://barcelona.research.yahoo.net/webspam/datasets

Applying the ERACD_H algorithm, we formed 258 maximal IFSs from the 102 features. Among the maximal IFSs, 21 are of size 3, the largest size of all. The top ERAC returned by ERACD_H consists of 12 hosts, all true spammers (including *englandguide.co.uk* and *posters.co.uk* that are still actively spamming despite having being labeled as spammers since 2006). This collection is associated with the maximal IFS {"Number of words", "Top 100 corpus precision", "Independent LH" }, where corpus precision refers to the fraction of words that appear in the set of popular terms, and Independent LH is a measure of the independence of the distribution of trigrams in the page content.

Now we look further into the representative extremity indices of this collection, and explain why it is anomalous. It turns out that the web hosts in this collection are clustered in the top 18 positions on "Number of words", top 22 on "Top 100 corpus precision" and top 45 on "Independent LH". This means that the neighborhood of each host in this collection is very homogeneous in terms of number of words, tendency to use very popular keywords, and pattern of using many unrelated keywords. The experiment demonstrates that our approach is able to discover true spammer collections as well as explain why they are anomalous.

3.7.2 Comparison to Spam Detection Approaches

Next, we compare our ERACD_H algorithm with unsupervised TrustRank [GGMP04] [GBGMP06] and supervised decision tree techniques employed in [CDG$^+$07] and [BCD$^+$08]. These spam detection approaches aim at detecting individual spammers, not spam collections. We therefore treat the websites in our top-K ERACs as individual spammers to compare with the precisions of those of TrustRank and decision tree based methods.

TrustRank starts with a seed set of trusted nodes, and propagates their scores by simulating a random walk with restart to the trusted pages. The estimated non-spam mass [GBGMP06] of a page is the amount of score it receives from trusted pages. We refer to this non-spam mass as the trustrank score. The lower the trustrank score of a node, the more likely it is a spammer.

To select the trusted nodes, we follow the guidance of [CDG+07] and randomly sampled 3800 hosts from the **.UK** domain in the Open Directory Project[5]. In computing the trustrank score, we set the probability of following an out-link from a trusted web page to 0.85. We then rank web hosts in ascending order by the trustrank score of their respective home page (hp), as well as the page with the highest page rank (mp). Thus, the hosts at the top are likely to be spammers. We denote the approach involving home pages as TR_hp and the ranked list it produces as $EL(\mathsf{TR\_hp})$. The approach involving the pages with the highest page rank is denoted as TR_mp and its ranked list as $EL(\mathsf{TR\_mp})$.

For decision tree DT, we use J48 of Weka[6] with 5-fold cross validation. The features used are the same set of derived neighborhood features for ERACD_H. To derive a ranked list of web hosts for comparing with other approaches, we sort the hosts in descending order of the prediction values assigned to them by the decision tree. The ranked list is denoted as $EL(\mathsf{DT})$.

Since our heuristic algorithm works incrementally each loop of n from 1 to N outputs the top-K ERACs of size no greater than n, denoted as $\mathbb{S}^*(n,K)$. We therefore are able to compare the ERACD_H approach with collections of various sizes by keeping the intermediate top-K ERACs produced by ERACD_H for various $n \leq N$. In the experiments, we try $n \in \{4,8,12\}$.

Given n and K, let $\tau(n,K) = |\bigcup_{e \in \bigcup_{S \in \mathbb{S}^*(n,K)} e}|$ denote the number of distinct websites in $\mathbb{S}^*(n,K)$. We assume that all the websites in $\mathbb{S}^*(n,K)$ are spammers and compare the top-$\tau(n,K)$ websites of each approach. Let $EL(O,\tau(n,K))$ denote the top-$\tau(n,K)$ websites returned by approach $O \in \{\mathsf{ERACD\_H}, \mathsf{TR\_hp}, \mathsf{TR\_mp}, \mathsf{DT}\}$. The precision of the top-$\tau(n,K)$ websites is defined as $\frac{|EL(O,\tau(n,K)) \cap true\_spammer\_set|}{|EL(O,\tau(n,K))|}$.

Figures 3.7(a), 3.7(b) and 3.7(c) plot the precision against K for ERACD_H versus {DT, TR_hp and TR_mp}. In the figures, ERACD_H is represented by solid line, while the competing approaches are in doted lines.

As we observe in the figures, ERACD_H outperforms DT, TR_hp and TR_mp for all n settings. This demonstrates that our approach, although not specifically de-

[5] http://rdf.dmoz.org/

[6] www.cs.waikato.ac.nz/ml/weka

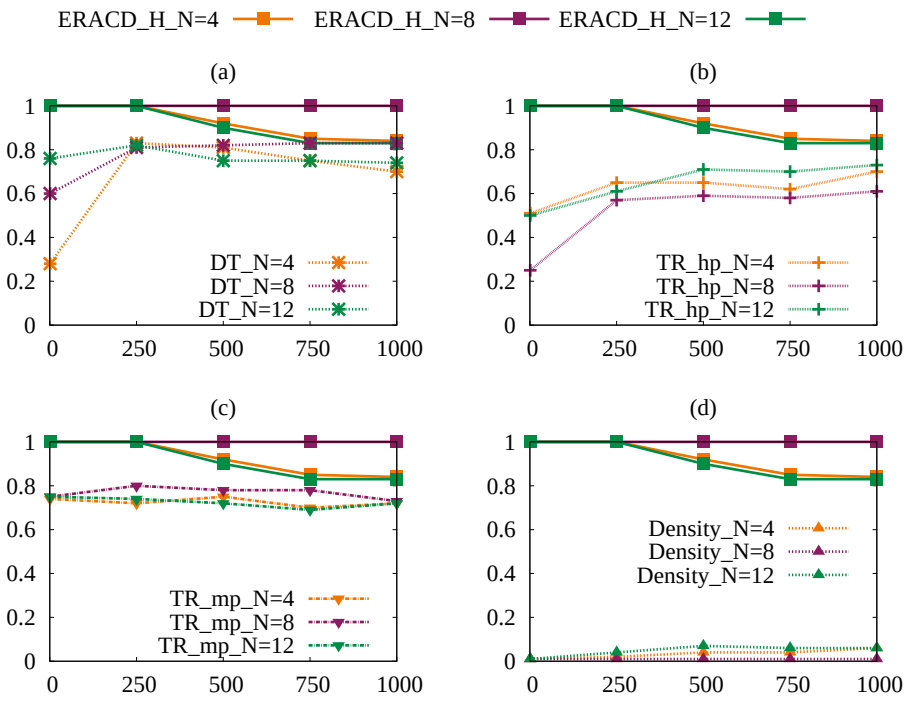

Figure 3.7: Comparison of ERACD_H to other approaches in terms of precision. In each the plot, X-axis shows K as in Top-K, and Y-axis shows precision. Figure(a): ERACD_H vs DT, Figure(b): ERACD_H vs TR_hp, Figure(c): ERACD_H vs TR_mp and Figure(d): ERACD_H vs Density-based

signed to detect spammers, still outperforms the other methods in precision.

The recall levels achieved by our approach are all around 0.03 for $n = 4, 8, 12$ respectively and with $K = 1000$. The low recall levels are expected, as our approach is designed to discover the most anomalous collections of websites, with no attempt to avoid overlap between collections.

Next, to check whether our approach finds unique spammers, we check the overlap between the top-$\tau(n,K)$ websites returned by our approach and that of each competing approach. We define the overlap ratio as $\frac{|EL(\text{ERACD\_H},\tau(n,K)) \cap EL(O,\tau(n,K)|}{\tau(n,K)}$, where $O \in \{\text{TR\_hp}, \text{TR\_mp}, \text{DT}\}$. For $K = 1000$ and $n \in \{4, 8, 12\}$, all the overlap ratios are smaller than 0.05, indicating ERACD_H detects unique spammer hosts that are missed by the competing methods.

3.7.3 Comparison to Anomaly Detection Approaches

Having compared our approach with spammer detection approaches that produce point anomalies, we compare with the general density-based and clustering-based anomaly detection approaches. Note that the density-based approach returns individual websites whereas the clustering-based one returns collections of websites. We are interested to know whether the anomaly score and precision of the collections returned by our approach are better.

We first show the comparison with the density-based approach [BKNS00]. Figure 3.7(d) shows the precision curve. The top websites returned by the density-based approach are mostly non-spammers. This is because the density-based approach assumes anomalies appear in sparse regions. However, true spammers are likely to employ common spamming tricks, making them less likely to appear in sparse regions of the feature space.

Next, we consider clustering-based outlier detection approaches, which consider "small clusters" to be anomalous. We follow [LTS04] in defining a small cluster as the one with a size smaller than half of the average cluster size.

We apply agglomerative hierarchical clustering with complete link and Euclidian distance to cluster E. The clustering algorithm is run on each maximal IFS calculated by our approach so as to find the most anomalous cluster across all maximal IFSs. In clustering for a given maximal IFS, we stop growing the cluster tree once the combination of the next two clusters would cause the average size of all the clusters to rise above $2 \cdot N$. This is to make sure that all resultant clusters of size smaller than or equal to N are "small" clusters by the definition of the clustering-based approach. Since the clusters returned by our approach are of size no greater than N, we can make a fair comparison with the "small" clusters returned by the clustering-based approach.

In the comparison, we keep the intermediate top-K results as N varies, and show the top-1 ERAC and top-1000 ERAC of ERACD_H for each N together with the most anomalous cluster discovered by the clustering-based approach.

In Figure 3.8, we see that for all $N = \{2,4,6,8,10,12\}$, the collections discov-

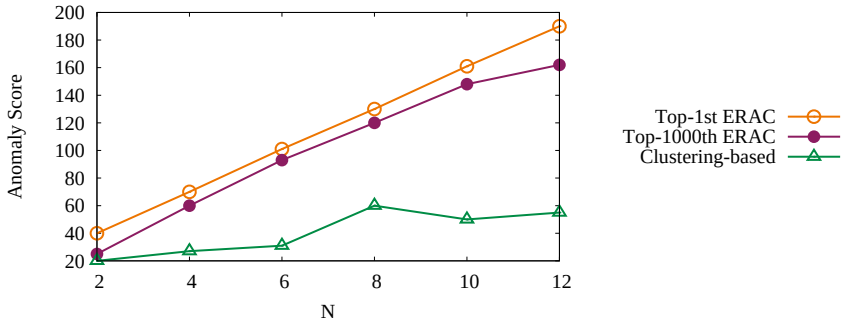

Figure 3.8: ERACD_H vs Clustering-based method on web spam data in terms of anomaly scores

ered by our approach are more anomalous than the clustering-based ones, because ERACD_H is optimized to collections that exhibit extreme behaviors.

To take a close look of the extremity of the top ERAC and the most anomalous cluster produced by the clustering-based approach on each of the three features in the maximal IFS, we use star charts to plot the relative rankings of the members in the two collections. For better visualization, we select the corresponding ERAC and cluster for N=4. The result is shown in Figure 3.9. The center point of each star chart represents the middle ranking (i.e., 2817), as we have 5634 websites in total. The distance to the center represents the extremity in ranking for a feature. It is easy to see that although the members in the most anomalous cluster by the clustering-based approach are similar to each other, they are not extreme on any of the features. In contrast, members of the top ERAC are extreme on all the three features. Interestingly, all the four websites in the top ERAC are spammers, whereas all the websites in the cluster are normal websites.

As for precision, we compute for each given maximal IFS the ratio of the number of true spammers in all the small clusters over the total number of websites in all the small clusters. The maximal ratio is selected as the precision across all maximal IFSs. We also compute this precision for all $N = \{2,4,6,8,10,12\}$. The results show that the precision of the small clusters is quite low for each N, with a maximum of 0.35 at N=10 which is as good as random guess. This suggests that most of the websites in small clusters are not necessarily spammers. Obviously we cannot rely on clustering-based anomaly detection techniques to find spammer collections.

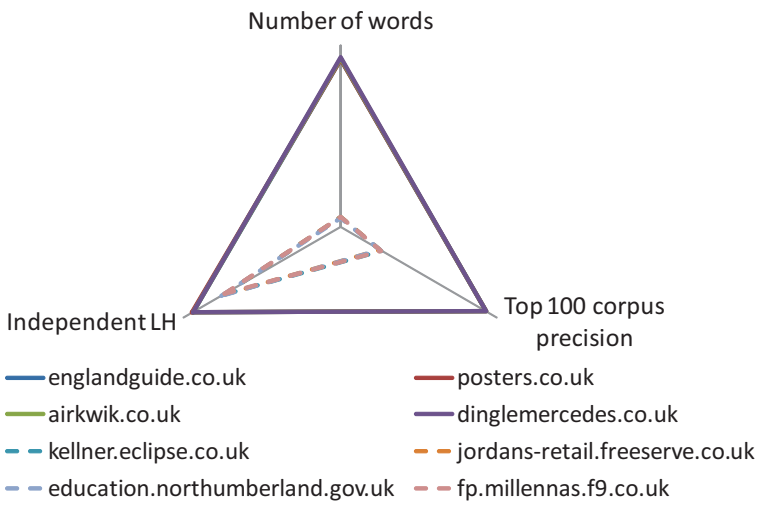

Figure 3.9: Top-1 ERAC of size 4 by ERACD_H (in solid lines) vs the most anomalous cluster of size 4 by clustering-based method (in dotted lines) on web spam data

Method	ERAC expansion	DT	TR_hp	TR_mp
Precision	0.80	0.76	0.71	0.69

Table 3.4: The precisions of ERAC expansion and other competing methods

3.7.4 ERAC Expansion in Web Spam Data

We also apply our expansion algorithm ERAC_exp on this web spam data after we retrieve the top-1000 ERACs for each parameter setting of $N = \{4, 8, 12\}$. For each setting of N, we feed every top-1000 ERACs into ERAC_exp.

Interestingly, for all settings of N, the top-1000 ERACs are expanded to the same ERAC of size 79. This suggests the top-1000 ERACs are part of a larger cohesive ERAC. Note that all detected ERAC of different sizes have the same extreme pattern. In particular, they are located at the extreme top positions of the maximal IFS, {"Number of words", "Top 100 corpus precision", "Independent LH" }. This is the reason why the expansion algorithm ERAC_exp is able to retrieve the same ERAC for all the settings. The anomaly score of the expanded ERAC of size 79 is of 1345.34, much larger than the original ERACs of small sizes, which suggests ERAC_exp successfully retrieve the much more anomalous superset.

We also measure the precision of this expanded ERAC and compare with the competing methods {DT, TR_hp and TR_mp}. As the expanded ERAC has a size of 79, we take the top-79 spammers returned by each approach and measure their pre-

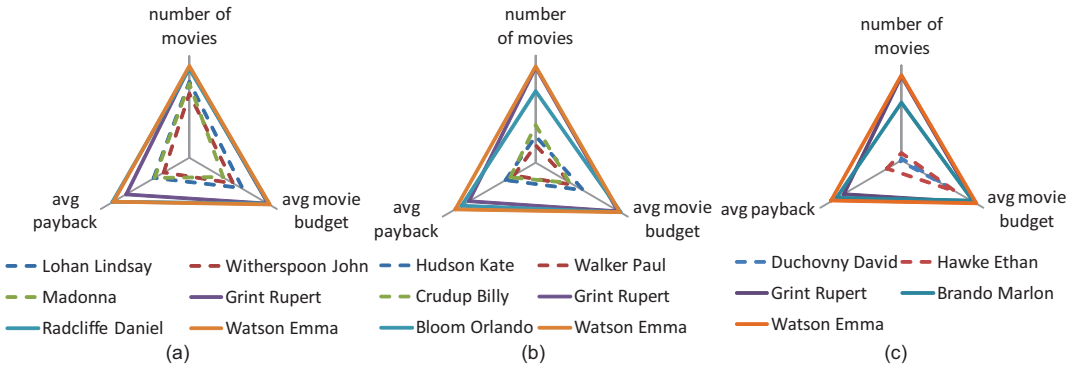

Figure 3.10: ERACD_H vs Clustering-based method on IMDB data in terms of top-1st collections shown in (a), top-2nd collections shown in (b), and top-3rd collections in (c).

cisions. As shown in Table 3.4, our expansion is still more effective in identifying spammers than the competing methods.

3.8 Experiments on IMDB Data

From the IMDB dataset[7], we focus on actors and actresses participating in movies shown between 1990 and 2008. We extract actors playing non-trivial roles in each movie by taking only those appearing among the top 10 names in the cast list. We extracted 6 actor features including number of movies, average rating of all movies, average salary, average movie budget, average movie box office and average payback (the ratio of average box office to average salary). After dropping those actors who have missing feature values, we are left with 183 actors.

3.8.1 ERAC Detection in IMDB Data

We apply ERACD_H on this preprocessed IMDB dataset with K=3 and N=3. We set K and N to be small so that the results are easier to analyze. There are 12 maximal IFSs extracted from the 6 features, including two maximal IFSs with the largest size 3. Each maximal IFS returns a top ERAC. According to our definition in Section 3.3.4, we choose the maximal IFS that leads to the most anomalous ERAC, which

[7]http://www.imdb.com/interfaces

is {number of movies, average movie budget, average payback}.

Using this maximal independent feature set, we also run the clustering-based approach on the IMDB dataset. Since we set N=3 for ERACD_H, it is only fair for the clustering-based approach to assume that clusters of size no greater than 3 are anomalous. Altogether, the clustering-based approach returned 29 clusters, including 3 anomalous ones, all of which are less anomalous than the top-3 ERACs returned by ERACD_H.

Specifically, the three anomalous clusters returned by the clustering-based approach are {Lohan Lindsay, Witherspoon John, Madonna}, {Hudson Kate, Walker Paul, Crudup Billy} and {Duchovny David, Hawke Ethan}. The top-3 ERACs are {Grint Rupert, Radcliffe Daniel, Watson Emma}, {Bloom Orlando, Grint Rupert, Watson Emma}, {Brando Marlon, Grint Rupert, Watson Emma}. We plot their relative feature rankings in Figure 3.10(a), Figure 3.10(b) and Figure 3.10(c). The center point of each star chart represents the middle ranking (i.e., 91), as we have 183 actors. The distance to the center point represents the extremity in ranking for a feature. By visual inspection, we find that the ERACs are more extreme on all the three features, compared to the three clusters returned by the clustering-based approach. This demonstrates again that while clustering-based approaches could return sets of similar actors, they are not designed to capture interesting collections that exhibit extreme behavior.

We note that several of the ERACs are potentially useful in practice. For example, a movie producer may want to find actors who perform in few large-budget movies but have big payback. Our results suggest the producer should go for the Harry Potter actors, or other top ERACs.

3.8.2 ERAC Expansion in IMDB Data

Algorithm ERAC_exp is applied to the top-3 ERACs detected by ERACD_H. Since all the top-3 ERACs have the same extreme patterns on the maximal IFS {number of movies, average movie budget, average payback}, they expand to the same ERAC of size 12, which is {Grint Rupert, Radcliffe Daniel, Watson Emma, Bloom Orlando,

Brando Marlon, Dane Eric, Donner Richard, Englund Robert, Frakes Jonathan, Gibson Tyrese, Howard Bryce Dallas, Jackson Janet}. As expected, this expanded ERAC has the same extreme pattern as the top-3 ERACs. The anomaly score grows from 37.3 for the original top ERAC of size 3, to 53.18 for the expanded superset of size 12.

To compare fairly with our ERAC detection algorithm, the clustering-based algorithm assumes that clusters that are no larger than 12 are anomalous, as our expanded ERAC is of size 12. Altogether, the clustering-based approach returned 8 clusters, including 2 anomalous ones. One cluster is of size 12 with anomaly score of 13.7, which contains {Gere Richard, Chase Chevy, Baldwin William, Jackman Hugh, Zellweger Renee, Daniels Jeff, Penn Sean, Barrymore Drew, Hackman Gene, Duchovny David, Vaughn Vince, Grant Hugh} The other is of size 8 with anomaly score of 6.84, which contains {Scorsese Martin, Silverstone Alicia, Jackson Samuel L., Farrell Colin, Kirkland Sally, Bower Michael, Brosnan Pierce, Ryder Winona}. They are much less anomalous than the expanded ERAC. These results show that our ERAC expansion successfully retrieves a larger ERAC that is more anomalous than the original ERACs as well as the anomalous clusters returned by the competing method.

3.9 Experiments on Chinese Online Forum Data

In this section, we apply our algorithm ERACD_H to identify the infamous Chinese online "water army" (网络水军) spammers[8], who are hired to post or comment on threads in many popular online forums, with the aim of influencing public opinion on targeted events or products. It is reported that during the Qihoo 360[9] vs Tencent QQ[10] dispute, both sides hired water armies to post favorable comments on themselves while disparaging the other[11]. Another reported event is in the Chinese dairy industry, where the brand manager of MengNiu(蒙牛) company was arrested for

[8] http://en.wikipedia.org/wiki/Internet_Water_Army

[9] www.360.cn, the number 1 computer network security service provider in China

[10] QQ.com, the number 1 instant messaging and online community service provider in China

[11] http://www.chinadaily.com.cn/bizchina/2010-11/05/content_11509557.htm

hiring online water army to frame the competing company YiLi(伊利)[12].

Water army operates by soliciting various tasks from internet public relations agencies (e.g. shuijunwang.com) and get paid according to the quality and quantity of posts or comments generated on popular online forums. It is often the case that hundreds or thousands of spammers are hired to form a water army that participates very actively in many popular online forums.

Each water army spammer commonly post comments from multiple user accounts. This helps to hide his/her true identity, as readers may not suspect that these comments come from the same person due to the different account names. If some lazy spammers simply copy and paste his message multiple times in the same post or in multiple posts, they do not get paid by the agencies, as the spammers are instructed not to perform such easily detectable behavior and should post relevant and meaningful messages using different accounts. This makes detecting water army spammers difficult, as they put in effort to post non-repetitive and relevant comments to appear like normal users.

We detect spammer collections in Tianya.cn[13], one of the most popular online Chinese forums. We focus on three hot events namely, 360 vs QQ, Yaojiaxin(药家鑫), and Ligangmen(李刚门), which are highly suspected[14] to involve water army activities. For each event, we submit the event name as query to the search interface of Tianya.cn, which returns the 750 most relevant threads. We keep only those threads that are posted within the relevant period of each event. The start and end dates are determined from news reports and are shown in Table 3.5.

With the relevant threads, we extract all the user accounts that have posted in the threads. We then filter accounts by their membership scores. The membership score assigned by Tianya.cn reflects how active an account is and how much contribution it has made writing acknowledged posts and getting involved in different interactive activities. Since a water army spammer normally has many accounts and would not take too much effort to build up membership scores for each of them, we remove the top 50 percent of accounts that have high membership scores. We also

[12]http://www.chinadaily.com.cn/china/2010-10/21/content_11437735.htm
[13]http://www.tianya.cn
[14]http://www.shuijunshiwan.com/wenku/

events	time period	num of threads	num of users
360 vs QQ	2010.9-2010.12	413	3545
Yaojiaxin(药家鑫)	2010.10-2011.4	345	6443
Ligangmen(李刚门)	2010.10-2011.1	359	3269

Table 3.5: 3 focused events for detecting water army in Tianya.cn

filter out accounts with only one post or comment, as they are also not likely to be involved in water army. Table 3.5 summarizes the profile of the three events after the aforementioned preprocessing.

Given a particular event, we treat each thread of this event as a feature, and the number of times that an account comments in this thread as the feature value. When accounts are ranked according to each thread in descending order, we expect collections of water army spammers to appear in top positions of a few threads, as a few spammers are collaborating, or the same spammer is using multiple accounts in these threads. This collective extreme behavior of water army spammers fits our ERAC principles. Note that a normal active account may also appear in the top positions of a few threads, but not likely in the top position of many threads consistently.

3.9.1 Effectiveness of ERAC Detection Algorithm

We set $N = 5$ and $K = 1$ for our heuristic algorithm ERACD_H. The results of the top ERAC returned for each event are listed in Table 3.6. The third column $|F^s|$ denotes the number of threads w.r.t. which the ERAC is significant (i.e., the corresponding representative p-value is below α). For each significant feature, we find the corresponding extremity index r and intersection i value as in $p(i,r,n)$. avg-r and avg-i in the fourth column shows their average values.

For example, in the event of 360 vs QQ, the top ERAC is {湃大性,ayaya118,时光倒溜,大红鹰的翅膀,让读书人有钱}. By checking the representative p-value of this collection w.r.t. each of the 357 threads, we know their collective extreme behavior is significant in 13 threads, with average r=50.2 and average i=3.7. This suggests that they actively commented on the 13 threads. Reading through the com-

events	top ERAC	$\|F^S\|$	avg-r (avg-i)
360 vs QQ	湃大性,ayaya118,时光倒溜 大红鹰的翅膀,让读书人有钱	13	50.2(3.7)
Yaojiaxin (药家鑫)	武者刺,probit,国强民负 洪德阀紫,天易在线	35	67.3(2.9)
Ligangmen (李刚门)	www9w,热血春秋,muzi840719 曾经是败类，苍天不开眼	15	45.8(3.6)

Table 3.6: Top ERAC for each event in Tianya.cn

ments of the five accounts, it is clear that their posts are either against 360 or for QQ and the posts are generated within a 5-day period. Furthermore, most of their comments were posted to the threads related to 360 vs QQ. All the above observations clearly indicate collaboration among the five accounts.

3.9.2 User Evaluation

Since we do not have the ground truth on water army spammers in Tianya.cn, we recruit human evaluators to judge the accounts on both individual level and collection level.

We hired four Chinese evaluators who are familiar with Tianya.cn forum and the three hot events. They are requested to read the Chinese Wiki page and news reports on water army beforehand. For each user account, the evaluators are provided with the account's homepage in Tianya.cn, which contains the membership score, number of visits, last visit time, registration time and all the threads the account has posted or commented on. As Tianya.cn lists each account's threads page by page, it is very tedious for the evaluators to navigate through the threads and the comments posted from the account. We therefore crawled each account's home page, all the threads as well as the comments in each thread, and presented them together to the evaluators.

User Evaluation at Individual Level.

Firstly, we are interested to know whether the user accounts in the top-K ERACs really are water army spammers on Tianya.cn.

To alleviate the demand placed on the evaluators, we only select accounts in the top ERAC of each focused event, altogether 15 of them, for judging. The evaluators are asked to judge whether each account is a spammer according to his understanding of how a normal account in any online forum should behave. The evaluators assign a score between 1 and 5 to each account, with 1 being normal, 5 being spammer and 3 being not-sure. The final score of an account is the average score assigned by the evaluators. We consider the accounts with final score greater than 3 to be spammers and the rest to be normal.

The scores from the evaluators are shown in Table 3.7. Out of 15 accounts identified by our approach, 13 of them are judged by the evaluators to be spammers. This gives a precision of 0.87.

For user account "大红鹰的翅膀", the evaluators report that it has not logged in after the event of 360 vs QQ, almost all of its threads are about 360 vs QQ, and almost all of its comments contain links to a voting website supporting 360. These observations cast strong suspicions on the account.

There are two accounts "热血春秋" and "武者刺" that the evaluators give low scores as they find that although the accounts are involved in many threads on the corresponding event, they also commented on other events, which makes them less suspicious. Our approach considers only the number of comments across threads but not the actual content of the comments and therefore misclassify them.

User Evaluation at Collection Level.

Now we conduct our user evaluation at collection level. We are interested to know (I) Are the accounts in the top ERAC spamming in the corresponding significant threads? (II) Are the ERAC rankings consistent with the evaluators' perception? In this user evaluation, we focus on the 360 vs QQ event.

We derive a random set of accounts to compare with our top ERAC. We randomly choose from the users whose number of comments are comparable to the average number of comments of the accounts in the top ERAC. This way, the chosen accounts are as active as the members in top ERAC. We end up with the ERAC {珠珠雨粒,treegreen12010,爱顶嘴0,再见能否,quanyeke}.

event	user count	E1	E2	E3	E4	avg
360 vs QQ	大红鹰的翅膀	5	5	5	5	5
	湃大性	3	4	4	4	3.75
	让读书人有钱	5	3	5	4	4.25
	时光倒溜	4	4	4	4	4
	ayaya118	5	3	5	5	4.5
(李刚门) Ligangmen	www9w	3	3	3	4	3.25
	热血春秋	3	1	1	2	1.75
	曾经是败类	4	4	5	3	4
	苍天不开眼	5	5	4	4	4.5
	muzi840719	5	4	4	4	4.25
(药家鑫) Yaojiaxin	国强民负	3	5	4	2	3.5
	武者刺	3	2	2	3	2.5
	天易在线	4	4	4	4	4
	洪德阀紫	2	4	5	5	4
	probit	3	5	5	4	4.25

Table 3.7: User evaluation at individual level. E1 to E4 denote the four evaluators

ERACs	E1	E2	E3	E4	avg
top-1	2.56	3.18	4.01	3.08	3.21
random collection	1.14	1.21	1.91	2.17	1.61

Table 3.8: User evaluation at collection level. E1 to E4 denote the four evaluators

We present the 5 accounts in the top ERAC with its 13 significant threads, together with the 5 accounts in the random user collection with its 16 significant threads to the evaluators. For each account, the evaluators are given its comments in all the significant threads and asked to judge whether the account is spamming in a particular thread based on the comments it posted.

For each account, the evaluator assigns a score from 1 to 5 according to each of the significant thread, with 1 being non-spamming, 5 being spamming and 3 being not-sure. The evaluators also need to provide their reasons.

The final score of an ERAC is the average score of its member accounts across the significant threads. Table 3.8 shows the results. As we see from the table, the average score of the top ERAC is larger than 3, indicating that its members indeed are spamming the significant threads. We also observe that the evaluators consider the top ERAC more suspicious than the less anomalous random account collection, suggesting our ranking are in line with the evaluators' perception.

3.9.3 ERAC Expansion in Chinese Online Forum Data

The previous experiments on detecting water army spammer collections revealed ERACs with predefined size N=5. To uncover water army collections that are supersets of the detected ERACs, we expand the top-K ERACs returned for the event of 360 vs QQ by the ERAC_exp algorithms described in Section 3.4.2.

We first set $N = 5$, $K = 100$ and run ERACD_H to return the top-100 ERACs of size up to 5. We then apply ERAC_exp to expand each ERAC.

Interestingly, 67 of the 100 ERACs expand to the same ERAC of size 11, whereas the remaining 33 ERACs are expand to another ERAC of size 14. The first expanded ERAC of size 11 is significant on 13 features, which are exactly the same 13 features on which the original 67 ERACs of size 5 are significant. On the other hand, the second expanded ERAC of size 14 are significant on 15 features, which overlap with the significant features of the original 33 ERACs on 14.63 features on average.

These observations indicate that the added members in the expanded ERACs have very similar extreme patterns as members in the original ERACs. Specifically, in Tianya.cn, the accounts in the same ERAC posted many times on almost the same set of threads, which substantiates our expansion heuristic.

The 67 ERACs of size 5 have a maximum anomaly score of 314.98, whereas the expanded ERAC of size 11 scores 661.41. On the other hand, the maximum anomaly score of the 33 ERACs is 309.67, much less than 754.36, the anomaly score of the ERAC of size 14 after expansion. This shows that ERAC expansion indeed produces much more anomalous supersets.

Another observation is that the two expanded ERACs do not overlap, neither do their significant features. After reading through the posts from users in each expanded ERAC on their corresponding significant features (i.e., threads), we discover that the two collections of users had condemned 360 harshly with emotive remarks, but providing little factual support. This suggests the existence of at least two independent water army collections spamming on Tianya.cn on behalf of QQ.

3.10 Summary

In this chapter, we detect a new type of anomaly collections, called extreme rank anomaly collections (ERAC). Members of an ERAC exhibit similar extreme behaviors and thus appear at extreme ranking positions of multiple features. Due to the existence of large number of ERACs of various sizes, for efficiency reasons, we first propose the problem of discovering top-K ERACs with a predefined size limit. To uncover the anomalous supersets of the detected ERACs, we then propose the problem of ERAC expansion without having to specify the size of supersets.

We apply ERAC detection and expansion algorithms to discover injected ERACs in synthetic datasets, web spammer collections in a web spam dataset, unusual actor collections in an IMDB dataset and water army spammer collections in a Chinese online forum dataset. The results show that our algorithms are able to uncover the anomalous collections in all datasets. Moreover, we achieve higher precision in web spam detection than existing approaches. We detect anomalous actor collections that are not easily identified by other approaches. We reveal collaborating water army spammer collections in the Chinese online forum, which are agreed by human evaluators.

Chapter 4

Detecting Coherent Anomaly Collections

4.1 Introduction

In the previous chapter, we describe our ERAC work, which better serves the applications where entities are likely to participate in multiple anomalous collections. When users are interested in disjoint anomaly collections, ERAC work is not suitable due to the overlap of members among different ERACs. Furthermore, the ERAC definition does not capture the coherence in the unusual behavior of members in an anomaly collection. We therefore propose a new anomaly collection definition, *Coherent Anomaly Collection(CAC)* emphasizing the *shared* anomalous behavior patterns among members and propose the problem of detecting top-K disjoint CACs.

Examples of this kind of coherent anomaly collections are prominent in social media. In Twitter, a group of users may collaboratively spam on popular hashtags to promote their websites or businesses. Their strategy is to post a large number of tweets containing both their advertisement content and the popular hashtags, so that other users querying any of these hashtags would see their spamming tweets. These activities are classified as spamming according to Twitter's rules[1]. Let us consider a real example.

[1] http://support.twitter.com/articles/18311-the-twitter-rules

Figure 1.2 shows three collections of real spammers in Twitter detected by our approach. Members of each collection post tweets with no real content other than a large number of hashtags appended with short URLs. The URLs show that group S_1 promotes travel guidance books for various cities including London and Hong Kong, corresponding to the hashtags they use for spamming. S_2 and S_3 promote the same pornography website in Brazil. As the unusually high usage of popular hashtags offers a clue for finding these spammers, we rank all the users in descending order by the number of times they used a hashtag. Here we show four popular hashtags (#london, #hongkong, #philipines and #mongolia), and the 34 top-ranked users out of 1899 for each hashtag. We observe that members of the three identified spammer collections are highly consistent in their heavy usage of the same set of hashtags. In contrast, for users in the blank cells, despite their heavy hashtag usage, they are not found to be spammers because their usage patterns do not appear to be shared by other users.

These observations show that the key to detecting suspicious collaborative accounts is to identify their *shared* anomalous behavior patterns. We call such a user group a *Coherent Anomaly Collection(CAC)*, and propose an information theory based definition to characterize it. We mine top-K disjoint coherent anomaly collections. Furthermore, to relieve the burden from users, they do not need to specify either the number or sizes of the target collections as we do in ERAC work.

The rest of the chapter is organized as follows. We formulate our problem in Section 4.2. Our exact algorithm is presented in Section 4.3, followed by the heuristic algorithm described in Section 4.4. Section 4.5, Section 4.6, Section 4.7 and Section 4.8 report on experiments. We conclude in Section 4.9.

This chapter is based on our publication in ACM International Conference on Information and Knowledge Management (CIKM 2012) [DZLP12c].

4.2 Coherent Anomaly Collection

The definition of coherent anomaly collection or CAC is based on the definition of ERAC in the previous chapter. We first show how to measure the coherence,

Notation	Meaning	Notation	Meaning
E	the universal entity set	F	a feature set $\{f\}$
S	an entity collection	F^S	significant feature set
$r_f(S)$	representative r of S on f	$S_f(r)$	entities in S within r on f
$p_f(S,r)$	p-value of S w.r.t. r on f	$\widehat{p}_f(S)$	S' representative p-value on f
$M(S)$	the extremity matrix of S		

Table 4.1: Notations

followed by the problem definition. The frequently used notations are summarized in Table 4.2.1.

4.2.1 Measuring Coherence

By definition, the anomaly score of an entity collection is determined by the subset of its members which are most extremely ranked w.r.t. some features. It is possible that different subsets of members are extremely ranked w.r.t. different feature subsets. However, for many applications, we are most interested in ERACs whose members are extremely ranked w.r.t the same set of features. For instance, in our Twitter example in Figure 1.2, we prefer to identify S_1, S_2 and S_3 as three different ERACs instead of consider them as a single ERAC.

This kind of coherent behavior pattern as a group, which usually reflects highly coordinated activities, could serve as a strong indication that members of the ERAC have engaged in collaboration. To capture this important notion of coherence in our problem definition, we formally define "coherence" by first representing an ERAC in a matrix form and using the matrix encoding cost from information theory in [CPMF04] to evaluate the coherence of the ERAC.

For a given ERAC S and its significant feature set F^S, we denote $E(S,F^S)$ as the members of S that appear in the positions indicated by the representative extremity index of any significant feature, i.e., $E(S,F^S) = \bigcup_{f\in F^S} S_f(r_f(S))$.

To tell how coherent an ERAC is, we represent it by a $|F^S|$ by $|E(S,F^S)|$ matrix. Specifically, given an ERAC S, its significant feature set F^S and its extreme subset $E(S,F^S)$, with f_a,$(a = 1,...,|F^S|)$ being the a-th feature in F^S and e_b,$(b = 1,...,|E(S,F^S)|)$ being the b-th entity in $E(S,F^S)$, the **extreme matrix** is $M(S) =$

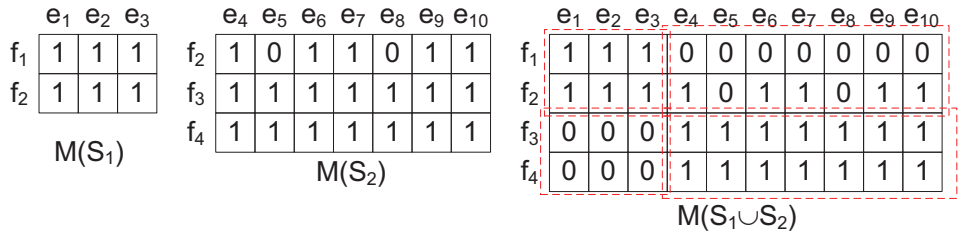

Figure 4.1: The extreme matrixes of $S_1 = \{e_1, e_2, e_3\}$, $S_2 = \{e_4, ..., e_{10}\}$ and $S_1 \cup S_2$ in Figure 1.2. S_1 and S_2 are coherent but $S_1 \cup S_2$ is not.

$[m_{ab}]$, where

$$m_{ab} = \begin{cases} 1, & \text{if } e_b \in E_{f_a}(S, r_{f_a}(S)); \\ 0, & \text{otherwise.} \end{cases}$$

For example, given the collections in Figure 1.2, we show the extreme matrixes of S_1, S_2 and $S_1 \cup S_2$ in Figure 4.1. Given $E(S_1, F^{S_1}) = \{e_1, e_2, e_3\}$ and $F^{S_1} = \{f_1, f_2\}$, $M(S_1)$ contains all 1s. Similarly, we derive $M(S_2)$ and $M(S_1 \cup S_2)$.

According to [CPMF04], any matrix can be encoded as one or more row and column clusters. The encoding cost is the sum of the coding cost and description cost, where the first cost is for encoding each row and column cluster and the second cost is for describing the grouping information and the exceptions. If a matrix is highly homogeneous, e.g., it contains all 1s or all 0s like $M(S_1)$, its encoding cost as one cluster is low.

If a matrix is not homogeneous, e.g., it contains multiple homogeneous clusters like $M(S_1 \cup S_2)$, we should be able to find a minimum cost to encode this matrix by encoding each homogeneous cluster within and describing the grouping information of these clusters. in Figure 4.1, encoding $M(S_1 \cup S_2)$ as two row clusters and two column clusters, as indicated by the red rectangles with dotted line, would have a lower cost than encoding it as one cluster. This is because, for the upper-right cluster, conceptually we use 1 bit to code it as a matrix with all 0s and 5 bit to describe the 5 exceptions (i.e., five 1s); for the other three clusters contain purely 1s or 0s, we only need 1 bit to encode each cluster. Thus $M(S_1 \cup S_2)$ as four clusters needs 9 bits totally for encoding. In contrast, if we encode $M(S_1 \cup S_2)$ as one cluster, we need 1 bit to code it as a matrix with all 1s and 15 bits for describing where the fifteen 0s are, leading to a total of 16 bits, larger than that of four clusters.

Definition 4. *[CAC] Given an entity universe E and an entity set S, $S \subset E$, a set of independent features F and a threshold α, S is a* ***Coherent Anomaly Collection (CAC)*** *if (I) S is an ERAC; (II) the cost of encoding $M(S)$ as one cluster is lower than the minimum cost of encoding $M(S)$ as multiple homogeneous clusters; (III) the number of 1s in its extreme matrix must be greater than half of the size of its extreme matrix.*

We need condition (III) as the information theory based condition does not distinguish between matrix M with mostly 1s and the matrix $I - M$ with mostly 0s. In other words, the extreme matrix of an ERAC containing mostly 0s can satisfy condition (II), but we do not consider this ERAC to be coherent. In this case, most members of the corresponding ERAC do not even appear in extreme positions w.r.t. most of the significant features.

4.2.2 Problem Formulation and Analysis

We formally define our problem of detecting top-K disjoint CACs as follows.

Definition 5. *[TOPK_CAC] Given K, the entity universe E, a set F of independent features and the ranking of E on F, let $S^* = (S_1, S_2, \ldots, S_N)$ be the sequence of coherent anomaly collections ranked in descending anomaly score order. The problem of TOPK_CAC is to find the length-K disjoint subsequence $\hat{S}$ of S^* where $\hat{S} = (S_{\hat{a}_1}, S_{\hat{a}_2}, \ldots, S_{\hat{a}_K})$ such that (I) $S_{\hat{a}_i} \bigcap S_{\hat{a}_j} = \emptyset$ and $1 \leq \hat{a}_i < \hat{a}_j \leq N$, for $1 \leq i < j \leq K$; and (II) for any other length-K disjoint subsequence S' of S^* where $S' = (S_{a'_1}, S_{a'_2}, \ldots, S_{a'_K})$ such that $S_{a'_i} \bigcap S_{a'_j} = \emptyset$ and $1 \leq a'_i < a'_j \leq N$, for $1 \leq i < j \leq K$, there exists an index $j^*, 1 \leq j^* \leq K$ such that $\hat{a}_{j^*} < a'_{j^*}$ and $\hat{a}_i = a'_i$, for all $1 \leq i \leq j^* - 1$.*

Conceptually, we have two ways to approach the problem. One is to scan through S^*, the list of all CACs as stated in the problem definition. However, we can prove that in the worst case any algorithm would need at least a running time on the order of $2^{|E|}$ to find the top-K disjoint CACs from scanning S^*. This is because, by using adversary argument, any algorithm would need to compute at least $|S^*|/c$ portion of S^* for some constant c. In particular, we can prove that for any algorithm computing

less than $|S^*|/c$ portion of S^*, there are TOPK_CAC problem instances such that the algorithm must fail to return even the correct top-2 CACs.

We have the following theorem.

Theorem 2. *Given K, the entity universe E, a set F of independent features and the ranking of E on F, any algorithm would need at least a running time on the order of* $2^{|E|}$ *to find the top-K disjoint CACs.*

Proof. Let S^* be the sequence of all coherent anomaly collections ranked in descending order by their anomaly scores. It is easy to see that, in the worst case, S^* is of size $\Omega(2^{|E|})$, and we can solve the TOPK*_CAC by computing the entire S^*. Thus, the problem complexity is $O(2^{|E|})$. To prove that it is $\Omega(2^{|E|})$, we prove the following: *To solve the TOPK*_CAC problem, any algorithm would need to compute* $\Omega(|S^*|/c)$ *portion of* S^* for some constant c. We prove this by using adversary argument to show that, for any algorithm computing less than $\Omega(|S^*|/c)$ portion of S^*, there are TOPK*_CAC problem instances such that the algorithm must fail to return even the correct top-2 CACs.

The illustrative problem instance we are going to construct is assumed to have four features f_1 to f_4 over a set of entities E. first, we select a set of entities S=$\{e_1, e_2, ..., e_{|S|}\}$, $S \subset E$ to be the top entities on feature f_1 and f_2. To make sure that $\{f_1,f_2\}$ is an independent feature set, i.e., $dep(f_1, f_2) \leq 1.96$ as defined in 3.2.1, we compute the upper-bound of $|S|$ as follows. According to the definition of dependency, $|n_c - n_d|$ directly affects its value, which summarizes the orderings of all entity pairs in f_1 and f_2. We assume the entities pairs from $S \times S$ or $(E-S) \times (E-S)$ makes no contribution to $|n_c - n_d|$, suggesting half of these entity pairs are concordant and the rest half are discordant. This can be easily achieved by randomly permutate entities within S and $E-S$ on both features. On the other hand, the entity pairs from $S \times (E-S)$ contribute $|S| \times |E-S|$ to $|n_c - n_d|$. This is because S sits at the top-$|S|$ positions of both features, every pair of entities from $S \times (E-S)$ are concordant. Therefore, if $dep(f_1, f_2) = 1.96$, we have the equation $|n_c - n_d| = |S| \times |E-S| = \frac{1.96 \cdot \sqrt{|E| \cdot (|E|-1) \cdot (2|E|+5)/2}}{3}$ to tell us the maximum size of S. We take the positive solution of the equation and know that $|S|$ is at least $a \cdot |E|$, where a is constant. In this way, all entities in S are in the extremity index $|S|$,

therefore S is a CAC.

Next, we generate another two random entity lists over E for feature f_3 and f_4. We then switch entities $e_{|S|+1}$ and $e_{|S|+2}$ with the entities sitting on the top-(|S|+1)-th and top-(|S|+2)-th position of f_3 and f_4. This way, (I)F =$\{f_1$,f_2,f_3,$f_4\}$ are independent of each other; (II)$S' = \{e_{|S|+1}, e_{|S|+2}\}$ does not overlap S; (III)$\{f_1$, $f_2\}$ and $\{f_3$, $f_4\}$ are the significant features of S and S' respectively. We can also guarantee the collection S' is a CAC by definition, if $p(2,|S|+2,2) < \alpha/|F|$. This is easily achievable for a reasonably large size E, as the p value deceases as the population size grows. Therefore, we have created another CAC S' of size 2, which is also coherent as both its entities are in the extreme region of $|S|+2$. At the same time, S' is much less anomalous than S, i.e., S is the top-first CAC and S' is top-2nd CAC.

Since S' is a CAC, suggesting $p(2,|S|+2,2) < \alpha/|F|$. According the p-value properties shown in [DZLP12b] [Tal05], $p(i,|S|,n) < \alpha/|F|$, for $2 \leq i \leq n$ and $2 \leq i \leq |S|$. This indicates that any subset of S of size greater than 1 is also a CAC with significant features f_1 and f_2, which leaves us $2^{|S|}$ number of CACs. Since all these subsets of size at least 2 sit at more extreme positions than S' and are at least as large as S', they are therefore more anomalous than S'. In other words, in S^*, there are at least $2^{|S|}$ of CACs ranked higher than S'. Since $2^{|S|}$ is bounded by $2^{a\cdot|E|}$ and $|S^*|$ is also bounded by $2^{|E|}$, we need to go through at least $\Omega(|S^*|/c)$ portion of S^* for some constant c, before finding S' as the top-2nd CAC. Therefore, in this problem instance we created, if any algorithm computing less than $\Omega(|S^*|/c)$ portion of S^*, this algorithm fail to return even the correct top-2 CACs.

Since the size of S^* is of size $\Omega(2^{|E|})$ in the worst case, we conclude any algorithm would need at least a running time on the order of $2^{|E|}$ to find the top-K disjoint CACs. □

Instead of going through S^*, which incurs exponential execution time. The other way to solve the problem is to first find the top CAC and then find the next most anomalous CAC that does not overlap with any of the previously detected CACs and so on, until K CACs are returned. In the following two sections, based on this search paradigm, we design our exact and heuristic algorithms to solve the problem.

4.3 The Exact Algorithm for CAC Detection

In this section, we describe the exact algorithm for solving the TOPK_CAC problem. According to the search paradigm of first finding the top CAC and then the next most anomalous CAC that does not overlap with any of the previously detected CACs and so on, any algorithm would mine the most anomalous CAC with the constraint of being disjoint with a set of entities C, $C \subset E$. We call this C the **constraint set**. To do this, the exact algorithm needs to find the most anomalous CAC of all sizes from 2 to $|E|/2$. Algorithm 8 shows the detail.

Algorithm 8 CACD_E, *detecting exact top-K disjoint CACs*

Input: E, F, K
Output: exact top-K CACs: $\hat{S}$

1: $\hat{S} \leftarrow \emptyset$; $C \leftarrow \emptyset$ {the constraint set C keeps all entities in detected CACs.}
2: **repeat**
3: $n \leftarrow 2$
4: $S \leftarrow topCAC\_size\_E(E,F,n,C)$
5: **while** $n < |E|/2$ **do**
6: $n \leftarrow n$+1
7: $S' \leftarrow topCAC\_size\_E(E,F,n,C)$
8: **if** $S' \neq null$ AND $\Omega(S,F) < \Omega(S',F)$ **then**
9: $S \leftarrow S'$ {when S is null, $\Omega(S,F)$ is 0}
10: **if** $S \neq null$ **then**
11: add S to $\hat{S}$
12: add all entities in S to C
13: **until** $S == null$ OR $|\hat{S}| > K$
14: **return** $\hat{S}$

In step 4 and step 7 of Algorithm 8, function $topCAC\_size\_E(E,F,n,C)$ finds the top CAC of size n that does not overlap with the constraint set C. The idea is to first find the top-k anomalous collections w.r.t. each single feature $f \in F$ that do not overlap with the constraint set C. From these collections, we then derive the top CAC w.r.t. the given set of feature F. Specifically, we start from top anomalous collections w.r.t. each feature and gradually pull in the next most anomalous collections w.r.t. each feature until the exact top CACs w.r.t. F is guaranteed.

We will first discuss for any given k, how to derive the k-th most anomalous collections w.r.t. any feature that do not overlap with the constraint set C, followed by the detailed algorithm of function $topCAC\_size\_E(E,F,n,C)$, with a stopping criterion on k in order to guarantee the exact top CAC w.r.t. F.

4.3.1 Deriving k-th most Anomalous Collections w.r.t. Single Feature

To derive the k-th most anomalous collections w.r.t. a feature, a naive way is to enumerate all collections and sort them by their anomalousness w.r.t this feature. However, the number of collections may be too large to enumerate. Besides, not all enumerated collections will be qualified in the final top CAC w.r.t. F. We propose to derive the k-th most anomalous collections by the ordering of p-values.

Recall that given an entity set S and a significant feature f, the anomaly score of S w.r.t. f depends on the representative p-value of S w.r.t. f. If we can order the representative p-values of all possible collections of size n, and then find the set of collections corresponding to each of these p-values, we will be able to derive the collections with larger anomaly score w.r.t. f. In other words, the ordering of collections by anomaly score can be derived from the ordering of p-values.

Ordering of P-values.

Recall that given any collection S of size n, the p-value of S is determined by the extremity index r and i which is the number of entities in S that appear in top-r positions. Hence any p-value can be represented as $p(i,r,n)$.

For all collections of size n, we could derive their representative p-values by enumerating all possible (i,r,n) combinations with the constraints $1 \leq r < |E|/2$, $1 \leq i \leq min(r,n)$. However, the total number of (i,r,n) combinations is large. Furthermore, we are more interested in those with the small p-values, as they indicate more anomalous collections. We therefore make use of the intrinsic partial orderings among $p(i,r,n)$ values for deriving the next smallest p-value without enumerating all p-values.

To illustrate, we organize the $p(i,r,n)$ values of the same i into the same column and order the columns by descending i from left to right. In each column, we list the p-values according to the r values in ascending order from top to bottom. We call the first p-value of a column its **anchor**. Suppose $|E| = 10$ and $n = 3$, all the

column-3	column-2	column-1
p(3,3,3)=0.008	p(2,2,3)=0.067	p(1,1,3)=0.3
p(3,4,3)=0.033	p(2,3,3)=0.183	p(1,2,3)=0.533
p(3,5,3)=0.083	p(2,4,3)=0.333	p(1,3,3)=0.708
	p(2,5,3)=0.5	p(1,4,3)=0.833
		p(1,5,3)=0.917

Table 4.2: All $p(i,r,n)$ values of $n = 3$.

$p(i,r,n)$ values are shown in Table 4.2.

We utilize two partial orders: column order and anchor order. The **column order** states that within a column, each p-value is smaller than the p-values below. Formally, $p(i,r,n) < p(i,r+1,n)$, for all $i \leq r \leq |E|/2$.

The **anchor order** states that the anchor p-value of a column is smaller than the anchors of all columns to its right. Formally, $p(i,i,n) < p(i-1,i-1,n)$, for all $1 \leq i \leq n$.

To prove these two partial orders is to prove the associated p-value properties of the hypergeometric distribution, which can be found in [Tal05].

With the partial orders, we are able to decide which $p(i,r,n)$ value is the next smallest one, without enumerating all possible $p(i,r,n)$ values. We define the **p-value frontier** of $p(i,r,n)$ as the set of p-values that are the immediate smaller p-values according to each partial order. The partial orders lead to the following Lemmas for deriving the next smallest $p(i,r,n)$ value.

Lemma 5. *The anchor of column-n, i.e., $p(n,n,n)$, is the smallest p-value of all.*

This lemma is straightforward. Firstly according to column order, each anchor is the smallest within the column. Secondly, according to anchor order, the anchor of column-n is the smallest of all anchors and hence the smallest p-value of all.

Lemma 6. *Given any $p(i,r,n)$ value, the next smallest p-value lies in the p-value frontier of $p(i,r,n)$ or the frontiers of the p-values that are no greater than $p(i,r,n)$.*

Lemma 6 holds because according to the definition of frontier, p-values in $p(i,r,n)$'s frontier are larger than $p(i,r,n)$ and smaller than those p-values in their frontiers.

According to these two lemmas, starting from the smallest p-value $p(n,n,n)$,

we just need to keep all the frontiers of the p-values being searched, and find the smallest one in the frontiers to be the next smallest p-value.

In Table 4.2, the frontier of $p(3,3,3)$ contains $p(3,4,3)$ by column order, and $p(2,2,3)$ by anchor order. The p-value frontier of $p(3,4,3)$ contains $p(3,5,3)$ by column order. Note that the elements in the frontiers of p-values that are smaller than $p(i,r,n)$ can be smaller than the elements in the frontier of $p(i,r,n)$. Suppose the current p-value is $p(3,3,3)$, the next smallest p-value is among its frontier $\{p(3,4,3),\ p(2,2,3)\}$. If the current p-value is $p(2,2,3)$, then the next smallest p-value is in its frontier $\{p(2,3,3),\ p(1,1,3)\}$ or in $p(3,3,3)$'s frontier. From these frontiers, we know the next smallest p-value after $p(2,2,3)$ is $p(3,4,3)$.

Deriving the Set of Collections Whose Representative P-value is $p(i,r,n)$.

Now the question is how to derive the set of collections whose representative p-value is of a given $p(i,r,n)$. For a given $p(i,r,n)$, multiple collections have $p(i,r,n)$ as their representative p-value. For example, in Figure 4.2, in the entity list of f_2, $\{e_1,e_2,e_6\}$, $\{e_1,e_3,e_6\}$, and $\{e_2,e_3,e_6\}$ all have representative p-value of $p(3,4,3)$. Here, we describe given $p(i,r,n)$, how to find the set of size-n collections with the representative p-value of $p(i,r,n)$ w.r.t. any feature f. The idea is that we enumerate all the corresponding size-n collections by moving n non-overlapping indices, each index points to a ranking from 1 to $|E|$ on the entity list of f. We also need to make sure that these indices skip the entities in the constraint set C.

Next, we need a way of pinpointing individual entities in the entity list corresponding to each feature. Intuitively, we need n pointers, each points to an individual entity. We denote the list of rankings indicated by the n pointers as π. For example, in Figure 4.2, there are three pointers on f_2, with the first one pointing to e_1 (first), the second pointing to e_6 (4th) and the third pointing to e_8 (5th), hence π=(1,4,5). Let $\pi(j)$ denotes the ranking indicated by the j-th pointer counted from left to right, with $1 \leq j \leq n$. E.g. in Figure 4.2, $\pi(1) = 1$, $\pi(2) = 4$ and $\pi(3) = 5$. Note that by definition, $\pi(j) > \pi(j-1)$.

For a given feature f, π uniquely indicates one size-n collection. We denote $E^f(\pi)$ as the set of entities associated with π for f. E.g. in Figure 4.2, $E^{f_2}(\pi) =$

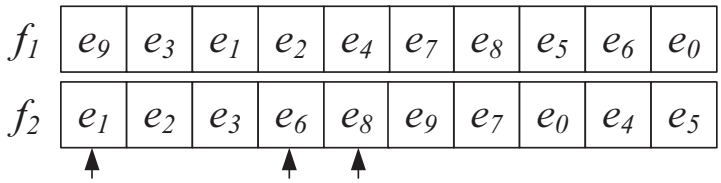

Figure 4.2: 10 entities $\{e_0, ..., e_9\}$ are ranked according to each 2 features $F = \{f_1, f_2\}$.

$\{e_1, e_6, e_8\}$.

Now given a $p(i,r,n)$, we need to figure out where to put n pointers such that $E^f(\pi)$ has $p(i,r,n)$ as representative p-value. Suppose that $i = 3$, $r = 5$, $n = 3$, and we want to get collections in f_2 whose representative p-values are $p(3,5,3)$ in Figure 4.2. Firstly, we know that the third pointer should point to e_8. In general, by the definition of p-value, the i-th pointer should point to the ranking of r, i.e., $\pi(i) = r$.

Regarding the first and the second indices, we may think that the first index ranges from ranking 1 to 3, while the second pointer ranges from 2 to 4. However, this may not be the case, as the corresponding collections' representative p-values may not be $p(3,5,3)$. For example, suppose $\pi(1) = 1$, $\pi(2) = 2$ and $\pi(3) = 5$, then $E^{f_2}(\pi) = \{e_1, e_2, e_8\}$. But $\widehat{p}_{f_2}(\{e_1, e_2, e_8\}) = p(2,2,3)$ instead of $p(3,5,3)$. This is because $p(2,2,3) < p(3,5,3) = 0.5$.

In general, given $p(i,r,n)$ and the constraint set C, we say π is **representatively-correct** iff. (I) $\pi(i) = r$; (II) $\forall j$, $1 \leq j \leq n$ and $j \neq i$, we have $p(j, \pi(j), n) > p(i,r,n)$; (III) $E^f(\pi) \cap C = \emptyset$. In other words, if π is representatively-correct, $E^f(\pi)$ has $p_f(i,r,n)$ as its representative p-value and does not overlap with C.

We now illustrate the process of finding all collections whose representative p-value is $p(3,5,3)$ with the entity list of f_2 in Figure 4.2, assuming the constraint set is empty. We start by putting $\pi(1) = 1$, $\pi(2) = 2$ and $\pi(3) = 5$. Since $\pi(3)$ is pinned to 5, we move $\pi(2)$ rightwards until $\pi(2) = 4$. we then move $\pi(1)$ to its immediate position to its right, i.e., $\pi(1) = 2$ and move $\pi(2)$ back to 3. After we repeat the process of moving $\pi(2)$ rightwards until $\pi(2) = 4$, we then move $\pi(1)$ to $\pi(1) = 3$ and move $\pi(3)$ back to 4. During the whole process, we always output $E^{f_2}(\pi)$ if π is representatively-correct.

Algorithm 9 getcollections

Input: i, r, n, feature f, constraint set C
Output: the set of collections associated with $p(i,r,n)$ denoted as $\hat{S}$

```
1: for j = 1 to n do
2:     if j < i then
3:         π(j) ← j
4:     else if j > i then
5:         π(j) ← r + (j − i)
6:     else
7:         π(j) ← r
8:     while π(j) points to any entity in C do
9:         π(j)++
10:    if π(j) > |E| then
11:        return null
       {Initialize π, if the any pointer points to entities in C, then move to the
       next appropriate position}
12: Traverse(0, π, Ŝ, i, r, n)
13: return Ŝ
```

The detailed algorithm for finding collections whose representative p-value is $p(i,r,n)$ is given in Algorithm 9. This algorithm calls the Traverse function of Algorithm 10 that iteratively calls itself to traverse all ranking positions by changing π. It is easy to see that the execution time of Algorithm 9 is dominated by the Traverse function, which is $O(|E|^n)$. Therefore the time complexity of Algorithm 9 is $O(|E|^n)$.

As we already solve the problem of sorting p-values, with this algorithm, we are able to derive the set of k-th anomalous collections w.r.t. a single feature whose representative p-value is the k-th smallest of all p-values.

4.3.2 Finding Exact Top CAC of Size n

Now, we describe the algorithm for computing the top CAC of a given size n. This algorithm is similar to Fagin's Threshold Algorithm(TA) [FLN01]. We denote the k-th smallest p-value as $p(i^k, r^k, n^k)$, and the set of k-th anomalous collections w.r.t. feature f whose representative p-values are $p(i^k, r^k, n^k)$ as $C(k,f)$.

To find top CAC, we compute the anomaly score of each collection in $\bigcup_{1\leq k\leq K^*}\bigcup_{f\in F} C(k,f)$ and find our top there. Therefore, we will only miss a top

Algorithm 10 Function Traverse$(j, \pi, \hat{S}, i, r, n)$

if $j == n$ **then**
    if $j == i$ **then**
        if π is representatively-correct, add $E^f(\pi)$ to $\hat{S}$
    else
        while $\pi(n) \leq |E|$ **do**
            if π is representatively-correct, add $E^f(\pi)$ to $\hat{S}$
            $\pi(n) \leftarrow \pi(n) + 1$
    return
else
    Traverse$(j+1, \pi, \hat{S}, i, r, n)$
    if $(j == i) \parallel (j < i \;\&\&\; \pi(j) == r - (j - i)) \parallel (j > i \;\&\&\; \pi(j) == |E| - (n - j))$ **then**
        return
    else
        $\pi(j) = \pi(j) + 1$
        for $j = 1 + 1$ to n **do**
            if $j < i, \pi(j) \leftarrow j$; if $j == i, \pi(j) \leftarrow r$; if $j > i, \pi(j) \leftarrow r + (j - i)$.
    Traverse$(j, \pi, \hat{S}, i, r, n)$

CAC if it does not appear in $\bigcup_{1 \leq k \leq K^*} \bigcup_{f \in F} C(k, f)$. To guarantee the exact top CAC, we need to decide K^*, i.e., how far do we search along each single feature. We hence have the following lemma:

Lemma 7. *The upper-bound of the anomaly score of the collections that never appear in top-K^* anomalous collections of any feature (i.e., $\bigcup_{1 \leq k \leq K^*} \bigcup_{f \in F} C(k, f)$) is $-\log p(i^{K^*+1}, r^{K^*+1}, n^{K^*+1}) \cdot |F|$.*

Lemma 7 estimates the upper-bound of collections that never appear among the top-K^* of any feature by assuming they have the "best case p-value" over every feature. The proof of this lemma can be easily derived from [FLN01]. With Lemma 7, we can derive K^* such that the anomaly score upper-bound of the unseen collections is smaller than the smallest score in the current top CAC.

The exact algorithm always maintains the current top CAC during the search process. It starts from $k = 1$ and computes the set of k-th most anomalous collection w.r.t. each feature, i.e., $\bigcup_{f \in F} C(1, f)$. It then compute the anomaly score of each collection w.r.t. F in $\bigcup_{f \in F} C(k, f)$ and update the current top CAC. The algorithm then check whether the upper-bound of the collections that never appear in top-k anomalous collections of any feature is smaller than the anomaly score of the

current top CAC. If the stopping criterion has not been met, the algorithm increases k by 1 and continues to evaluate the collections in $\bigcup_{f \in F} C(k, f)$. Whenever the stopping criterion is met, we stop increase k and return the top CAC.

The details of the exact algorithm is shown in Algorithm 11.

Algorithm 11 topCAC_size_E(E, F, n, C), *the exact algorithm to detect top CAC of size n*

Input: E, F, collection size n, constraint set C
Output: S

1: Let $pt \leftarrow (n, n, n)$ and $S = \emptyset$ { pt keeps the current p-value (i, r, n) tuple}
2: Let $\Psi \leftarrow frontier\_p(pt)$ { frontier() function computes the frontier of the $p(i, r, n)$ indicated by pt}
3: **repeat**
4: **for** each $f \in F$ **do**
5: $\hat{S} \leftarrow getcollections(pt.i, pt.r, pt.n, f)$
6: **for** each $S' \in \hat{S}$ **do**
7: **if** $\Omega(S', F) > \Omega(S, F)$ **then**
8: $S \leftarrow S'$
9: $pt \leftarrow pop(\Psi)$ {pop out the (i, r, n) tuple with the smallest p-value in the frontiers Ψ}
10: $\Psi \leftarrow \Psi \cup frontier\_p(pt)$
11: **until** $\Omega(S, F) > -\log(p(pt.i, pt.r, pt.n)) \cdot |F|$
12: **return** S

We illustrate the exact algorithm topCAC_size_E with the run example in Figure 4.2 with $n = 3$. The algorithm starts with $p(3,3,3)$, and finds $p(3,3,3)$'s frontier. In the repeat until loop, it gets $\{e_9, e_3, e_1\}$ and $\{e_1, e_2, e_3\}$ as the corresponding collection of $p(3,3,3)$ for f_1 and f_2 respectively. After updating the top CAC, the current smallest anomaly score of the collections in Top-K CACs, i.e., $\Omega(\{e_1, e_2, e_3\}, F) = -\log p(3,3,3) - \log p(3,4,3) = -\log 0.008 - \log 0.033 = 8.24$. The upper-bound of the collections that have yet to search is computed as Lemma 7: $-\log p(3,4,3) \times 2 = 6.82$, which is smaller than 8.24. Now the upper-bound is smaller than the current top CAC, the algorithm stops and returns $\{e_1, e_2, e_3\}$ as the final result.

Time Complexity of algorithm CACD_E. We first analyze the complexity of Algorithm 11. Since the time complexity of $getcollections(pt.i, pt.r, pt.n, f)$ is $O(|E|^n)$, in the worse case, the size of $\hat{S}$ is also $|E|^n$. Thus, the time complexity of Algorithm 11 is $O(|F| \cdot |E|^n \cdot \lambda)$, where λ is the number of (i, r, n) tuples popped out in step 9 until the algorithm stops.

Algorithm 8 CACD_E calls Algorithm 11 at step 4 and step 7. It is easy to see that Algorithm 11 is called $|E|/2$ times. Therefore, the time complexity of algorithm CACD_E is $O(K \cdot |F| \cdot |E|^{n+1} \cdot \lambda)$.

4.4 The Heuristic Algorithm for CAC Detection

The high complexity of the exact algorithm leads us to propose heuristics to solve the TOPK_CAC problem by sampling the candidate collections that are potentially more anomalous.

We sample candidates in increasing sizes. This is because, (I) anomalies are minorities, and anomaly collections are generally small; (II) collections of larger sizes may have larger anomaly scores, but they are less likely to be coherent. Before showing the first heuristic regarding sampling candidates from small to large sizes, we define **first-maximal CAC with constraint** C as follows.

Given E and C, let $(S^2, \ldots, S^{|E|/2-1})$ be the sequence of top CACs of size from 2 to $|E|/2-1$, s.t. $S^i \cap C = \emptyset$, $\forall 2 \leq i < |E|/2-1$. The **first-maximal CAC with constraint** C is the S^i, $2 \leq i < |E|/2-1$ such that (I) $\Omega(S^j, F) \leq \Omega(S^{j+1}, F)$, for all $1 < j < i$; (II) $\Omega(S^i, F) > \Omega(S^{i+1}, F)$.

Intuitively, the first-maximal CAC with constraint C is the CAC that does not overlap with C and are more anomalous than all smaller collections. With this, our first heuristic is as follows.

Heuristic 1. ***[first-maximal property of Top-K CACs]*** *Let $\hat{S} = (S_1, S_2, \ldots, S_K)$ be the top-K CACs. $\forall i, 1 \leq i \leq K$, S_i is the first-maximal CAC with constraint $\bigcup_{1 \leq j < i} S_j$.*

This heuristic states that every CAC in the final sequence $\hat{S}$ is the first local maximal with the constraint of being disjoint with other CACs that are more anomalous in $\hat{S}$. Based on this heuristic, we propose the heuristic algorithm CACD_H shown in Algorithm 12. We compute the top CAC incrementally for size starting from 2, and stop the search as soon as the anomaly score of the top CAC drops. The resultant CAC is one first-maximal with constraint C.

Similar to the exact algorithm, in step 7 and step 9 of Algorithm 12, we need to

Algorithm 12 CACD_H, *Heuristically detecting top-K disjoint CACs*

Input: E, F, K
Output: heuristic top-K CACs: $\hat{S}$

```
1: Ŝ ← ∅; C ← ∅
2: repeat
3:     n ← 2
4:     S ← topCAC_size_H(E, F, n, C)
5:     while S ≠ null AND n < |E|/2 do
6:         n ← n+1
7:         S' ← topCAC_size_H(E, F, n, C)
8:         if S' ≠ null AND Ω(S, F) < Ω(S', F) then
9:             S ← S' {when S is null, Ω(S, F) is 0}
10:        else
11:            break {apply heuristic 1}
12:    if S ≠ null then
13:        add S to Ŝ
14:        add all entities in S to C
15: until S == null OR |Ŝ| > K
16: return Ŝ
```

find the top CAC of a given size n. However, the number of CACs of size n is $|E|^n$ in the worst case, as we showed for the exact algorithm. We therefore propose the second heuristic regarding *the importance of local extremity*.

Heuristic 2. ***[Importance of Local Extremity]*** *Given E and F, let S be the top CAC w.r.t. F. There exists a feature $f \in F$ and a small integer threshold θ such that S is among the top-θ CACs w.r.t. f.*

The rationale is that if a collection is not very anomalous w.r.t. any single feature $f \in F$, it is unlikely to be the top CAC w.r.t. F. Based on this heuristic, we first sample collections with large anomaly score w.r.t. each single feature $f \in F$, from which we then derive the top CAC w.r.t. F. To find the collections that are more anomalous w.r.t. a single feature, we take advantage of the ordering of p-values, as we do for the exact algorithm.

4.4.1 Sampling the Set of Collections Whose Representative P-value is $p(i,r,n)$

Now the question is how to efficiently derive the set of collections whose representative p-value is of a given $p(i,r,n)$. For a given $p(i,r,n)$, multiple collections may

$\pi(1)$	$\Omega(\{e\},F)$	$\pi(2)$	$\Omega(\{e\},F)$	$\pi(3)$	$\Omega(\{e\},F)$
$\{e_1\}$	3.51	$\{e_2\}$	2.53	$\{e_8\}$	1.39
$\{e_3\}$	2.81	$\{e_3\}$	2.81		
$\{e_2\}$	2.53	$\{e_6\}$	1.61		

Table 4.3: Candidate entity lists of each $\pi(j)$
for generating entity collections of representative p-value $p(3,5,3)$ on feature f_2.

have $p(i,r,n)$ as their representative p-value. Recall that we have shown in the previous section that the number of collections having the same representative p-value w.r.t. any feature can be as many as $\binom{r}{i}\binom{|E|-r}{n-i}$. We therefore want to sample only a subset of these collections that have larger anomaly score not only w.r.t. a single feature, but also w.r.t. the whole feature set F.

Our idea is to select individual entities that are more anomalous w.r.t. F and construct collections from them. Naturally, an individual entity e is more anomalous if its **singular anomaly score**, i.e., $\Omega(\{e\},F)$ is larger. We hence take the heuristic that those collections whose elements have larger sum of singular anomaly scores would have larger anomaly scores. In other words, we approximate $\Omega(S,F)$ by $\Sigma_{e\in S}\Omega(\{e\},F)$. For example, in Figure 4.2, we have $\Omega(\{1\},F) = -\log p(1,1,1) - \log p(1,3,1)$=3.51 and $\Omega(\{2\},F)$=$-\log p(1,2,1) - \log p(1,4,1)$=2.53. This aligns with our intuition that e_1 is more anomalous than e_2. Note that this heuristic omits the collection level structure among entities and therefore does not guarantee exact results.

As we do in the exact algorithm design, we use π to denote the list of rankings indicated by the n pointers, and we apply the same scheme to put n pointers such that $E^f(\pi)$ has $p(i,r,n)$ as representative p-value. Then, for each pointer, we get the set of entities it can point to and rank the entities by their singular anomaly scores in descending order. Thus we have n candidate entity lists, each corresponding to a $\pi(j)$, $(1 \leq j \leq n)$. After computing $\Omega(\{e\},F)$ for each of the 10 entities, we arrive at the candidate entity lists (e_1,e_2,e_3), (e_2,e_3,e_6) and (e_8), corresponding to $\pi(1)$, $\pi(2)$ and $\pi(3)$ respectively. The lists are shown in Table 4.3 in columns along with their anomaly scores.

Next, we form a collection from the first entity on each candidate entity list, i.e., e_1, e_2 and e_8. As the representative p-value $\widehat{p}_{f_2}(\{e_1,e_2,e_8\})$ is $p(2,2,3)$ rather than

the desired $p(3,5,3)$, we discard this collection. In order to find the collection that has the next largest sum of singular anomaly score and $p(3,5,3)$ as representative p-value, we reuse the concept of "frontier" to keep the pool of collections, from which the next collection is selected.

Given any collection $E^f(\pi)$ and a p-value p, the **pointer frontier** of $E^f(\pi)$ on f is the set of collections, each generated by replacing one entity $E^f(\pi)$ with the next entity on the candidate entity list corresponding to $\pi(j)$ $(1 \leq j \leq n)$. Each collection is generated subject to the constraints: (I) $\pi(j) < \pi(j+1)$; (II) p is its representative p-value; and (III) it does not overlap with C. Next, we aggregate the collections selected from $|F|$ number of pointer frontiers by anomaly score.

4.4.2 Heuristically Finding Top CAC of Size n

Putting the ideas together, we have Algorithm 13 topCAC_size_H that heuristically computes the top CAC of a given size n. It needs two additional parameters θ^c and θ^p for applying heuristic 2. The first parameter θ^c is used for selecting collections for a given p-value. Specifically, for $p(i,r,n)$, we repeatedly pop a collection across multiple features and evaluate whether it is larger than the current top CAC. If the top CAC remains unchanged after θ^c number of times, we stop the collection selection process for $p(i,r,n)$. As the algorithm searches progressively larger p-values, the second parameter θ^p serves as a ceiling on the number of p-values that have contributed no collections to the current top CAC. The intuition is that if θ^p number of p-values have not contributed any collection to the top CAC, the unseen p-values which are even smaller are unlikely to be able to contribute collections. θ^c and θ^p can be decided empirically. Larger θ^c and θ^p settings imply going through more candidate collections, which necessitates a longer execution time. The setting of θ^c and θ^p will be studied in the experiments section.

Time Complexity of algorithm CACD_H. We first analyze the time complexity of Algorithm 13 topCAC_size_H. The for loop in step 6 is of $|F|$, assuming step 7 and 8 can be computed in constant time with proper data structures. In the previous chapter, we know that the time complexity of computing $\Omega(S,F)$ is $O(|S|^2 \cdot |F|)$. Thus, the time complexity of step 10 is $O(|S|^2 \cdot |F|^2)$. According to [CPMF04],

Algorithm 13 topCAC_size_H(E,F,n,C), *the heuristic algorithm to detect top CAC of size n*

Input: E, F, collection size n, constraint set C, θ^c and θ^p for applying heuristic 2
Output: the top CAC of size n: S

1: Let $pt \leftarrow (n,n,n)$, let $S \leftarrow \emptyset$ { pt keeps the current p-value (i,r,n) tuple}
2: Let $\Psi \leftarrow frontier\_p(pt)$ { $frontier\_p()$ computes the frontier of the $p(i,r,n)$ indicated by pt}
3: $y \leftarrow 0$
4: **repeat**
5: $x \leftarrow 0$
6: **for** each $f \in F$ **do**
7: Initialize $S(f)$ as the collection having the largest sum of singular anomaly score for feature f and disjoint with C.
8: $\Gamma(f) \leftarrow frontier\_i(S(f),C)$ {$\Gamma(f)$ keeps the pointer frontier for feature f; $frontier\_i(S(f),C)$ returns the pointer frontier of collection $S(f)$ and the collections are disjoint with C}
9: **while** $(x < \theta^c)$ AND $\exists f \in F$ s.t. $S(f) \neq \emptyset$ **do**
10: $f \leftarrow argmax_{f \in F} \Omega(S(f),F)$
11: **if** $\Omega(S(f),F) > \Omega(S,F)$ AND $S(f)$ is coherent **then**
12: $S \leftarrow S(f)$
13: **else**
14: x++
15: $S(f) \leftarrow pop(\Gamma(f))$ {pop out the collection with the largest anomaly score in $\Gamma(f)$}
16: $\Gamma(f) \leftarrow \Gamma(f) \cup frontier\_i(S(f),C)$
17: **if** S is never updated for this pt **then**
18: y++
19: $pt \leftarrow pop(\Psi)$ {returns the (i,r,n) tuple with the smallest p-value in the frontiers Ψ}
20: $\Psi \leftarrow \Psi \cup frontier\_p(pt)$
21: **until** $y > \theta^p$
22: **return** S

the time complexity of coherence checking is $O(|S|^2)$. Then, the while loop is $O(\theta^c \cdot (|F|^2+1) \cdot |S|^2)$. Therefore, the time complexity of Algorithm 13 is $O(\theta^p \cdot |F| + \theta^p \cdot \theta^c \cdot (|F|^2+1) \cdot |S|^2)$.

Algorithm 12 CACD_H calls Algorithm topCAC_size_H at step 7 and step 9. Instead of calling Algorithm topCAC_size_H $|E|/2$ times as CACD_E calls topCAC_size_E, CACD_H is assumed to call topCAC_size_H μ times, which is much smaller than $|E|/2$. Therefore, the time complexity of algorithm CACD_H is $O(K \cdot \mu \cdot (\theta^p \cdot |F| + \theta^p \cdot \theta^c \cdot (|F|^2+1) \cdot \mu^2))$.

4.5 Experiments on Synthetic Data.

In this section, we present experimental studies of our CAC detection on synthetic data. We are interested to know compared to existing baselines, how efficient our exact and heuristic algorithms are; and how well our proposed heuristics retrieve injected CACs with varied parameter settings.

4.5.1 Synthetic Data Generation

This synthetic data generation algorithm is based on Algorithm 7 of Chapter 3. Besides the population size, the number of features, we assume that for any injected CAC, its lower-bound on the minimum extremity index across all significant features is given in order to control the anomalousness of the injected CACs. The output is the set of entity lists $\{EL_f\}$ and the set of injected CACs.

We start from randomized entity lists and inject multiple CACs of various sizes on different features until the entity lists are no longer independent of each other. After this step, in addition to Algorithm 7, we need to estimate the upper bound x on the size of CAC, such that any two features that with this CAC appearing at the top extreme positions are still independent. We do not need to add the condition of $|\mathbb{S}| < n_s$ of step 3, as we do not directly control the number of injected CACs by an input parameter. Entering the while loop, we add a step of "Randomly select n, $n < x$ to be the size for an injected CAC.". Lastly, in step 6, we not only need the randomly generated extreme index $r_f < z$, but also need $min_{f \in F'} r_f < max(n, r^*)$. The rest of generation algorithm is identical to Algorithm 7.

4.5.2 Efficiency of CAC Detection Algorithms

To evaluate the efficiency of our proposed algorithms, we design two baseline approaches by modifying the exact algorithm ERACD_E and the heuristic algorithm ERACD_H for detecting ERACs proposed in [DZLP12b]. Both algorithms search collections in a bottom-up fashion with some pruning strategies for the most anomalous collections of all sizes. We add the coherence checking part to the two algo-

rithms and use each of the two algorithms to compute the top CAC and consecutively find the next most anomalous CACs that do not overlap with any of the previously detected CACs. We thus have a competing exact algorithm that returns the exact top-K CACs denoted as ERAC_E_CH and another heuristic one denoted as ERAC_H_CH. Altogether, we have four methods to compare.

Since both exact algorithms CACD_E and ERAC_E_CH take too long to run on large data, we first generate synthetic data with relatively small population sizes. We set $|F|$ =10, r^*=10 and vary the population size $|E|$. This makes sure that some really anomalous CACs are present in the datasets so that ERAC_E_CH and ERAC_H_CH would prune more data points and run faster. For each population size, we generate 3 synthetic datasets and run all three methods on the generated data. Regarding the parameters of CACD_H, we set θ^c=100 and θ^p=20. For all methods, K is set to the number of injected CACs. Note that K may be different for different generated datasets, depending on when the stopping criterion at step 3 of Algorithm 7 is met.

The average execution time of the four methods(in seconds) are shown in log10 scale in Figure 4.3(a). We see that in general when the population size grows, each method needs more time to execute. Both heuristic algorithms are fast, whereas the exact algorithms run much slower than the heuristic ones and their running time grows exponentially with population size as expected. Nevertheless, CACD_E runs faster than ERAC_E_CH. All methods are able to retrieve the injected CACs, if they can terminate within 24 hours.

We also create datasets with larger populations to test the limitation of ERAC_H_CH. Other parameters of the synthetic data generation algorithm are set as before. The average running times are plotted in Figure 4.3(b). We observe that CACD_H runs much faster than ERAC_H_CH and stays on the scale of hundreds of seconds when size goes up. However, when the population size reaches 500, ERAC_H_CH takes more than 24 hours to output the result. Since CACD_H is the most efficiently over large datasets, we use it to detect CACs in the remaining experiments.

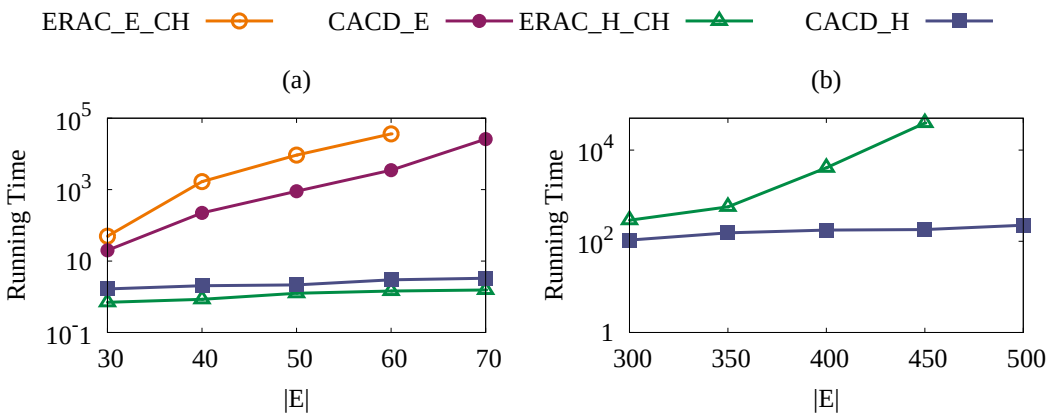

Figure 4.3: Comparison of the efficiency. The running time is shown in seconds on log10 scale.

4.5.3 Effectiveness of CACD_H

To evaluate the accuracy of CACD_H on detecting CACs, we need some baseline method for comparison. As our method can simultaneously detect anomaly collections and their corresponding significant features, one may suspect that similar results can be obtained by modeling the problem as a co-clustering task by clustering rows (features) and columns (users) of a matrix at the same time. We thus compare with the results of a co-clustering algorithm from [CPMF04]. This algorithm is chosen as it does not need the number of clusters as input and is denoted as Co-clustering.

Co-clustering takes a matrix containing 1s and 0s as input. To detect anomaly collections, a cell value of the input matrix should encode whether an entity appears within the extreme positions of a given feature. We thus take the entities that appear within the extremity index of any injected CAC as the target entity set. To generate one cell value (f,e) of the input matrix, where $f \in F$ and e belongs to the target entity set, we set (f,e)=1, if entity e appears within the extremity index of any injected CAC w.r.t. feature f. We set (f,e)=0, otherwise. Note that in this way, Co-clustering is already fed with partial information on the ground truth, which CACD_H does not have.

We set $|E|$=5000, $|F|$ =10 and vary r^*. For each parameter setting, we generate 3 datasets and for both methods we measure precision@K, where K is the number of

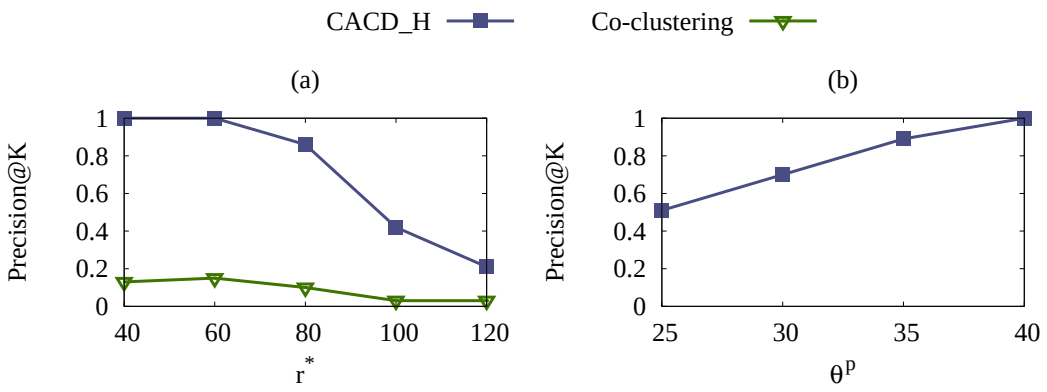

Figure 4.4: (a) Precision@K of CACD_H and Co-clustering; (b) Precision@K of CACD_H with r^*=120.

injected CACs. Regarding other parameters of CACD_H, we set θ^c=100 and θ^p=20. The results are plotted in Figure 4.4(a).

As we see in the figure, CACD_H achieves much higher precision than Co-clustering despite Co-clustering knowing partial information about the ground truth. As Co-clustering considers each feature equally important when optimizing the clustering function, it may not successfully distinguish different anomalous collections appearing at the extreme positions w.r.t. different features. In contrast, CACD_H is able to identify different significant features for different collections.

Another observation is that when r^* increases, the precision of CACD_H drops. This is expected as CACD_H samples candidate collections from small to large representative p-values w.r.t. single features. With a fixed θ^p=20, CACD_H can only search a fixed number of p-values. If the p-values being searched is smaller than the smallest representative p-value of an injected CAC, then CACD_H will miss this CAC. As r^* increases, the smallest representative p-value of each injected CAC may increase, making it less likely for CACD_H with θ^p=20 to successfully sample the injected CACs.

In order to achieve higher precision when r^* is large, we increase θ^p to sample more candidates with larger representative p-values. Figure 4.4(b) shows that with r^*=120 and θ^c=100, increasing θ^p leads to better precisions.

We also vary θ^c from small to large while fixing θ^p=40 and r^*=120. Figure 4.5 shows that as θ^c increases, precision also increases as expected. Once θ^c is set

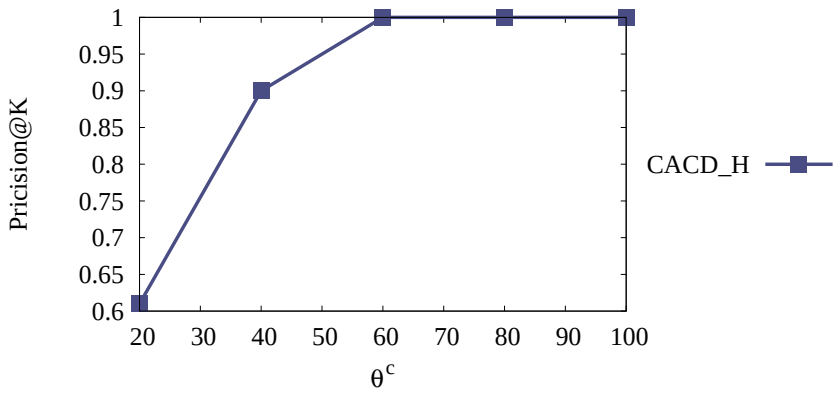

Figure 4.5: Precision@K of CACD_H, varying θ^c.

to greater than 60, the precisions are stable at 1.0, suggesting the collections we sampled corresponding to each p-value w.r.t. single features are very anomalous w.r.t. the set of all features.

4.6 Experiments on Twitter Data

4.6.1 Data Setting

Our Twitter data[2] is composed of all the tweets published between September 8th 2011 and November 15th 2011, containing any of the 9 hashtags related to Singapore including #sg, #singapore and #sosingaporean. Altogether, there are 231,803 tweets from 21,666 users. With a total of 11,901 hashtags, a hashtag is used by 4.58 users on average. Ranked by the number of users, the top 5 percent of these hashtags are considered popular. We remove the rest of the hashtags along with the ones that we used to collect the data. We also remove all the retweets, as hashtags in retweets do not indicate that the retweeting user is spamming on the hashtags. In addition, we filter away users who use a hashtag only once, since they are unlikely to be spamming on hashtags. After the preprocessing, we are left with 1899 users with 587 popular hashtags, which are considered independent features. For each hashtag, we rank all the users in descending order by usage frequency. If a user never mentions a particular hashtag, the corresponding feature value is zero. For

[2]http://research.larc.smu.edu.sg/palanteer/index_tracker.php

$K(\Omega)$	CAC (size) — [Spamming Behavior]
1(272.09)	{LiveSEXsheLOve, LOVEsexyFREE, PornLOveCamFree, SexFullFreeCam, SEXsheylaPOrn, SHEYLLAsexPORN, LOVEsexCamFRee4} (7) — [Each posted $\geq$ 20 duplicate tweets with identical short URLs pointing to a pornographic website in Brazil – palcoprincipal.com.br]
2 (265.06)	{SexFreeLive1, ChinaSexPOrnFRE, SEXYloveCAMFree, TokioSEXfreeLiv, FullSEXpornFREE } (5) — [Same as # 1]
3 (262.19)	{SEXyFreCAMLOVe, SEXdownloadFRE2, DownloadSexFree, LOVEsexCAMfree2, LOVEsexCAMlive, SEX_FREEpornTOP, SexPornFreeLive } (7) — [Same as # 1]
4 (214.45)	{Lostgetfound, hmseetho, ChanHeng71, russan10, loveFannyWong, shao29, jonathan_ben, AllisonZoe1, JeanetteBatisah, willow_11, cheokjung, Amabelle_Kok, VictoriaKaylee, Finding4Job, MedhatSaaleh, FionaWee2, PositiveInspira} (17) — [Identical tweets with the same URL pointing to the same job-hunting related website – dougleschan.com]
5 (214.05)	{TopSEXFreeLIve, SEXpornFREEcam, LoveSexCamFree3, LOVEpornSEXfree} (4) — [Same as # 1]
6 (193.41)	{LOVEsexFREE, LoveSexyFreeRio, SexyCamRioFree, SexLoveDownload, SEXdownloadCAM} (5) — [Same as # 1]
7 (189.99)	{SEXfreeCAMteen, NIkitaSHOWsexy, LoveSexFreeCam, iLoveHKSAR, Rock_in_Rio2012} (5) — [Same as # 1]
8 (178.05)	{blackberrypros, randomwireless, greenerblogs} (3) — [Identical tweets followed by the same URL pointing to webpages promoting travel guidance publications, e.g. stumbleupon.com/su/3P4etP/livetravelcity.com/2011/09/little-city-travelers/]
9 (141.15)	{HirenDarlami_EI, HumanRevolutioo} (2) — [All their tweets contains messages "Occupy MANIFESTO FEARLESS HUMAN- ISM" followed by a URL pointing to a website "www.occupybucharest.com"]
10 (124.19)	{DevanLum, LJeanneP, jasminechua1223, ChuaYeeCheow, lydiasoh1221, JoshuaMason6, Jade_Dinnadge, jeanchloethoo} (8) — [Same as #4]

Table 4.4: Top-10 disjoint CACs in Twitter data.

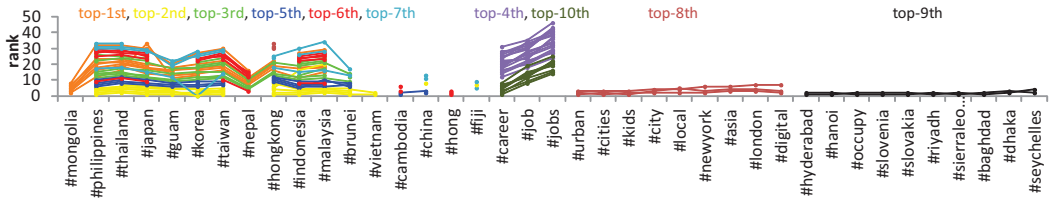

Figure 4.6: Top-10 CACs corresponding 63 users and 39 significant features (i.e., hashtags).

each feature, rankings of users with identical feature values are randomized.

4.6.2 Effectiveness of CACD_H

In this experiment, we empirically set θ^c=100 and θ^p=20 for our algorithm CACD_H. Larger parameter values such as θ^c=500 and θ^p=50 have also been tried and give the same results with longer running time. We set K to a large value, so that CACD_H stops when it finds all the disjoint CACs.

As a result, CACD_H produces 36 disjoint CACs. Table 4.4 shows the top 10 CACs with their identified spamming behavior. It is clear that members of a CAC collaborate in the same spamming campaign, as their tweets are often identical, with no real content other than a large number of hashtags appended with short URLs pointing to some website(s).

All members in the top-10 CACs are visualized in Figure 4.6 by parallel coordinates. Each member is represented by a line connecting the ranks of the user's usage of all the 39 significant features (i.e., hashtags). Members of the same CAC are given the same color. It is visually telling that all the 10 CACs are both extreme and coherent in their usage patterns of the hashtags. Moreover, our algorithm can identify subtle differences in the extreme behavior of CACs which seemingly belong to the same group. For example, the first and second most anomalous collections of "pornographic" spammers are in fact slightly different in their spamming patterns: (I) Besides the 8 hashtags in common, the top group spams on #mongolia and #nepal while the top-2nd group spams on #brunei and #vietnam; (II) the second

top CAC, despite having fewer members, uses most of the hashtags more heavily than the top CAC.

We again compare the effectiveness of our algorithm to that of the co-clustering approach. To apply the co-clustering algorithm, we need to first derive the input matrix. The direct way of representing the input matrix on this Twitter data is to give a value of 1 to a cell if the corresponding user has used this hashtag, and give a value of 0 otherwise. Consequently, we have a 587 (number of features) by 1899 (number of users) matrix to feed into the co-clustering algorithm [CPMF04]. In the result, the matrix is co-clustered into 8 feature groups and 10 user groups, the largest user group being of size 467 and the smallest of size 2. Of all the 10 user groups, none are coherent and only 5 of them can be considered to be anomaly collections. We manually go through the 5 anomaly collections and find that they are anomalous only because they contain subsets of members that are ranked at extreme positions on a small number of features. This is not surprising as co-clustering aims to group users using similar sets of hashtags, not necessarily those who *heavily* use these hashtags. While identified behavior are shared, they are not necessarily anomalous.

Even if we take only the union of users ranked in the top positions of each feature, the co-clustering algorithm would not output some extremely ranked collections as expected. We choose the top-31 positions of each feature so that the input matrix contains information on all the users in our top-10 CACs. Yet, out of the 10 user collections identified by co-clustering, none are both anomalous and coherent. The most anomalous collection returned has a size of 124; it contains some of the pornographic spammers, and many other users that are not even sharing the same significant features with the pornographic spammers. This observation is in line with what we observed in the synthetic data above. The poor performance of co-clustering is due to its treating every feature the same when trying to simultaneously group users and features. In contrast, our approach is able to identify the significant features along with the anomalous users.

4.7 Experiments on Web Spam Data

4.7.1 Data Setting

We extract the web host graph from WEBSPAM-UK2006[3] published by Yahoo! Research by iteratively removing nodes with less than 2 incoming neighbors, assuming they are not spammers. This gives one big connected web host graph, with 5634 nodes (including 1709 spammers). We extract 102 features including the 96 content features identified in [CDG+07] [BCD+08] and the 6 structural features at the host level, including the number of 1-hop and 2-hop incoming neighbors. The features of a host are represented by its home page as well as the page with the highest PageRank score on the host.

As reported in [BCD+08], the incoming neighborhoods of spammers are extremely homogeneous or heterogeneous compared to those of normal ones that are gradually built up. We therefore derive neighborhood features that capture the homogeneity of a host's 1-hop neighborhood for each of the content and structure features. Ranked by the neighborhood features, web spammers are expected to appear in groups at the the top or bottom positions. We hence use neighborhood features to detect web spammer groups. As the features adopted may be dependent, we adopt the algorithm in [Epp05] to extract the independent feature set that produces the most anomalous top CAC among all independent feature sets. The resultant feature set is {"Number of words", "Top 100 corpus precision", "Independent LH" }, where corpus precision refers to the fraction of words that appear in the set of popular terms, and Independent LH is a measure of the independence of the distribution of trigrams. We use this feature set for the rest of the experiments.

4.7.2 Effectiveness of CACD_H

We apply our heuristic algorithm CACD_H with θ^c=100 and θ^p=20. We again set K to a large value in order to get all disjoint CACs. The competing methods are four

[3] http://barcelona.research.yahoo.net/webspam/datasets

representative algorithms for mining individual anomalies or spammers: (I)The two variants of unsupervised TrustRank [GBGMP06] designed for spam detection. We denote the variant involving home pages as TR_hp and the variant involving the pages with highest PageRank score as TR_mp. (II)The supervised decision tree DT techniques employed in [CDG$^+$07] and [BCD$^+$08]. (III)The density-based approach LOF [BKNS00] for individual anomaly detection.

Since the ground truth is labeled on individual hosts, we assume that the hosts in the top-K CACs are spammers, and compare with the same number of hosts ranked at the top by the other approaches. For example, our top CAC contains 26 hosts (including 25 true spammers) with a precision of 0.96. We take the top 26 hosts returned by each competing approach and measure their precisions.

Figure 4.7 shows the precision curves of the five algorithms. Since our algorithm returns altogether 30 CACs, we let K range from 1 to 30. It can be observed that our approach outperforms the other methods for $1 \leq K \leq 13$. As K increases, the CACs found become less anomalous and hence the lower precision. When $K = 13$, CACD_H identified 195 hosts (including 141 true spammers) with a recall of 0.08. When $K = 30$, we are able to reach a recall of 0.12. Note that our aim is to detect hosts sharing extreme group behavior, not spammers of all types with or without extreme group behavior. For $1 \leq K \leq 30$, all the pairwise overlap ratios are below 0.07 between the hosts in our top-K CACs and the same number of hosts returned by each competing approach. This suggests that CACD_H is able to capture spammer hosts with anomalous group behavior that are missed by the competing methods.

4.8 Experiments on Chinese Online Forum Data

We demonstrate the effectiveness of our framework by detecting the infamous Chinese online "water army" (网络水军) collections. Spammers solicits tasks from internet public relations agencies such as shuijunwang.com. Since these spammers are paid according to the quality and the quantity of posts or comments generated, each of them often possess multiple user accounts. To avoid being identified by the forum as spammers that leads to a cease of the user account, spammers would

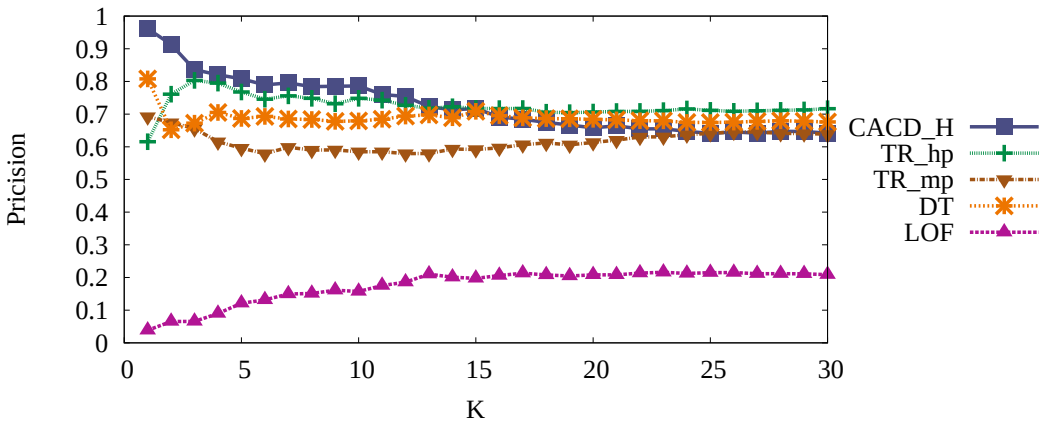

Figure 4.7: Precision curves against K.

constantly switching from their accounts to hide his/her true identities. Moreover, spammers are instructed by agencies not to perform lazy detectable behaviors such as copy and paste. Instead spammers should try their best to post relevant and meaningful messages. This makes detecting water army spammers even harder, as they may disguise themselves just as any other normal users in terms of messages posted.

Nevertheless, our method can be applied to detect these kind of water army spammers by capturing their collective and coherent posting behavior. We consider one thread regarding a focused event (e.g. QQ v.s. 360 dispute) as a feature, and the number of times a user comments in this thread as the feature value. When users are ranked according to each thread in descending order, we expect that collections of water army members would appear in the top positions of multiple threads, due to the fact that many water army spammers may collaborate, or the same water army spammer may post with multiple accounts.

Note that a normal active user may also appear in the top position of few threads, as he or she may get involved with some arguments with some other users and they ended up commenting a lot on these few threads. However, normal users would not have the time and energy to comment intensively on multiple threads. Even if they do, they are less likely to be involved with the same group of users.

4.8.1 Data Setting

We choose Tianya.cn[4], one of the most popular online Chinese forums, to test our approaches. We pick the 360 v.s. QQ dispute as the target event. The data was crawled in Oct 2011 by first submitting the event name as keywords to the search engine of Tianya.cn, which by default returns 750 threads. We keep those threads that are posted within the time period when the 360 v.s. QQ dispute was on-going, starting from Sept 2010 to Dec 2010.

With the relevant threads of the target event, we extract all users that have posted or commented in these threads. We filter users by their membership scores. The membership score assigned by Tianya.cn reflects how active a user is and how much contribution he has given including writing acknowledged posts and getting involved in different kinds of interactive activities. Since a water army spammer normally has many dummy user accounts and would not take too much effort in building up membership scores for each of them, we remove the top 50 percent of user accounts having high membership scores. We also filter out users with only one post or comment, as they are also not likely to involve in water army either. In the end, we have altogether 413 threads and 3545 users and all users are ranked in descending order by feature value on each thread.

4.8.2 Effectiveness of CACD_H

We set θ^c=100 and θ^p=20 for our heuristic algorithm as before, which returns 101 disjoint CACs. We also tried θ^c=500 and θ^p=50, which gives identical results. We list the top-10 CACs along with their anomaly score and significant features in Table 4.5.

To better evaluate our results, we gather all information available in each user's homepage in Tianya.cn, which contains the membership score, number of visits, last visit time, registration time and all the threads this user has commented. As the homepage simply lists all threads of a user without the actual comments made

[4]http://www.tianya.cn

K	CAC	Ω	$\lvert F^S \rvert$
1	{luanqioere, huisiooo, tuanduer, kankse, sasahie}	286.98	15
2	{hongdou66, sihaiwei, tinghohoo}	272.08	19
3	{存在定义, 再见能否, yingema, 抓紧了想念}	266.66	16
4	{qiangdong8, qiwanci0, 油面小豆丁}	239.24	18
5	{山 里 的 树9, 珠 珠 雨 粒, treegreen12010, yuan9polin}	229.07	15
6	{天高路通, 透_明_天_空, 湃大性, 热带果粒001, 在不小里, 轻轻飘离, 大红气球}	197.78	11
7	{yanyizie, 三千三百, taiyaniii}	190.25	14
8	{无奈岁月8, qanliepo, binmalim, duanduanee, quanyeke}	184.80	13
9	{wenzihe, 河里鱼花, manrahe}	179.46	14
10	{douahaoe88, banbane90, niakdker}	177.46	13

Table 4.5: Top-10 disjoint CACs in Tianya data

this user, it is very hard to navigate through all the threads to find evidence. We therefore crawl for each user account, all information listed on its home page and the threads together with the comments this account involves, so that we can better observe their behaviors.

After putting all information of the 5 members in the top CAC together, we discover that all 5 members have commented altogether 100 threads. Most of these threads are related to 360 v.s. QQ dispute, out of which our algorithm shows that they heavily comment on the same 15 threads. Moreover, the comments they give in all these threads are consistently talking about bad things about 360 or good things about QQ. Interestingly, besides the 15 threads about the dispute, they all comment on threads including "妞儿,看见没,这才叫过日子", which is clearly promoting the 129T.com website, and "宝妈妈注意了,蓝瓶要升值了,赶紧攒吧!", which is promoting some health care product. We can also see that they almost never repeat their comments both within the threads and across threads, showing they put a lot of effort in commenting in order to be considered as normal by other users that reading these threads. All of the above observations indicate that these 5 users are collaborating in opinion spamming, which cannot be easily identified just by reading their comments individually.

The second most anomalous CAC contains 3 users. They all post on 19 threads with good words about QQ and bad words towards 360. Besides these threads,

they all comment on multiple threads promoting a cosmetic product of the brand "雅顿". Other than threads regarding cosmetic product and QQ v.s. 360, they seldom comment on other threads. We also observe that they seldom repeat their own comments and different members post different comments, which shows they put more efforts than simply copy and paste existing comments. Interestingly, all three users are registered on the same day, 2010-02-06, which is another strong indication of being involved in collaborative spamming activities.

After going though all top-10 CACs, we find that they are all condemning 360. It seems that the water army collections are in favor of QQ on Tianya.cn.

4.9 Summary

In this chapter, we propose the problem of detecting top-K disjoint Coherent Anomaly Collections (CAC). We present both exact and heuristic algorithms to identify CACs that does not need the number of collections or collection size to be specified beforehand. The efficiency and effectiveness of the proposed algorithms are demonstrated by comparing against existing approaches on synthetic data. Our heuristic algorithm is also tested on three real-life datasets, namely Twitter.com, a Webspam dataset and a Chinese online forum, to detect hashtag spammer collections, spamming website collections and opinion spammer collections respectively. The experiment results demonstrate that in all three datasets, our approach successfully finds suspicious spammer groups which are not easily identified with other approaches.

Chapter 5

Detecting Anomalies in Graphs

5.1 Introduction

In the previous two chapters, we exploit the anomalous behavior of unusual coalition among a collection of entities to detect anomaly collections. Here in this chapter, we are interested in another type of anomalous behavior: expressing unusual conflicting views among human actors towards the same resource. To better study this type of anomalous behavior in social data, we model different types of entities including human actors and resources as nodes of different partites in bipartite graphs. In these bipartite graphs, a directed edge carries the "opinion" of a source node towards a target node. For example, in the users-rating-products context in online marketplaces, an edge conveys the rating given to a product by a user. in the users-clicking-webpages context on the World Wide Web, an edge conveys the degree of endorsement from a user to a webpage.

Moreover, from the perspective of a target node, an edge can be identified as **agreeing** or **disagreeing** by whether the opinion carried by this edge agrees with the majority opinion on the target node. For example in Figure 1.1, we show a toy example with 5 users (represented by s_1 to s_5) and 3 products (represented by t_1 to t_3) with the edges carrying ratings on a scale of 5. We observe that edge (s_4, t_2) and edge (s_2, t_3) do not agree with the majority opinion on target nodes t_2 and t_3 respectively. We denote the two disagreeing edges with dotted lines and the agreeing edges with solid lines.

In general, in a bipartite graph, anomalous nodes are the minority and are inconsistent with the rest of the nodes in the same partite. From this toy example, we first try to manually identify node anomalies that are inconsistent with the rest in terms of opinions. We observe from Figure 1.1 that users s_1, s_2 and s_3 hold the same opinion toward products t_1 and t_2, on the other hand, s_4 and s_5 agree on t_3. However, disagreeing edges (s_4,t_2) and (s_2,t_3) show that $\{s_1,s_2,s_3\}$ and $\{s_4,s_5\}$ hold different opinions on products. Since the majority of the source nodes should be normal, users in $\{s_1,s_2,s_3\}$ are considered to be normal. As a result, products t_1 and t_2 that are agreed upon by the normal users are unspammed products, whereas the product t_3 agreed by anomalous users are considered to be anomalous. A possible scenario is: $\{s_1,s_2,s_3\}$ are normal users who like products t_1 and t_2. Also s_2 thinks that product t_3 is very good. However, products t_2 and t_3 that are given high ratings by normal users are given very low ratings by s_4 and s_5, who are possibly two spammers hired to demote t_2 and t_3.

In reality, we are not likely to be able to observe a clear split between opinion groups. Rather we need to derive "local" principles for identifying anomalous nodes in both partites. One observation is that we cannot judge a node by its edges alone, instead we should also involve the linked nodes in the other partite. For example, we cannot say s_2 is anomalous simply because he gives t_3 a minority rating. In fact, it is natural for s_2, a normal node, to give a minority rating to t_3, which is mostly demoted by spammers. In contrast, s_4 giving a minority rating to t_2 should be identified as anomalous. This is because t_2 is an unspammed product and thus any user who disagrees with the majority who gave high ratings is suspicious. Similarly, we cannot say s_5 is normal simply because he gives t_3 a majority rating, as the fact that t_3 is anomalous makes s_5 giving an agreeing edge to t_3 also anomalous.

Thus, an agreeing edge plays a **positive mutual dependency** role on its source and target. For example, s_1 is more normal if t_1 is more normal and vice versa; similarly, s_5 is more anomalous if t_3 is more anomalous and vice versa. It can also be observed that an disagreeing edge acts as a **negative mutual dependency** channel on its ends. For example, s_2 is more normal if t_3 is more anomalous and vice versa.

We emphasize that both positive and negative mutual dependencies are important. For example, s_5 would not be marked as anonymous without the positive mutual dependency on the anomalous t_3. Further, t_3 is flagged as anomalous only due to the negative mutual dependency on s_2, a normal user.

We therefore arrive at the following integral set of **mutual dependency principles**: (I)**Positive mutual dependency** states that a source is more anomalous if it connects to anomalous targets with agreeing edges, and more normal if it connects to normal targets with agreeing edges. (II)**Negative mutual dependency** states that a source is more anomalous if it connects to normal targets with disagreeing edges, and more normal if it connects to anomalous targets with disagreeing edges. Although the principles are stated in terms of judging source nodes, equivalent principles apply on target nodes.

Previous studies have proposed to model only one of the principles. The mutual reinforcement principle is utilized to rank webpages [Kle99], to identify salient terms and sentences [Zha02], to detect reliable users and contents in social media [BLZ+09] and to find suspicious reviewers [WXLY11]. The negative mutual dependency principle is used to detect biased users and controversial products in evaluation systems [LLW08]. We are the first to propose a generic anomaly detection framework that integrates both sets of mutual dependency principles.

The rest of the chapter is organized as follows. We formulate our problem and present our model in Section 5.2, and Section 5.3, Section 5.4 and Section 5.5 cover experiments and their results. We conclude the chapter in Section 5.6.

This chapter is based on our publication in IEEE International Conference on Data Mining (ICDM 2012) [DZLP12a].

5.2 Anomaly Detection Framework

In this section, we first define the anomaly detection problem, followed by model formulation. We then discuss the iterative computation for solving the problem and its convergence. The frequently used notations are summarized in Table 5.2.

Notation	Meaning	Notation	Meaning
G	the bipartite graph	S	nodes in source partite
T	nodes in target partite	E	directed edges from source to target nodes
A	edge labels		

Table 5.1: Notations

principles	source	edge	target
positive mutual dependency	normal	agreeing	normal
	anomalous	agreeing	anomalous
negative mutual dependency	normal	disagreeing	anomalous
	anomalous	disagreeing	normal

Table 5.2: Mutual dependency principles.

5.2.1 Problem Definition

Given a bipartite graph $G = \langle S \cup T, E, A \rangle$, where $S = \{s_1, ..., s_{|S|}\}$ is a set of nodes in the source partite, $T = \{t_1, ..., t_{|T|}\}$ is a set of nodes in the target partite, $E \subset S \times T$ is a set of directed edges from the source partite to the target partite, each directed edge carries an edge label, an edge can be identified as agreeing or disagreeing by whether its label agrees with the majority edge labels towards the target node, and $A = \{a_{ij}\}$ is a set of labels attached to edges, such that

$$a_{ij} = \begin{cases} 0, & \text{if } (s_i, t_j) \text{ is an agreeing edge;} \\ 1, & \text{if } (s_i, t_j) \text{ is a disagreeing edge.} \end{cases}$$

The **anomaly detection problem** is to assign an anomaly score to each node in each partite. The **anomaly score** of a node is a value in $[0,1]$, with $[0,0.5)$ being the normal range and $(0.5,1]$ being the anomalous range. In particular, 0 indicating absolute normality and 1 indicating absolute abnormality.

5.2.2 Model Formulation

Our model is formulated based on the aforementioned principles summarized in Table 5.2.

We first show how to compute the anomaly score $\mathfrak{s}_i$ of source node s_i from an edge label a_{ij} and the corresponding anomaly score $\mathfrak{t}_j$ of target node t_j. As we observe from Table 5.2, our mutual dependency principles can be illustrated as:

(I)When $a_{ij} = 0$, i.e., for agreeing edges, $\mathfrak{s}_i$ mirrors $\mathfrak{t}_j$, i.e., $\mathfrak{s}_i = \mathfrak{t}_j$; (II)When $a_{ij} = 1$, i.e., for disagreeing edges, $\mathfrak{s}_i$ is the opposite of $\mathfrak{t}_j$, i.e., $\mathfrak{s}_i = 1 - \mathfrak{t}_j$.

Note that since $\mathfrak{s}_i$ and $\mathfrak{t}_j$ are mutually dependent of each other, Table 5.2 only shows two extreme cases of this mutual dependency relationship with nodes associated with anomaly score of integers 0 and 1 respectively. In order to quantify the mutual dependency in general cases, we assume that the anomaly scores $\mathfrak{s}_i$ and $\mathfrak{t}_j$ change linearly with $\mathfrak{s}_i = \mathfrak{t}_j$ being positive dependency and $\mathfrak{s}_i = 1 - \mathfrak{t}_j$ being negative dependency. For example, if a source node s_i is of anomaly score 0.8, and the corresponding edge is agreeing, then it endorses its connected target nodes with an anomaly score of 0.8.

The two conditions together give rise to:

$$\mathfrak{s}_i = \begin{cases} \mathfrak{t}_j, & a_{ij}=0; \\ 1 - \mathfrak{t}_j, & a_{ij}=1. \end{cases}$$

It is easy to see that the above formula can be simplified as:
$\mathfrak{s}_i = (1 - 2a_{ij})\mathfrak{t}_j + a_{ij}$.

The above formula is for source partite, when it comes to compute the anomaly score $\mathfrak{t}_j$ from $\mathfrak{s}_i$ and a_{ij}, an equivalent formula applies.

$$\mathfrak{t}_j = \begin{cases} \mathfrak{s}_i, & a_{ij}=0; \\ 1 - \mathfrak{s}_i, & a_{ij}=1. \end{cases}$$

We hence have $\mathfrak{t}_j = (1 - 2a_{ij})\mathfrak{s}_i + a_{ij}$, as source and target are symmetric in our principles.

Next, we define the anomaly score of a node s_i or t_j as the aggregated anomaly score of all its linked target or source nodes. As we want to account for the impact of all the connected nodes, in the absence of information on the relative importance of various connected nodes, the reasonable option is to give them equal weights. The simple average achieves this, while keeping the score within [0,1]. Furthermore, average allows nice matrix transformation of our formula. Hence we have the following formula:

$$\begin{cases} \mathfrak{s}_i = AVG_{t_j:(s_i,t_j)\in E}(1 - 2a_{ij})\mathfrak{t}_j + a_{ij} \\ \mathfrak{t}_j = AVG_{s_i:(s_i,t_j)\in E}(1 - 2a_{ij})\mathfrak{s}_i + a_{ij} \end{cases} \tag{5.1}$$

5.2.3 Iterative Computation

We can now design the iterative process to compute the anomaly scores of sources and targets by translating Formula 5.1 into a matrix form. This way we can simplify computations by using existing matrix computation techniques.

Let $W^S = [w^S_{ij}]$ be a $|S|$ by $|T|$ matrix s.t.

$$w^S_{ij} = \begin{cases} \frac{1}{out\_degree\_of\_s_i}, & \text{if } (s_i, t_j) \in E; \\ 0, & \text{otherwise.} \end{cases}$$

Similarly, Let $W^T = [w^T_{ji}]$ be a $|T|$ by $|S|$ matrix, s.t.,

$$w^T_{ji} = \begin{cases} \frac{1}{in\_degree\_of\_t_j}, & \text{if } (s_i, t_j) \in E; \\ 0, & \text{otherwise.} \end{cases}$$

We then define $X = [x_{ij}]$ as a $|S|$ by $|T|$ matrix s.t.

$$x_{ij} = \begin{cases} a_{ij} w^S_{ij}, & \text{if } (s_i, t_j) \in E; \\ 0, & \text{otherwise.} \end{cases}$$

$Y = [y_{ji}]$ as a $|T|$ by $|S|$ matrix s.t.

$$y_{ji} = \begin{cases} a_{ij} w^T_{ji}, & \text{if } (s_i, t_j) \in E; \\ 0, & \text{otherwise.} \end{cases}$$

Let $\overline{X}$ be a vector with $|S|$ rows s.t. $\overline{x_i} = \sum_{m=1}^{|T|} x_{im}$ and $\overline{Y}$ be a vector with $|T|$ rows s.t. $\overline{y_j} = \sum_{m=1}^{|S|} y_{jm}$.

Let $\mathfrak{S}$ and $\mathfrak{T}$ denote the vectors of anomaly scores of sources and anomaly scores of targets respectively, Formula 5.1 can be translated to the matrix form as follows:

$$\begin{cases} \mathfrak{S} = (W^S - 2X)\mathfrak{T} + \overline{X} \\ \mathfrak{T} = (W^T - 2Y)\mathfrak{S} + \overline{Y} \end{cases}$$

$\mathfrak{S}$ and $\mathfrak{T}$ are computed iteratively. Let $\mathfrak{S}^k$ and $\mathfrak{T}^k$ denote the vectors of anomaly scores of sources $\mathfrak{s}_i$ and anomaly scores of targets $\mathfrak{t}_j$ in iteration k respectively, then we have the following iterative formula to compute the anomaly scores of source and target nodes.

$$\begin{cases} \mathfrak{S}^k = (W^S - 2X)(W^T - 2Y)\mathfrak{S}^{k-1} + (W^S - 2X)\overline{Y} + \overline{X} \\ \mathfrak{T}^k = (W^T - 2Y)(W^S - 2X)\mathfrak{T}^{k-1} + (W^T - 2Y)\overline{X} + \overline{Y} \end{cases} \tag{5.2}$$

5.2.4 Convergence

The formula of HITS [Kle99] based on the mutual reinforcement principle can be represented as the eigenvector equation and the iteration computation of hub and authority scores will converge to the principle eigenvector of A^TA and AA^T respectively, where A is the adjacency matrix of the graph. Other existing work including [WLLH08] transforms their corresponding matrix to a stochastic and irreducible one by column normalization in order to guarantee convergence. In [LLW08], although their formulation is not in the form of eigenvector equation, they translate their formula into an eigenvector equation by assuming the sum of the anomaly scores of all source or target is 1. To guarantee convergence, they normalize the scores during the iterative computation.

However, in our case, we do not make the assumption about the sum of anomaly scores. Nor can we normalize the corresponding matrix as done in the previous studies. This is because, since any score above 0.5 is considered as anomalous and any score below 0.5 implies normality, normalization may convert a node's score above 0.5 to a score below 0.5, which leads to a "wrong" perception about this node in the iteration process.

In this section, we prove that under certain assumptions, the ranking of nodes in each partite stays unchanged after a certain number of iterations. We only show the proof for source nodes here. The proof for target nodes is similar, as source and target are symmetric in our model.

Let $Q = (W^S - 2X)(W^T - 2Y)$ and $b = (W^S - 2X)\overline{Y} + \overline{X}$. Here, Q is a $|S|$ by $|S|$ matrix, b is a vector with $|S|$ rows and $\mathfrak{S}^k$ is the score vector with $|S|$ rows. Thus, we have $\mathfrak{S}^k = Q\mathfrak{S}^{k-1} + b$.

Lemma 8. *For $\mathfrak{S}^k = Q\mathfrak{S}^{k-1} + b$, where $k \geq 1$, if the initial value $\mathfrak{S}^0 = (0.5, 0.5, ..., 0.5)'$, $\mathfrak{S}^k$ always has the solution of $(0.5, 0.5, ..., 0.5)'$, for any $k > 1$.*

This lemma is easily proven. According to our formulation, if the anomaly scores of all targets are 0.5, then the anomaly scores of all sources are also 0.5, regardless of the edge label. Therefore, if the initial anomaly score vector for source is $(0.5, 0.5, ..., 0.5)'$, all scores of source nodes stay at 0.5 for any number of iterations.

We then have the following Lemma 9 to transform $\mathfrak{S}^k = Q\mathfrak{S}^{k-1} + b$ to facilitate the convergence study.

Lemma 9. *Let e be a $|S|$ dimensional vector with all its elements being 1, i.e. $e = (1,1,...,1)'$. If the largest eigenvalue of Q is smaller than 1, given any initial vector $\mathfrak{S}^0$ such that $\forall\ i,j \leq |S|$ $(i \neq j)$, $\mathfrak{s}_i{}^0 = \mathfrak{s}_j{}^0 \neq 0.5$, then $\mathfrak{S}^k = Q\mathfrak{S}^{k-1} + b$ can be represented as: $\frac{\mathfrak{S}^k}{\|Q^k e\|_1} = \frac{\theta Q^k e}{\|Q^k e\|_1} + \frac{0.5e}{\|Q^k e\|_1}$ where $-0.5 \leq \theta \leq 0.5$.*

Proof. Since $\mathfrak{S}^k = Q\mathfrak{S}^{k-1} + b$, after we substitute $\mathfrak{S}^{k-1}$ by $Q\mathfrak{S}^{k-2} + b$ and then substitute $\mathfrak{S}^{k-2}$ and so on, we arrive at $\mathfrak{S}^k = Q^k\mathfrak{S}^0 + (I + Q + ... + Q^{k-1})b$.

As the largest eigenvalue $\rho(Q) < 1$, we have $\lim_{k\to\infty} Q^k = \mathbf{0}$. $I + Q + ... + Q^k$ is in fact the Neumann Series, which has been shown to converge to $(I-Q)^{-1}$ [Mey01]. Therefore, $\mathfrak{S}^k$ converges to $(I-Q)^{-1}b$ for an arbitrary $\mathfrak{S}^0$.

Let R^k be another anomaly score vector and $R^0 = 0.5e$. According to Lemma 8, we have $R^1 = QR^0 + b = 0.5e$, $R^2 = QR^1 + b = 0.5e$, and in general, $R^k = QR^{k-1} + b = 0.5e$, for every $k \geq 1$.

Since the initial vector $\mathfrak{S}^0$ has equal elements that are not 0.5, we can denote any initial vector as $\mathfrak{S}^0 = (0.5 + \theta)e$. As the initial anomaly score is assumed to be within [0,1], we have $-0.5 \leq \theta \leq 0.5$.

For any θ, $(-0.5 \leq \theta \leq 0.5)$, we have $\mathfrak{S}^1 = Q\mathfrak{S}^0 + b = Q(0.5e + \theta e) + b = QR^0 + b + Q\theta e = R^1 + Q\theta e$. Similarly, we can show that $\mathfrak{S}^2 = R^2 + Q^2\theta e$.

Therefore, we have $\mathfrak{S}^k = \theta Q^k e + R^k = \theta Q^k e + 0.5e$, which can be represented as: $\frac{\mathfrak{S}^k}{\|Q^k e\|_1} = \frac{\theta Q^k e}{\|Q^k e\|_1} + \frac{0.5e}{\|Q^k e\|_1}$. □

With Lemma 9, in order to study the convergence of $\mathfrak{S}^k$, we can prove the convergence of $\frac{Q^k e}{\|Q^k e\|_1}$, since other parts of $\frac{\theta Q^k e}{\|Q^k e\|_1} + \frac{0.5e}{\|Q^k e\|_1}$ are of known value.

Lemma 10. $\frac{Q^k e}{\|Q^k e\|_1}$ *converges when $k \to \infty$.*

Proof. Let $\sigma(Q) = \{\lambda_1, \lambda_2, ..., \lambda_x\}$ denote x number of distinct eigenvalues of Q. According to [Mey01], there exists a nonsingular matrix P s.t.

$J = P^{-1}QP = \begin{pmatrix} J(\lambda_1) & 0 & \cdots & 0 \\ 0 & J(\lambda_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & J(\lambda_x) \end{pmatrix}$, J is called the Jordan form of Q, and each of the $J(\lambda_i)$ takes the form of $\begin{pmatrix} J_1(\lambda_i) & 0 & \cdots & 0 \\ 0 & J_2(\lambda_i) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & J_y(\lambda_i) \end{pmatrix}$, where y can be calculated as in [Mey01], $J_\star(\lambda_i) = \begin{pmatrix} \lambda_i & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{pmatrix}_{m \times m}$, where m can be calculated as in [Mey01]. Therefore, we have $Q = PJP^{-1}$, and $Q^k = PJ^kP^{-1}$. Since J is block diagonal, we have $J^k = \begin{pmatrix} J(\lambda_1)^k & 0 & \cdots & 0 \\ 0 & J(\lambda_2)^k & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & J(\lambda_x)^k \end{pmatrix}$. Since $J(\lambda_i)$ is also block diagonal, we are interested to know the form of $J_\star(\lambda_i)^k$ with $m \times m$, which can be shown as

$$J_\star(\lambda_i)^k = \begin{pmatrix} \lambda_i^k & \binom{k}{1}\lambda_i^{k-1} & \cdots & \binom{k}{m-1}\lambda_i^{k-m+1} \\ & \lambda_i^k & \ddots & \vdots \\ & & \ddots & \binom{k}{1}\lambda_i^{k-1} \\ & & & \lambda_i^k \end{pmatrix}$$

Let D be the set of distinct terms that appear in at least one element of $Q^k e$. Each term in D is of form $\begin{pmatrix} k \\ m-1 \end{pmatrix} \lambda^{k-m+1}$, where $m \geq 1$ and $\lambda \in \sigma(Q)$. Therefore, $Q^k e = \begin{pmatrix} \vdots \\ \sum_j c_{i,j} d_{i,j} \\ \vdots \end{pmatrix}$. Here i represents the i-th element of $Q^k e$, j represents the j-th term of an element, and $c_{i,j}$ is a non-zero real value.

It can be easily proven as follows that there exists a $d^* \in D$, such that $\forall d \in D - \{d^*\}$, $\lim_{k \to \infty} \frac{|d|}{|d^*|} = 0$. For any given two terms in D, we have

(I) if the terms contain the same λ, then $\begin{pmatrix} k \\ m_1 \end{pmatrix} / \begin{pmatrix} k \\ m_2 \end{pmatrix} \to 0$, when $k \to \infty, m_1 < m_2$.

(II) if the terms of an element contain different λ_1 and λ_2, then $\begin{pmatrix} k \\ m \end{pmatrix} (\frac{\lambda_1}{\lambda_2})^k \to 0$, when $k \to \infty, \lambda_1 < \lambda_2$.

Therefore, there exists a $d^* \in D$ such that $\forall d \in D - \{d^*\}$, $\lim_{k\to\infty} \frac{|d|}{|d^*|} = 0$. As $\frac{Q^k e}{\|Q^k e\|_1} = \frac{Q^k e}{\Sigma_i |\Sigma_j c_{i,j} d_{i,j}|} = \frac{Q^k e/|d^*|}{\Sigma_i |\Sigma_j c_{i,j} d_{i,j}/d^*|}$, we have, $\lim_{k\to\infty} \frac{Q^k e}{\|Q^k e\|_1} = \lim_{k\to\infty} \frac{Q^k e/|d^*|}{\Sigma_i |\Sigma_j c_{i,j} d_{i,j}/d^*|}$.

For the denominator, for the terms $d_{i,j} = d^*$, the limit of $d_{i,j}/d^*$ is 1, whereas for the terms $d_{i,j} \neq d^*$, $d_{i,j}/d^*$ is 0. Hence, the limit of the denominator is the sum of the $c_{i,j}$ where the corresponding $d_{i,j} = d^*$. On the other hand, the limit of the numerator is a vector of real values where only the elements whose terms contain d^* is non-zero value. Hence, the whole fraction converges. Therefore, $\frac{Q^k e}{\|Q^k e\|_1}$ converges when $k \to \infty$. □

With Lemma 10, we have the following Theorem 3 regarding the convergence of anomaly scores $\mathfrak{S}^k$:

Theorem 3. *For $\mathfrak{S}^k = Q\mathfrak{S}^{k-1} + b$, where $k \geq 1$, if the largest eigenvalue of Q is smaller than 1, given any initial vector $\mathfrak{S}^0$ such that $\forall\, i, j \leq |S|$ $(i \neq j)$, $\mathfrak{s}_i{}^0 = \mathfrak{s}_j{}^0 \neq 0.5$, then $\mathfrak{S}^k$ converges when $k \to \infty$.*

With Lemma 9 and Lemma 10, this theorem is obvious.

We now study the convergence of rankings of nodes according to their anomaly scores $\mathfrak{S}^k$. We have the following:

Theorem 4. *For $\mathfrak{S}^k = Q\mathfrak{S}^{k-1} + b$, where $k \geq 1$, if the largest eigenvalue of Q is smaller than 1, given any initial vector $\mathfrak{S}^0$ such that $\forall\, i, j \leq |S|$ $(i \neq j)$, $\mathfrak{s}_i{}^0 = \mathfrak{s}_j{}^0 \neq 0.5$, then $\exists$ an integer $K > 0$ such that the ranking of elements in $\mathfrak{S}^k$ will stay unchanged $\forall\, k > K$.*

Proof. Since Q's largest eigenvalue is smaller than 1, and the initial vector $\mathfrak{S}^0$ contains identical values not equal to 0.5, according to Lemma 9, we have $\frac{\mathfrak{S}^k}{\|Q^k e\|_1} = \frac{\theta Q^k e}{\|Q^k e\|_1} + \frac{0.5e}{\|Q^k e\|_1}$. Let $\frac{Q^k e}{\|Q^k e\|_1}(i)$ denote the i-th element of vector $\frac{Q^k e}{\|Q^k e\|_1}$. Let $\Delta =$

$min_{i,j}(\frac{Q^k e}{\|Q^k e\|_1}(i) - \frac{Q^k e}{\|Q^k e\|_1}(j))$, then for any i, j, we have $|\frac{Q^k e}{\|Q^k e\|_1}(i) - \frac{Q^k e}{\|Q^k e\|_1}(j)| \geq \Delta$.

According to Lemma 10, $\frac{Q^k e}{\|Q^k e\|_1}$ converges. Hence, for any real number $\varepsilon > 0, \exists$ an integer $K > 0$, s.t., $\forall k > K$, $max_i(|\frac{Q^{k+1} e}{\|Q^{k+1} e\|_1}(i) - \frac{Q^k e}{\|Q^k e\|_1}(i)|) < \varepsilon$. Therefore, if ε is smaller than $\Delta/2$, the above still holds.

Suppose $\frac{Q^k e}{\|Q^k e\|_1}(i) > \frac{Q^k e}{\|Q^k e\|_1}(j)$, for any $\varepsilon < \Delta/2$, we have $\frac{Q^{k+1} e}{\|Q^{k+1} e\|_1}(i) > \frac{Q^k e}{\|Q^k e\|_1}(i) - \varepsilon$, and $\frac{Q^{k+1} e}{\|Q^{k+1} e\|_1}(j) < \frac{Q^k e}{\|Q^k e\|_1}(j) + \varepsilon$. Since $|\frac{Q^{k+1} e}{\|Q^{k+1} e\|_1}(i) - \frac{Q^k e}{\|Q^k e\|_1}(i)| < \varepsilon$ and $|\frac{Q^{k+1} e}{\|Q^{k+1} e\|_1}(j) - \frac{Q^k e}{\|Q^k e\|_1}(j)| < \varepsilon$, we always have $\frac{Q^{k+1} e}{\|Q^{k+1} e\|_1}(i) > \frac{Q^{k+1} e}{\|Q^{k+1} e\|_1}(j)$.

Therefore, the relative order of $\frac{Q^k e}{\|Q^k e\|_1}(i)$ and $\frac{Q^k e}{\|Q^k e\|_1}(j)$ stays unchanged for all $k > K$. Therefore, we have proven that there exists an integer $K > 0$, s.t. the ranking of all elements in $\mathfrak{S}^k$ never changes $\forall k > K$. □

With the proven Theorem 4, we know that with certain assumption about Q and as long as we set the identical initial value for all source nodes, the ranking of source nodes will stay the same after a certain number of iterations. Now the question is, will different runs with initial vectors of different identical values converge to the same ranking?

The following theorem shows that (I)If two different runs involve different identical initial values that are both smaller 0.5, the final rankings are the same; (II)if two different runs involve different identical initial values that are both greater than 0.5, the final rankings are the same; (III)If one run involves initial values smaller than 0.5 and another run involves initial values greater than 0.5, the final rankings are the opposite.

Theorem 5. *For $\mathfrak{S}^k = Q\mathfrak{S}^{k-1} + b$, where $k \geq 1$, if the largest eigenvalue of Q is smaller than 1, we have:*
(I) given one initial vector $\mathfrak{S}^0$, such that $\forall\ i, j \leq |S|$ $(i \neq j)$, $\mathfrak{s}_i{}^0 = \mathfrak{s}_j{}^0 < 0.5$, and another initial vector $\mathfrak{S}^{*0}$, such that $\forall\ i, j \leq |S|$ $(i \neq j)$, $\mathfrak{s}_i{}^0 = \mathfrak{s}_j{}^0 < 0.5$, then there exist an integer $K > 0$, such that the two rankings $\mathfrak{S}^k$ and $\mathfrak{S}^{*k}$ are identical $\forall k > K$.
(II) given one initial vector $\mathfrak{S}^0$, such that $\forall\ i, j \leq |S|$ $(i \neq j)$, $\mathfrak{s}_i{}^0 = \mathfrak{s}_j{}^0 > 0.5$, and another initial vector $\mathfrak{S}^{*0}$, such that $\forall\ i, j \leq |S|$ $(i \neq j)$, $\mathfrak{s}_i{}^0 = \mathfrak{s}_j{}^0 > 0.5$, then there exist an integer $K > 0$, such that the two rankings $\mathfrak{S}^k$ and $\mathfrak{S}^{*k}$ are identical $\forall k > K$.

(III) given one initial vector $\mathfrak{S}^0$*, such that* $\forall\ i, j \leq |S|$ *(*$i \neq j$*),* $\mathfrak{s}_i{}^0 = \mathfrak{s}_j{}^0 < 0.5$*, and another initial vector* $\mathfrak{S}^{*0}$*, such that* $\forall\ i, j \leq |S|$ *(*$i \neq j$*),* $\mathfrak{s}_i{}^0 = \mathfrak{s}_j{}^0 > 0.5$*, then there exist an integer* $K > 0$*, such that the two rankings* $\mathfrak{S}^k$ *and* $\mathfrak{S}^{*k}$ *are exactly the opposite* $\forall k > K$*.*

Proof. Since Q's largest eigenvalue is smaller than 1, and the initial vector $\mathfrak{S}^0$ contains identical values not equal to 0.5, according to Lemma 9, we have $\frac{\mathfrak{S}^k}{\|Q^k e\|_1} = \frac{\theta Q^k e}{\|Q^k e\|_1} + \frac{0.5e}{\|Q^k e\|_1}$, where $-0.5 \leq \theta \leq 0.5$.

Since the initial vector $\mathfrak{S}^0$ has equal elements not equal to 0.5, we have denoted in Lemma 9 any initial vector as $\mathfrak{S}^0 = (0.5 + \theta)e$. When $\theta > 0$, the initial vector has elements larger than 0.5, whereas when $\theta < 0$, the initial vector has elements smaller than 0.5.

Therefore, (I) actually states $\mathfrak{S}^0 = (0.5 + \theta)e$, $\mathfrak{S}^{*0} = (0.5 + \theta^*)e$ with $\theta < 0$ and $\theta^* < 0$. (II) involves $\theta > 0$ and $\theta^* > 0$ and (III) suggests $\theta < 0$ and $\theta^* > 0$.

Thus, we have $\mathfrak{S}^k = \theta Q^k e + 0.5e$ and $\mathfrak{S}^{*k} = \theta^* Q^k e + 0.5e$. It is easy to see that if (I)$\theta < 0$ and $\theta^* < 0$ or (II)$\theta > 0$ and $\theta^* > 0$, $\mathfrak{S}^k$ will only differ from $\mathfrak{S}^{*k}$ by some scale. The rankings of all elements will be the same. When (III)$\theta < 0$ and $\theta^* > 0$, any two elements in $\mathfrak{S}^k$ will have a reverse ranking in $\mathfrak{S}^{*k}$. Thus $\mathfrak{S}^k$ is a reverse ranking of $\mathfrak{S}^{*k}$.

According to Theorem 4, suppose K and K^* are the two integers where the rankings of $\mathfrak{S}^k$ and $\mathfrak{S}^{*k}$ stay unchanged, we know $\forall k > \max(K, K^*)$, both rankings will stay unchanged. Therefore, we have proven (I), (II) and (III). □

Theorem 5 suggests that if we set the initial value to be smaller than 0.5, implying originally all nodes are normal, we will get the ranking that is exactly the opposite of that we get if we set the initial value to be larger than 0.5, implying originally all nodes are anomalous. Thus, the initial values act as the "prior view" towards all nodes. Since we assume the normal nodes are the majority, we therefore should always set the initial value to be smaller than 0.5.

5.2.5 Iterative Algorithm

With these theorems we now design our iterative algorithm IMD, Algorithm 14 to compute the ranking of source nodes. The ranking of target nodes can be computed similarly.

Algorithm 14 IMD, *Iterative algorithm to compute ranking of source nodes.*

Input: bipartite graph $G = \langle S \cup T, E, A \rangle$, initial value x,
number of iterations ranking stays unchanged y.
Output: rankings of S, $\widehat{S}$.

1: Set $\mathfrak{S}^0$=$(x, x, ..., x)'$.
2: Let $\widehat{S^*}$ be the ranking of S according to $\mathfrak{S}^0$.
3: c=0; k=1.
4: **while** $c \leq y$ **do**
5: Compute anomaly score for source nodes as in Formula 5.2,
$\mathfrak{S}^k = (W^S - 2X)(W^T - 2Y)\mathfrak{S}^{k-1} + (W^S - 2X)\overline{Y} + \overline{X}$.
6: $\widehat{S}$ keeps the ranking of S according to $\mathfrak{S}^k$.
7: **if** the distance between $\widehat{S}$ and $\widehat{S^*}$ is 0 **then**
8: c++.
9: **else**
10: $c = 0$.
11: $\widehat{S^*} = \widehat{S}$.
12: k++.
13: **return** $\widehat{S}$

We set the initial score vector of both source and target nodes as $(0.1, 0.1, ..., 0.1)'$, since we assume all nodes are normal in the beginning. Different values that are smaller than 0.5 gives the same ranking, as proven in Theorem 5. We measure the distance of the two consecutive iterations by Kullback-Leibler divergence [KL51]. If the rankings stay unchanged for a certain number of times y, which is set to a small number such as 10, the algorithm stops. Note that the purpose of y is for observing convergence, as we have proven the ranking should converge if certain conditions are met. In practice, the actual number of iterations needed is small, which is studied in the experiments section.

Time Complexity of algorithm IMD. We first analyze the complexity of step 5 of IMD. It involves multiple matrix multiplications, whose time complexity is of $O(n^2)$, where n is the number of nodes in the graph. Suppose the number of iterations needed to convergence is ρ, then the time complexity of algorithm IMD is of $O(n^2\rho)$.

5.3 Experiments on Synthetic Data

In this section, we evaluate our node anomaly detection framework on synthetic data. We generate synthetic bipartite graphs with properties that are necessary for testing different detection models. We show the precisions of our method as well as other existing methods on these synthetic graphs.

5.3.1 Synthetic Data Generation Algorithm without Noise

In Algorithm 15, we first set aside 4 sets of nodes as anomalous source nodes S^A, anomalous target nodes T^A, normal source nodes S^N and normal target nodes T^N and then generate edges among them. In order for an injected anomalous node to be anomalous, it has to possess at least one kind of anomalous characteristics: giving/receiving disagreeing edges to/by normal nodes or giving/receiving agreeing edges to/by anomalous nodes. Similarly, for an injected normal node to be normal, it has to have at least one kind of normal characteristics: giving/receiving disagreeing edges to/by anomalous nodes or giving/receiving agreeing edges to/by normal nodes.

We generate edges in the following sequence: from S^A to T^A, from S^A to T^N, from S^N to T^A, and then from S^N to T^N. We introduce parameters in the data generation algorithm such that it generates graphs of different properties (e.g., the ratio of disagreeing edges to agreeing edges), which are necessary to test our method as well as others. We also make sure that the algorithm generates these properties for source and target nodes simultaneously so that later steps would not mess up the properties generated by the previous steps. Parameter n controls the size of S^N and T^N, α is the ratio of $|S^A|$ to $|S^N|$ or the ratio of $|T^A|$ to $|T^N|$, β is the ratio of the expected number of disagreeing edges to the expected number of agreeing edges for any anomalous node, and $1/\gamma$ is the ratio of the expected number of disagreeing edges to the expected number of agreeing edges for any normal node.

Figure 5.1 illustrates the properties of our generated graphs. Specifically, after step 6, it can be easily shown that for any anomalous source, the expectation of the number of agreeing edges is $\alpha \cdot n/2$; and the expectation of the number of disagree-

Algorithm 15 *Generate synthetic bipartite graph*

Input: parameters: n, α, β, γ
Output: bipartite graph, G

1: Generate node sets for G: S^N, S^A, T^N, T^A such that $|S^N| = |T^N| = n$, $|S^A| = |T^A| = \alpha \cdot n$, $(0 < \alpha < 1)$
2: **for** each node s in S^A **do**
3: Randomly select a target node set $T^* \subset T^A$, where $|T^*|$ is randomly drawn from $[0, \alpha \cdot n]$, $(\alpha \cdot n \leq |T^A| = \alpha \cdot n)$.
4: Generate agreeing edges from s to each node in T^*.
5: Randomly select a target node set $T^* \subset T^N$, where $|T^*|$ is randomly drawn from $[0, \alpha \cdot n \cdot \beta]$, $(\alpha \cdot n \cdot \beta \leq |T^N| = n)$.
6: Generate disagreeing edges from s to each node in T^*.
7: **for** each node t in T^A **do**
8: Randomly select a source node set $S^* \subset S^N$, where $|S^*|$ is randomly drawn from $[0, \alpha \cdot n \cdot \beta]$, $(\alpha \cdot n \cdot \beta \leq |S^N| = n)$.
9: Generate disagreeing edges from each node in $|S^*|$ to t.
10: **for** each node s in S^N **do**
11: Randomly select a target node set $T^* \subset T^N$, where $|T^*|$ is randomly drawn from $[0, \alpha^2 \cdot n \cdot \beta \cdot \gamma]$, $(\alpha^2 \cdot n \cdot \beta \cdot \gamma \leq |T^N| = n)$.
12: Generate agreeing edges from s to each node in $|T^*|$.
13: **return** G

ing edges is $\alpha \cdot n \cdot \beta/2$.

We can also compute for any node in T^N, the expected number of edges linking from S^A as: $\frac{1}{\alpha \cdot n \cdot \beta + 1} \cdot \sum_{i=0}^{\alpha \cdot n \cdot \beta} \frac{i}{n} \cdot \alpha \cdot n = \alpha^2 \cdot n \cdot \beta/2$.

Similarly, since $|S^A| = |T^A|$, for any node in T^A, the expected number of edges linking from S^A is exactly $\alpha \cdot n/2$.

After step 9, for any anomalous target in T^A, the expectation of the number of edges from S^N is $\alpha \cdot n \cdot \beta/2$. Similarly, for any node in S^N, the expected number of edges to T^A is $\alpha^2 \cdot n \cdot \beta/2$.

After step 12, we can compute that for any node in S^N, the expectation of the number of edges to T^N is $\alpha^2 \cdot n \cdot \beta \cdot \gamma/2$. Since $|S^N| = |T^N|$, for any node in T^N, the expectation of the number of edges from S^N is also $\alpha^2 \cdot n \cdot \beta \cdot \gamma/2$.

Now, for any node in S^A or in T^A, the ratio of the expected number of disagreeing edges to the expected number of agreeing edges is $\frac{\alpha \cdot n \cdot \beta/2}{\alpha \cdot n/2} = \beta$. On the other hand, for any node in S^N or in T^N, the ratio of the expected number of disagreeing edges to the expected number of agreeing edges is $\frac{\alpha^2 \cdot n \cdot \beta/2}{\alpha^2 \cdot n \cdot \beta \cdot \gamma/2} = 1/\gamma$. These ratios are controlled by varying parameters of α, β and γ.

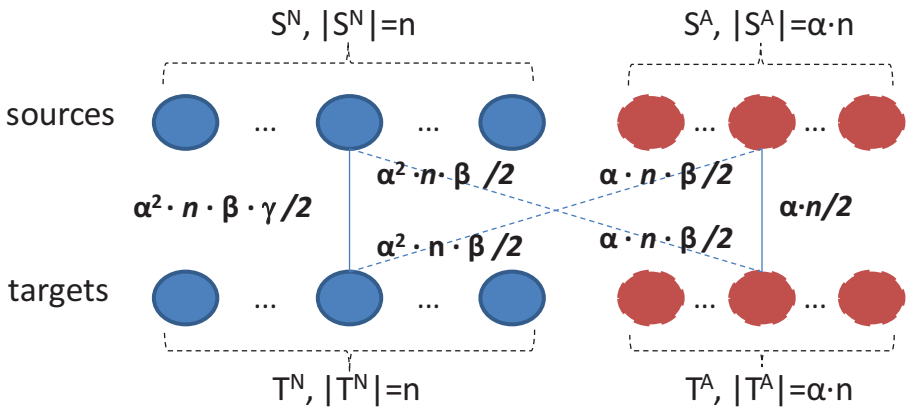

Figure 5.1: Synthetic bipartite graph. S^A: anomalous source nodes, T^A: anomalous target nodes, S^N: normal source nodes and T^N: normal target nodes.

In bipartite graphs, each edge carries the opinion of source node to target node. It is natural to expect that the opinions generated by the normal nodes prevail the opinions generated by the anomalous nodes, we should be able to set the parameters so as to control the opinions from anomalous nodes always being minority. Specifically, since disagreeing edges are always between anomalous and normal nodes, the opinions from normal and anomalous nodes carried by disagreeing edges are always equal. Thus, we want the opinions in the form of expected number of agreeing edges among normal nodes to be larger than those among anomalous nodes.

The expected number of agreeing edges among normal nodes is $(|S^N|+|T^N|)\cdot \alpha^2\cdot n\cdot\beta\cdot\gamma/2 = \alpha^2\cdot n^2\cdot\beta\cdot\gamma$. The expected number of agreeing edges among anomalous nodes is $(|S^A|+|T^A|)\cdot\alpha\cdot n/2 = \alpha^2\cdot n^2$. According to the anomaly being minority assumption, we have $\alpha^2\cdot n^2\cdot\beta\cdot\gamma > \alpha^2\cdot n^2$, which leads to $\beta > 1/\gamma$. Thus, we set β to be larger than $1/\gamma$ to guarantee the graphs are properly generated.

Other parameter constraints can be derived from the generation algorithm. They are $\alpha\cdot n\cdot\beta \leq 1$ derived from step 5 and step 8 and $\alpha^2\cdot\beta\cdot\gamma \leq 1$ derived from step 11.

5.3.2 Results

We compare our model denoted as IMD with the models that are based on only one of the mutual dependency principles. We denote the one with positive mutual dependency principle as PMD implemented as in [Zha02] and the one with negative mutual dependency principle as NMD implemented as in [LLW08]. We also incorporate the random guess method denoted as RG. We vary the parameters and test the precisions of all methods.

We first test with a fixed α and β, how $1/\gamma$ affects the results. Note that both β and $1/\gamma$ can be set as any real number. If they are smaller than one, agreeing edges dominate for each node. If they are larger than one, disagreeing edges dominate. We set $|S^N| = |T^N| = n$ as 200 and set $\alpha = 0.5$. β can be derived as smaller than or equal to 2. We set β=2 and set $1/\gamma$=$\{1.9, 1.5, 1, 0.5\}$ to test on graphs where agreeing edges dominate for anomalous nodes. We also try β=0.5 and vary $1/\gamma$ from $\{1/2.1, 1/4, 1/6, 1/8\}$ to test on graphs where agreeing edges dominate for anomalous nodes. The values for $1/\gamma$ satisfy the aforementioned parameter constraints. For each parameter setting, we generate 3 bipartite graphs and evaluate each method by precision@K, where K is the number of true anomalous source/target nodes. Our results with β=2 is shown in Figure 5.2(a) and the results with β=0.5 in Figure 5.2(b). Y-axis shows the average precision@K and X-axis shows different $1/\gamma$ values. The curve with IMD_s is regarding the performance of method IMD on source nodes. Similarly, IMD_t is for target nodes.

We can see from Figure 5.2(a) and Figure 5.2(b) that our method IMD gets perfect precision over all settings, much better than the competing ones. When β is fixed and $1/\gamma$ is varied. As $1/\gamma$ gets smaller, anomalous nodes are getting more characterized by disagreeing edges and less characterized by agreeing edges compared to normal ones. As a result, it becomes easier for the model NMD that propagates anomaly scores through disagreeing edges to identify anomalous nodes. At the same time it becomes harder for the model PMD that propagates anomaly scores through agreeing edges to identify anomalous nodes.

As IMD takes into consideration both mutual dependency principles and propagate anomaly scores on both types of edges, we are able to achieve the perfect

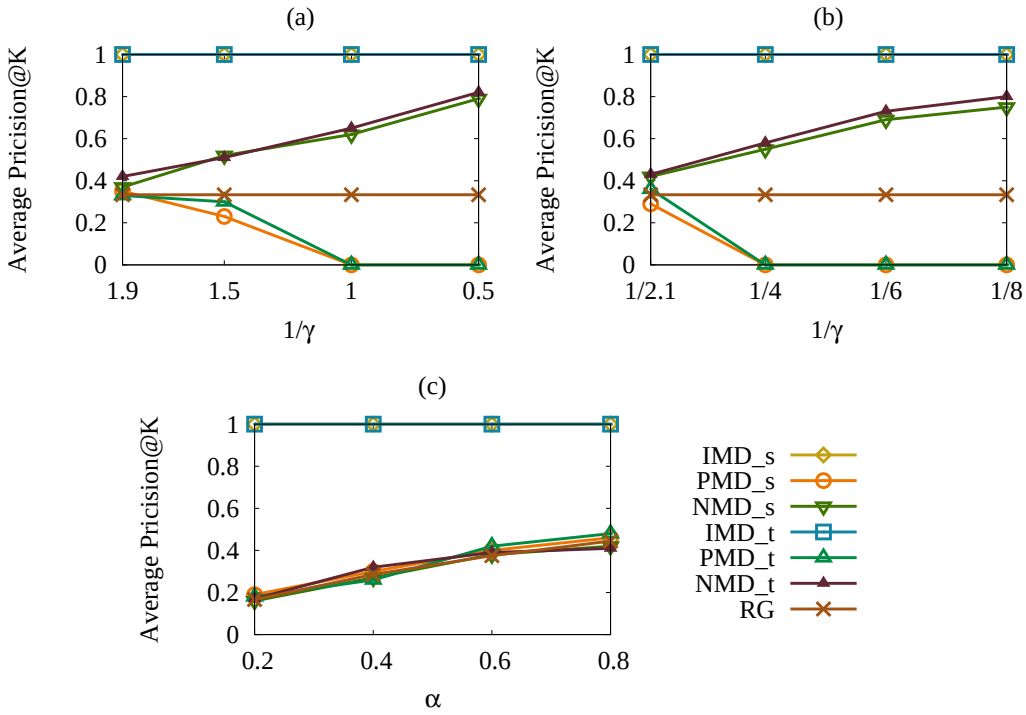

Figure 5.2: Results on synthetic data. Y-axis shows the average precision@K and X-axis shows different $1/\gamma$ values. (a) with α=0.5, β=2, varying $1/\gamma$; (b) with α=0.5, β=0.5 and varying $1/\gamma$; (c) with $\beta = 0.5$, $1/\gamma = 0.49$ and varying α.

precision. This shows that both the positive and negative mutual dependency relationships are necessary for anomaly detection.

To test whether the performance will be affected by the number of anomalous nodes α, we set $\beta = 0.5$ and $1/\gamma = 0.49$. The results are shown in Figure 5.2(c), which suggests that, as α changes, the performance of NMD and PMD are only as good as the random guess. The explanation is when $1/\gamma$ is only slightly smaller than β, the normal nodes and the anomalous nodes are giving out almost the same ratio of number of agreeing edges to disagreeing edges. This makes the anomalous nodes hard to discriminate from the normal ones for NMD and PMD.

5.3.3 Synthetic Data Generation Algorithm with Noise

We have shown the results on synthetic graphs generated under the assumption that normal nodes never behave anomalously and anomalous nodes never show any normal behaviors. This assumption makes sure that the anomalous and nor-

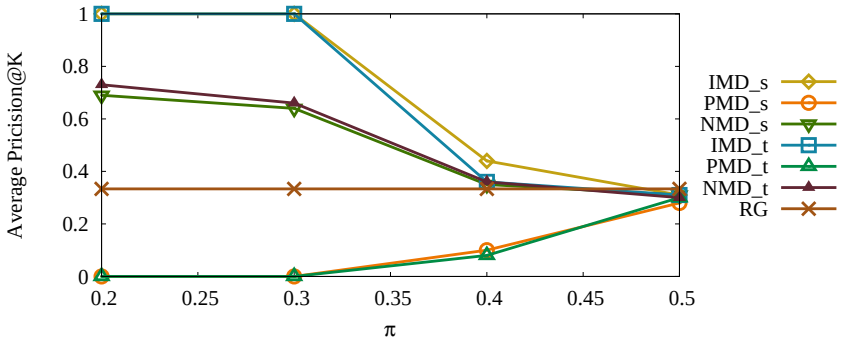

Figure 5.3: Results on synthetic data with varying noise level π.

mal nodes are truly the ground truth. Here we also experiment on synthetic bipartite graphs with noisy ground truth. In other words, anomalous nodes may possess some normal behaviors: giving/receiving disagreeing edges to/by anomalous nodes or giving/receiving agreeing edges to/by normal nodes; normal nodes may show anomalous behaviors: giving/receiving disagreeing edges to/by normal nodes or giving/receiving agreeing edges to/by anomalous nodes.

Parameter π is used to control the probability of a node having the behavior of its opposing role. Specifically, we modify the Algorithm 15 such that when agreeing edges are generated in step 4 and step 12, each edge has a probability of π of being a disagreeing edge. Similarly, when disagreeing edges are generated in step 6 and step 9, each edge has a probability of π of being an agreeing edge. Note that the larger the π is, the less likely the anomalous and normal nodes are the real ground truth.

5.3.4 Results

We vary π and set other parameters as $n = 200$, $\alpha = 0.5$, $\beta = 0.5$, $1/\gamma = 0.5$, as the competing method performs better on this setting. The results are shown in Figure 5.3. As we can see that, even in the presence of noise, IMD is still the best. Even when the noisy level is 0.3, IMD resists all noisy information. When $\pi = 0.5$, all methods are as good as the random guess. This is because, when anomalous nodes manifest the same amount of anomalous behavior as normal behaviors, anomalous nodes are no longer anomalous.

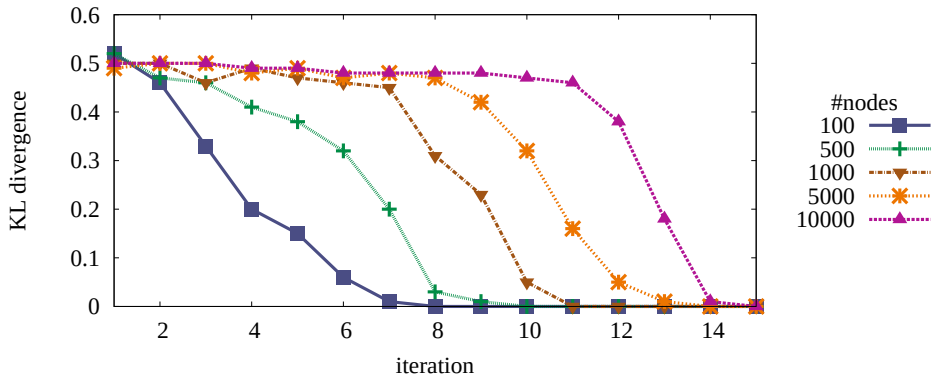

Figure 5.4: Results on convergence. Different curves corresponding to different graph sizes (number of nodes) show how the KL divergence between the node rankings of two consecutive iterations changes as the number of iterations increases.

Number of nodes n	100	500	1000	5000	10000
Running time	0.12	2.49	11.2	489	3193

Table 5.3: The running time of algorithm IMD with graphs of various number of nodes

5.3.5 Convergence Speed

Here we study during the iterative computation, how fast the rankings converge. We measure the distance between the rankings of two consecutive iterations by Kullback-Leibler divergence [KL51]. If two rankings are the same, the KL divergence distance is 0. We set $\alpha = 0.5$, $\beta = 0.5$, $1/\gamma = 0.5$ and $\pi = 0.1$. We vary the size of n from small to large, such that the total number of nodes in the generated graphs are from 100 to 10,000. The KL divergence distance v.s. the number of iterations curves for sources are shown in Figure 5.4. The curves for targets are similar and are omitted due to space constraint. Note that the figure shows the number of iterations until the distance becomes 0. In practice, our algorithm runs 10 more iterations to guarantee the distance does stay as 0 afterwards. We can see from the plot that our node ranking stays unchanged after only a small number of iterations. We also report the corresponding running time (in seconds) of algorithm IMD in Table 5.3. We can see that our algorithm scales as we expected in the time complexity analysis.

5.4 Experiments on Goodreads Data

Goodreads[1] is the largest website for people to write book reviews, rate books, recommend books to friends and socialize online with other readers or writers. According to its website, it has more than 9,000,000 members who have added more than 320,000,000 books to their shelves. To facilitate exploration, Goodreads creates "listopia" for users to quick find interesting books and maybe vote for their favorites. A book can be rated by any user on a scale of 5.

We utilize the API provided by Goodreads to crawl our data on May 20, 2012. Since the number of books and users are too large, we start from a popular book list in Goodreads' listopia, called "Dealbreakers: If You Like This Book, We Won't Get Along". We crawl from this list that books that have less than 1000 user ratings. We then crawl all the users who have rated these books and have rated less than 1000 books. We further go another hop to get all the books that are rated by the previously crawled users and have less than 1000 user ratings. The bipartite graph is thus generated with users as the source partite, books as the target partite and the edges suggests users rating books.

We only crawl the books and users with less than 1000 neighbors so that we would not run into some really popular books and end up with too large a dataset. The other reason is that the anomalous books and users with fewer ratings are easier to examine and make sense of.

As our model takes in edge labels which are either agreeing or disagreeing, we thus map the rating carried by an edge to a label as follows. If this rating is a minority rating of all ratings given to the same book, then this edge is given a disagreeing label; otherwise, this edge is given a agreeing label. The majority ratings are the ones within 2 standard deviation from the mean of all ratings given to a target.

We iteratively filter away target nodes with less than 10 edges and source nodes with less than 2 edges, as books rated by less than 10 users are not drawing enough attention from the audience and less likely to be spammed. Similarly users only rate one book are not influential enough, even if they are spammers. In the end, we have

[1] http://www.goodreads.com/

a bipartite graph with 7982 source nodes, 9169 target nodes and 163621 edges.

We apply our anomaly detection method on this bipartite graph. The top-1 anomalous book returned is "Justin Bieber: My World", published in August 1st 2010, and written by Justin Bieber. It received 100 ratings, with 48 of score 5, 4 of score 4, 4 of score 3, 3 of score 2 and 41 of score 1. Judging by this rating distribution, we know the opinions on this book are rather divided. Moreover, it is very hard to tell whether the users giving out 1 or the users giving 5 are spamming or anomalous simply by rating distribution. However, our approach is able to identify out of 99 users, 25 of them are anomalous users (i.e., users whose anomaly scores are greater than 0.5). Interestingly, these anomalous users are all giving low ratings (1 or 2) to this book and none of them gives textual comments to this book. Our model considers these 25 users anomalous as when they rate other books, they tend to disagree with normal users and agree with other anomalous users. We find that some of the 25 users including Angela and Kimiko, who rated around twenty books and mostly gave low ratings are not quite far away from the average rating. Based on our results, we may think that the book "Justin Bieber: My World" is unfairly demoted by some users.

Top-2nd is also a book by Justin Bieber, called "I ♡ Justin Bieber". It received 23 ratings, with 22 of them being 5. Unlike the top-1 book, this book is anomalous, as there are 9 identified anomalous users giving out score 5 to it. Almost all 9 users seem to be the fans of Justin Bieber. Their comments are about loving the person, not the book. One of them even has the user name as "JustinBieberLover". We therefore conclude that this book is somewhat spammed by his fans to promote the book.

Other top 10 anomalous books include 3 other Justin Bieber's books, "Kardashian Konfidential" by Kourtney Kardashian and "Birth Control Is Sinful in the Christian Marriages and Also Robbing God of Priesthood Children!!" by Eliyzabeth Yanne Strong-Anderson. All these books are identified by our model as being rated by some suspicious users.

5.5 Experiments on Buzzcity data

BuzzCity[2] is a global mobile advertising platform, where publishers host on their own websites the advertisement of advertisers. If anyone clicks on the advertisements, the publishers and the BuzzCity platform get money paid by advertisers. It is suspected that some fraudulent publishers would hire spammers to click on the advertisement hosted on their websites to gain money. To maintain a healthy ecosystem, BuzzCity has good intention to identify these spammers.

Buzzcity provides around 10 million click logs during a three day's period from Jan 26, 2012 to Jan 28, 2012. The dataset contains the encoded IP address of each click on each publisher. As a publisher may have multiple websites, the click data is already aggregated by publishers. Buzzcity has asked its own employees to label the publishers as "OK", "Observation" (meaning not sure) or "Fraud". This dataset will be made public for a fraud detection competition.

We apply our model to detect the fraudulent clickers (i.e., IP addresses) and publishers with clickers as one partite, publishers as the other partite and edges suggest users click on publishers. Since each edge carries the number of clicks from a clicker to a publisher, we map the number of clicks to agreeing and disagree edges as we do for the Goodreads data. We iteratively filter out nodes with less than 2 edges, as they are unlikely to be fraudulent. As a result, we have 132540 source nodes, 1428 target nodes, among which 1264 are OK, 77 are Observation and 87 are Fraud cases. Thus a random guess may achieve a precision of around 0.06.

The completing methods of our IMD are PMD with positive mutual dependency, NMD with negative mutual dependency, the distance-based anomaly detection approach DIST and a supervised approach, decision tree DT. For DIST, the distance between two publishers is defined as the KL divergence distance between the corresponding two click distributions of publishers. A click distribution of a publisher is the normalized histogram on the number of clicks from all clickers of this publisher. The distance-based anomaly score is computed as in [KN98]. As for DT, for each publisher, we define 7 features including total number of unique clickers, total number of clicks, the ratio of number of clicks to the total number of unique clickers,

[2]http://www.buzzcity.com/

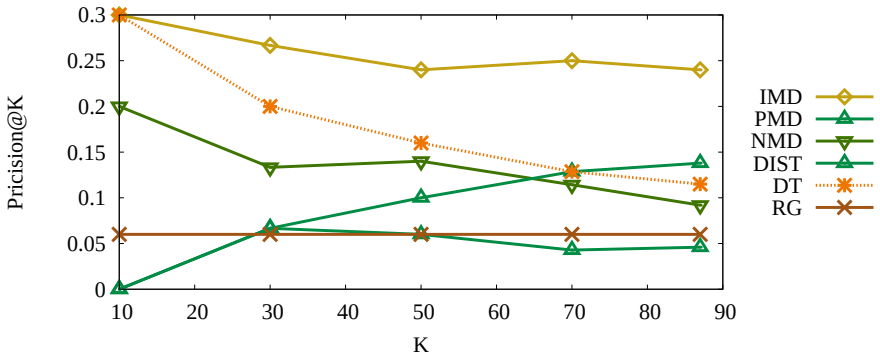

Figure 5.5: Precision@K curves on Buzzcity dataset.

as well as mean, median, standard deviation and skewness of the click distribution. We use C4.5 implemented by WEKA [HFH+09] with 10 fold cross-validation and output the prediction score of each publisher.

Since each method can output a ranking of the publishers according to its computed scores, we thus measure the performance of the top ranked publishers of each method by Precision@K. We vary K from small to as large as 87, the number of labeled fraudulent publishers. The results are shown in Figure 5.5. As we can see from the plot that our method IMD performs the best among all approaches.

5.6 Summary

We proposed a generic anomaly detection framework on bipartite graphs, based on the integral set of mutual dependency principles. We are the first to unify the positive and negative mutual dependency principles and design iterative algorithm with guarantee that the ranking of sources or targets will converge. We tested our framework on both the synthetic data and two real life datasets, namely Goodreads and Buzzcity. The results in these datasets show that our model outperforms the models with either positive or negative mutual dependency principles, which demonstrates the necessity of incorporating both principles for anomaly detection tasks. Moreover, we successfully identified suspicious users and books in Goodreads and achieved higher precision in detecting fraudulent publishers in Buzzcity than existing approaches.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This dissertation constitutes our research on detecting anomalies in social data. Thanks to the booming of online social media including Facebook and Twitter, large amount of social data is generated every second. The data captures the interactions and behaviors of multiple types of entities including human actors and resources. Another character of social data is its richness in both content and linkage information regarding the involving entities. Anomalies in social data are the entities that are inconsistent with the rest in terms of behaviors in the social network or features extracted from the social data. Examples of anomalies are spammers in Twitter, fraudulent users in online marketplace and opinion spammers in online forum.

Observations indicate that in social data anomalies often do not occur individually, but form small collections. Unlike the majority, entities in an anomalous collection tend to share certain extreme behavioral traits. We therefore propose the first type of problems, detecting anomalous collections. Furthermore, in a social network containing different types of entities, the anomalousness of one entity often depends on the entities it connects with, we therefore propose the second type of problems, detecting node anomalies in graphs. The first type of problems identifies anomalous collections of one type of entities by their collective extreme behaviors, whereas the second type of problems discovers node anomalies of two types of entities in social networks by their unusual interactions.

We introduce the problem of discovering extreme rank anomalous collections (ERAC). We propose both exact and heuristic algorithms for detecting top-K ERACs, for both independent and dependent feature sets. We apply our approach to detect web spammers in web host graph, anomalous actor groups in IMDB, as well as opinion spammers in Chinese online forum. The results on both data sets showed that the top ERACs are meaningful. Interestingly, the anomalous entities discovered by our approach are highly distinctive from those found by existing methods. Moreover, for web spam detection, our approach not only detects web spammer collections with higher precision than existing approaches, but also explains the anomaly statistically. The results on Chinese online forums indicate we find meaningful opinion spammer collections agreed by human evaluators. We also propose the follow-up problem of ERAC expansion, where the detected top-K ERACs are expanded to more anomalous supersets. Experiments on all datasets demonstrate the effectiveness of ERAC expansion.

We propose the problem of detecting top-K disjoint Coherent Anomaly Collections (CAC). We present an algorithm to identify CACs that does not need the collection number or collection size to be specified beforehand. The efficiency and effectiveness of the proposed algorithm are demonstrated by comparing with existing schemes on synthetic data. Our algorithm is also tested on three real world data sets, namely Twitter.com, a Webspam dataset and a Chinese online forum, to detect hashtag spammer collections, spamming website collections and opinion spammer collections respectively. The experiment results demonstrate that in all three datasets, our approach successfully finds suspicious spammer groups which are not easily identifiable with other approaches.

We also propose a generic anomaly detection framework on bipartite graphs, based on the integral set of mutual dependency principles. We are the first to unify the positive and negative mutual dependency principles and design iterative algorithm with guarantee that the ranking of sources or targets will stay unchanged for a certain number of iterations. We tested our framework on both the synthetic data and two real life datasets, namely Goodreads and Buzzcity. The results in these datasets show that our model outperforms the models with either positive or negative mutual dependency principles, which demonstrates the necessity of incorporating both

principles for anomaly detection tasks. Moreover, we successfully identified suspicious users and books in Goodreads and achieved higher precision in detecting fraudulent publishers in Buzzcity than existing approaches.

6.2 Future Work

We identify several future research directions to expand our work in detecting anomalous collections and anomalies in graphs.

Mining General Anomaly Collections.

Both of our ERAC and CAC detection frameworks are based on the "extreme rank" concept, where members of an anomaly collection consistently appear at extreme ranking positions of certain features. However, it is possible to have an anomalous collection defined to have members appearing consistently at some non-extreme positions. This general definition of anomaly collection would be able to utilize features where the behaviors of the anomaly collection are not extreme. One solution is to change the "extreme index" concept to a "closeness interval" in the entity ranking list. Given a collection and a closeness interval, the number of entities of this collection appearing in this closeness interval still follows the hypergeometric distribution. It would also be interesting to know whether more efficient algorithms can be designed and whether this new definition can retrieve more interesting anomalous collections in real data.

Mining Anomaly Collections in Data Streams

We have been assuming that the input data to our anomaly detection problems is static. In other words, we take a summarized view of the social data and then try to find the anomalies in this static view. It would be interesting to mine anomaly collections in data streams, where the entity population and the behavior of entities may change over time. For example spammers in Twitter, may be reported and deleted. New user accounts may be compromised and start to manifest spamming behaviors. These characteristics of data streams require a new definition of anomaly

collection incorporating the time dimension. We can derive the probability of each unusual pattern based on the previous n snapshots of the social network, and define the anomaly as the unusual patterns that are of small probabilities and are actually observed in the "future" snapshots.

Moreover, the stream data can be so huge in volume that we can only scan the data once. We thus need an efficient data structure and its corresponding algorithms to work on. One extension is to use a time window to sample the data along the time dimension. Efficient and appropriate ways to update our ranking lists associated with all features can be designed for detecting ERACs or CACs. We may also consider ways of weighting features by heuristics such as during the collection search process, if one feature contributes too few entities that are also appear in the extreme positions of other features, then we can prune this feature dynamically.

Mining Anomaly Collections in Graphs

Our work on detecting node anomalies in graphs derives the anomaly scores of nodes in bipartite graphs simultaneously. However, the scores are computed on individual node level based on the mutual dependency relationships among nodes. This mutual dependency may also exist at the collection level, where the anomalousness of a collection depends on that of other collections associated with it. The hard part is then how to derive the collections or how to aggregate the nodes to collections. One possible solution is to borrow the results of our work of ERAC and CAC detection to find these collections and then apply the mutual dependency principle. This combination may have the benefit of boosting precisions of finding spammers in real-life datasets, as this approach takes both the node feature and linkage information into consideration.

Bibliography

[ACCD11] Ery Arias-Castro, Emmanuel J. Candes, and Arnaud Durand. Detection of an anomalous cluster in a network. *Ann. Statist.*, 39(1), 2011.

[Agg05] C. Aggarwal. On abnormality detection in spuriously populated data streams. In *SDM Conf.*, 2005.

[Agg11] Charu C. Aggarwal. *Social Network Data Analytics.* Springer, 2011.

[AMF10] L. Akoglu, M. Mcglohon, and C. Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *PAKDD Conf.*, 2010.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *VLDB Conf.*, 1994.

[BCD+08] Luca Becchetti, Carlos Castillo, Debora Donato, Ricardo Baeza-YATES, and Stefano Leonardi. Link analysis for web spam detection. *ACM Trans. Web*, 2(1), 2008.

[BKNS00] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *SIGMOD Conf.*, 2000.

[BL94] V. Barnett and T. Lewis. *Outliers in statistical data.* John Wiley and Sons, 1994.

[BLZ+09] Jiang Bian, Yandong Liu, Ding Zhou, Eugene Agichtein, and Hongyuan Zha. Learning to recognize reliable users and content in social media with coupled mutual reinforcement. In *WWW Conf.*, 2009.

[BXG+13] Alex Beutel, Wanhong Xu, Venkatesan Guruswami, Christopher Palow, and Christos Faloutsos. Copycatch: stopping group attacks by spotting lockstep behavior in social networks. In *WWW Conf.*, 2013.

[CBK09] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), 2009.

[CDG+07] Carlos Castillo, Debora Donato, Aristides Gionis, Vanessa Murdock, and Fabrizio Silvestri. Know your neighbors: web spam detection using the web topology. In *SIGIR Conf.*, 2007.

[CPMF04] Deepayan Chakrabarti, Spiros Papadimitriou, Dharmendra S. Modha, and Christos Faloutsos. Fully automatic cross-associations. In *SIGKDD Conf.*, 2004.

[CW04] Cecil Eng Huang Chua and Jonathan Wareham. Fighting internet auction fraud: An assessment and proposal. *Computer*, 37(10), 2004.

[DSN08] Kaustav Das, Jeff Schneider, and Daniel B. Neill. Anomaly pattern detection in categorical datasets. In *SIGKDD Conf.*, 2008.

[Dun55] Charles W. Dunnett. A multiple comparison procedure for comparing several treatments with a control. *Journal of the American Statistical Association*, 50(272), 1955.

[DXLL09] Lian Duan, Lida Xu, Ying Liu, and Jun Lee. Cluster-based outlier detection. *Annals of Operations Research*, 168(1), 2009.

[DZLP12a] Hanbo Dai, Feida Zhu, Ee-Peng Lim, and HweeHwa Pang. Detecting anomalies in bipartite graphs with mutual dependency principles. In *ICDM Conf.*, 2012.

[DZLP12b] Hanbo Dai, Feida Zhu, Ee-Peng Lim, and HweeHwa Pang. Detecting extreme rank anomalous collections. In *SDM Conf.*, 2012.

[DZLP12c] Hanbo Dai, Feida Zhu, Ee-Peng Lim, and HweeHwa Pang. Mining coherent anomaly collections on web data. In *CIKM Conf.*, 2012.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *ICDM Conf.*, 1996.

[Epp05] David Eppstein. All maximal independent sets and dynamic dominance for sparse graphs. In *SODA Conf.*, 2005.

[FLN01] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In *PODS Conf.*, 2001.

[FMN04] Dennis Fetterly, Mark Manasse, and Marc Najork. Spam, damn spam, and statistics: using statistical analysis to locate spam web pages. In *Proc. of the 7th International Workshop on the Web and Databases*, 2004.

[GBGMP06] Zoltán Gyöngyi, Pavel Berkhin, Hector Garcia-Molina, and Jan Pedersen. Link spam detection based on mass estimation. In *VLDB Conf.*, 2006.

[GGMP04] Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. Combating web spam with trustrank. In *VLDB Conf.*, 2004.

[GRS99] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Rock: A robust clustering algorithm for categorical attributes. In *ICDE Conf.*, 1999.

[Har65] W. L. Harkness. Properties of the extended hypergeometric distribution. *The Annals of Mathematical Statistics*, 36(3), 1965.

[HFH+09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1), 2009.

[HLV03] Wenjie Hu, Yihua Liao, and V. Rao Vemuri. Robust anomaly detection using support vector machines. In *ICML Conf.*, 2003.

[HXD03] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recogn. Lett.*, 24(9), 2003.

[Ken48] Maurice G. Kendall. *Rank correlation methods*. Griffin, 1948.

[KJD09] Das Kaustav, Schneider Jeff, and B. Neill Daniel. *Detecting Anomalous Groups in Categorical Datasets*. CMU Technical Report, 2009.

[KL51] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1), 1951.

[Kle99] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5), 1999.

[Kle02] Jon Kleinberg. Bursty and hierarchical structure in streams. In *SIGKDD Conf.*, 2002.

[KN98] Edwin M. Knorr and Raymond T. Ng. Algorithms for mining distance-based outliers in large datasets. In *VLDB Conf.*, 1998.

[LC08] Shou-de Lin and Hans Chalupsky. Discovering and explaining abnormal nodes in semantic graphs. *IEEE Trans. on Knowl. and Data Eng.*, 20(8), 2008.

[LLW08] Hady W. Lauw, Ee-Peng Lim, and Ke Wang. Bias and controversy in evaluation systems. *IEEE Trans. on Knowl. and Data Eng.*, 20(11), 2008.

[LTS04] Antonio Loureiro, Luis Torgo, and Carlos Soares. Outlier detection using clustering methods: a data cleaning application. In *Proc. of the data mining for business workshop*, 2004.

[LTZ10] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. On detecting clustered anomalies using sciforest. In *ECML/PKDD Conf.*, 2010.

[Mey01] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra Book and Solutions Manual*. SIAM: Society for Industrial and Applied Mathematics, 2001.

[MLG12] Arjun Mukherjee, Bing Liu, and Natalie Glance. Spotting fake reviewer groups in consumer reviews. In *WWW Conf.*, 2012.

[MLW+11] Arjun Mukherjee, Bing Liu, Junhui Wang, Natalie Glance, and Nitin Jindal. Detecting group review spam. In *WWW Conf.*, 2011.

[NC03] Caleb C. Noble and Diane J. Cook. Graph-based anomaly detection. In *KDD Conf.*, 2003.

[PCWF07] Shashank Pandit, Duen Horng Chau, Samuel Wang, and Christos Faloutsos. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *WWW Conf.*, 2007.

[Rob86] J. M. Robson. Algorithms for maximum independent sets. *Journal of Algorithms*, 7(3), 1986.

[SEKX98] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data Min. Knowl. Discov.*, 2(2), 1998.

[SQCF05] Jimeng Sun, Huiming Qu, Deepayan Chakrabarti, and Christos Faloutsos. Neighborhood formation and anomaly detection in bipartite graphs. In *ICDM Conf.*, 2005.

[Tal05] Erik Talens. *Statistical auditing and the AOQL-method*. Labyrint Publication, 2005.

[TY06] Jun-ichi Takeuchi and Kenji Yamanishi. A unifying framework for detecting outliers and change points from time series. *IEEE Trans. on Knowl. and Data Eng.*, 18(4), 2006.

[WLLH08] Furu Wei, Wenjie Li, Qin Lu, and Yanxiang He. Query-sensitive mutual reinforcement chain and its application in query-oriented multi-document summarization. In *SIGIR Conf.*, 2008.

[WMCW03] Weng-Keen Wong, Andrew Moore, Gregory Cooper, and Michael Wagner. Bayesian network anomaly pattern detection for disease outbreaks. In *ICML Conf.*, 2003.

[Wu93] Trong Wu. An accurate computation of the hypergeometric distribution function. *ACM Trans. Math. Softw.*, 19(1), 1993.

[WXLY11] Guan Wang, Sihong Xie, Bing Liu, and Philip S. Yu. Review graph based online store review spammer detection. In *ICDM Conf.*, 2011.

[YHY07] Xiaoxin Yin, Jiawei Han, and Philip S. Yu. Truth discovery with multiple conflicting information providers on the web. In *SIGKDD Conf.*, 2007.

[YSZ02] Dantong Yu, Gholamhosein Sheikholeslami, and Aidong Zhang. Findout: finding outliers in very large datasets. *Knowl. Inf. Syst.*, 4(4), 2002.

[Zha02] Hongyuan Zha. Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. In *SIGIR Conf.*, 2002.