Ensemble and Multimodal Learning for Anomaly Mining: Algorithms and Applications

A Dissertation presented

by

Shebuti Rayana

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

Stony Brook University

August 2017

Stony Brook University

The Graduate School

Shebuti Rayana

We, the dissertation committe for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation

Leman Akoglu - Dissertation Advisor
Assistant Professor, Information Systems, Heinz College, Carnegie Mellon University
Research Assistant Professor, Computer Science, Stony Brook University

I. V. Ramakrishnan - Chairperson of Defense
Professor, Computer Science, Stony Brook University

Nickolaos Nikiforakis
Assistant Professor, Computer Science, Stony Brook University

Paul Fodor
Lecturer, Computer Science, Stony Brook University

Varun Chandola
Assistant Professor, Computer Science, SUNY Buffalo

This dissertation is accepted by the Graduate School

Charles Taber
Dean of the Graduate School

Abstract of the Dissertation

Ensemble and Multimodal Learning for Anomaly Mining: Algorithms and Applications

by

Shebuti Rayana

Doctor of Philosophy

in

Computer Science

Stony Brook University

2017

Anomaly detection is an important problem that has been studied in a broad spectrum of research areas due to its diverse applications in different domains. There exist many anomaly detection algorithms, among them some are domain specific and others are more generic. Despite a great amount of advance in this research area, there does not exist a single winning anomaly detector known to work well across different data sets. In fact, designing a single method that is effective on a wide range of domains is a challenging task. Moreover, real world data consists of multiple and diverse input modalities. Each modality is characterized by very different properties which make it difficult to ignore their differences. This requires designing of a multimodal learning approach by fusing various modalities into a single combined representation.

Ensemble techniques for classification and clustering have long proven effective, yet anomaly ensembles have been barely studied. In this dissertation, we tap into this gap and design new ensemble approaches for anomaly mining. Specifically, we design (i) an ensemble approach SELECT which employs novel techniques to systematically select the results from multiple anomaly detectors as well as consensus approaches to assemble, and (ii) a sequential ensemble approach CARE that employs a two-phase aggregation of the intermediate results of base detectors in each iteration to reach the final outcome by reducing both bias and variance. Both the approaches are fully unsupervised as ground truth is scarce in real-world data. We utilize SELECT for event detection in temporal graphs and both the ensemble approaches

for outlier detection in multidimensional point data (no-graph). We further improve CARE and develop iCARE a faster isolation based ensemble approach to be used for massive datasets.

Although diverse learning approaches for anomaly mining has been studied for decades, designing multimodal learning approaches for anomaly mining has been researched more recently. In this line of recent works, a useful application of multimodal learning is in opinion spam detection for online review data. We design a new holistic approach called SpEagle that utilizes clues from all metadata (text, timestamp, rating) as well as relational data (review-network), and harness them collectively under a unified framework to spot suspicious users and reviews. Moreover, this method can seamlessly integrate semi-supervision by incorporating labels and achieve improved performance. Furthermore, we improve the SpEagle framework with active inference. We design a method called Expected UnCertainty Reach (EUCR) which is used at each step to pick a node having high uncertainty from a dense region and close to other uncertain nodes. We evaluate our ensembles and multimodal learning approaches on large-scale real-world datasets and they provide improved performance over the existing baselines and state-of-the-art approaches.

Dedication Page

This dissertation is dedicated to my grandmother Begum Rizia Alam, my husband Ahsanul Karim and my son Audrik Arshan, for their endless love and sacrifices.

Contents

List of Figures

List of Tables

List of Symbols

Chapter 1

t, t' time points

Z Z-score (anomalousness score)

μ_t moving average

σ_t moving standard deviation

r_i rank of a point by detector i

R set of anomaly rank lists by different base detectors

O target anomalies (pseudo ground truth)

fR aggregated final rank list

$\mathbf{r}$ sorted normalized rank vector

$\hat{r}$ normalized ranks generated from uniform null distribution

$r_{(l)}$ normalized rank of a data point in list $l \in R$

T total time points

$pVals$ binomial probability matrix for normalized rank vectors

S_{sort} sorted index matrix for normalized rank vector

$p_{l,m}((\mathbf{r}))$ binomial probability of drawing at least l normalized rankings uniformly from $[0, 1]$ must be in the range $[0, r_{(l)}]$

ρ minimum of p-values

S set of anomaly score lists by different base detectors

P set of probability of anomalousness lists by different base detectors

$target$ pseudo ground truth

$wP()$ weighted Pearson correlation function

E set of selected lists by SELECT for ensemble

$class$ class labels, 1 for outliers and 0 for inliers

G_t snapshot of the graph G at time point t

M set of class labels list by different base detectors

m_{ind} index of minimum p-value

F list of inaccurate detectors for target anomalies

$count$ frequency of inaccurate detectors in F

C_l cluster of detectors with low $count$

C_h cluster of detectors with high $count$

w_i relative weight of base detector i for ULARA

D multi-dimensional point data

n number of nodes or number of data points

d dimension size of point data

w window size for EBED base detector

$u(t)$ eigen vector for time window t

$r(t)$ summary of past eigen vectors at time t

Chapter 2

D multi-dimensional point data

b number of feature bagged base detectors

ws weighted aggregation scorelist for data points

S subsample from D

th threshold calculated from Cantelli's inequality

E_A error events of a set of detectors in A

Y class labels of data points

f_i base detector i

e_A error rate of a set of detectors in A

ai, j agreement rate between detector i and detector j

n number of data points

U set of outliers across all the base detectors

$\hat{A}$ set of individual as well as pairs of detectors

$\epsilon_{\hat{A}}$ slack variable

w_i weight for detector i

T number of filtered top outliers

ψ subsample size for iNNE base detector

t number of random subsample of points to isolate them in hyperspheres

d dimension of point data

F set of base detectors

A set of all possible pairs of base detectors

fs outlierness scorelist for data points

r ranklist of data points

k number of nearest neighbors

Chapter 3 and 4

G user-review-product tripartite graph

ϕ_i prior potential of node i

s sign of an edge, e.g., '+' and '-'

ψ^s compatibility potential of edge with sign s

y assignment of labels to all nodes

y_i	node i's assigned label
Z	normalization constant
t	types of edges, e.g., '*belong*' and '*write*'
ψ_{ij}^t	compatibility potential of an edge from node i to node j with type t
$\mathcal{L}_U$	class labels for users
$\mathcal{L}_P$	class labels for products
V	set of nodes in network G
$\mathcal{L}_R$	class labels for reviews
F	number of features of a node
x_{li}	feature l value of node i
X_l	real values random variable associated with feature l
$f(.)$	a function to unify different feature values
S_i	spam score of a node i
T_i	type of node i
d_{ij}	rating deviation from node i's rating to node j's average rating
$m_{i \to j}$	message sent from each node i to each neighboring node j
$b_i(y_i)$	marginal probability called belief, of assigning each node of type T_i a label y_i
E	set of edges in network G
α, β	normalization constant
$\mathcal{B}$	budget of label acquisition strategy
x_A^*	most valuable node, where A is the label acquisition strategy
C	committee in QBC approach
$D(.)$	Kullback-Leibler (KL) Divergence
P_x	committee members that provide evidence for positive class
N_x	committee members that provide evidence for negative class

w_x	weight of a review node x calculated from its user-degree
p_x	random walk with restart probability vector of review node x
G_R	review-review graph, two reviews are connected if they share same reviewer
U	set of user nodes in network G
c	teleportation probability
W	column normalized adjacency matrix of G_R
e_X	unit vector containing 1 for node x and 0s for all other nodes
P	set of product nodes in network G
R	set of review nodes in network G
u	user node
p	product node
r	review node

Acknowledgements

First of all, I would like to thank Almighty for giving me the patience and perseverance to successfully complete my research work.

I am very fortunate to have Leman Akoglu as my advisor, who has been a great mentor for me both on my research as well as on my career. I wholeheartedly thank Leman for inspiring me all the time. Without her support and help this dissertation would not have been possible. I am also grateful to my dissertation committee chair I.V. Ramakrishnan and committee members Nickolaos Nikiforakis, Paul Fodor, and Varun Chandola for their time and invaluable feedback on my thesis.

I am grateful to Steven J. Cento and Tony Acosta from Northrop Grumman Aerospace Systems for giving me the opportunity to work on their research challenge at the initial stage of my PhD journey. Their critical feedback on my work helped me to improve my research strategy. I also want to thank Bing Liu from University of Illinois at Chicago, and Charu Aggarwal, Saket Sathe from IBM T. J. Watson for providing me important datasets for several experiments, without which my work would be incomplete. I also appreciate their brilliant comments and suggestions on my research work. My internship at IBM T. J. Watson Research Lab helped me to work with top class researchers from the industry. I am thankful to my manager Suresh Chari and my mentor Ian M Molloy for offering me a wonderful research environment.

I am also thankful to all my peers and friends of Stony Brook University, especially, Junting Ye, Wen Zhong, Hau Chan, Ariful Islam, and Rehana Akter for their help and for making my journey enjoyable. A special thanks to my friend Syed Ishtiaque Ahmed who has motivated me every time I stumbled during my PhD.

Thanks to the academic and administrative staffs of Computer Science Department: Betty Knittweis, Cynthia Scalzo and Kathy Germana. Their continuous help gave me a smooth life during my past five years journey at Stony Brook University. Thanks to Brian Tria, who was extremely helpful with the computing infrastructure.

I thank Stony Brook University for providing me the opportunity and the environment to grow as a researcher. I also want to thank to Northrop Grumman Aerospace Systems, National Science Foundation and Army Research Office for their financial support for my research work.

Finally, a special thanks to my family for their endless love and support. My grandmother Begum Rizia Alam; who inspired me in every step of my life, my father

Iqbal Ahmed; who believed in me and provided me all the support I needed. Thanks to my son Audrik Arshan for his love and all the sacrifices he made. His smile made me strong to achieve my goal during hardship. And last but not the least, I thank my husband Ahsanul Karim for his moral support, without him this journey would have been impossible.

Publications

1. Shebuti Rayana, Leman Akoglu. Less is More: Building Selective Anomaly Ensemble with Application to Event Detection in Temporal Graphs. *SIAM International Conference on Data Mining (SDM)*, Vancouver, BC, Canada, April, 2015

2. Shebuti Rayana, Leman Akoglu. Collective Opinion Spam Detection: Bridging Review Networks and Metadata. *ACM Special Interest Group on Knowledge Discovery and Data Mining (SIG-KDD)*, Sydney, Australia, August, 2015

3. Shebuti Rayana, Leman Akoglu. Collective Opinion Spam Detection using Active Inference. *SIAM International Conference on Data Mining (SDM)*, Miami, Florida, USA, May, 2016

4. Shebuti Rayana, Leman Akoglu. Less is More: Building Selective Anomaly Ensemble. *Transactions on Knowledge Discovery from Data (TKDD)*, May, 2016

5. Shebuti Rayana, Wen Zhong, Leman Akoglu. Sequential Ensemble Learning for Outlier Detection: A Bias-Variance Perspective. *IEEE International Conference on Data Mining (ICDM)*, Barcelona, Spain, 2016 (short paper)

Chapter 1

Introduction

In this thesis, we focus on mining *anomalies* in large-scale real-world data using robust and efficient algorithms and tools. Anomalies are points in the data that do not conform to the normal behavior. As such, anomaly detection refers to the problem of finding unusual points in the data that deviate from usual behavior. These non-conforming unusual points are often referred to as anomalies, outliers, exceptions, rare events etc. Most often, anomalies and outliers are two terms used interchangeably in various application domains.

In todays world, we create 2.5 quintillion bytes of data every day. Despite the availability of this enormous amount of data from different domains, spotting anomalies still remains challenging as their presence is often quite rare. For instance, the number of cyber attacks is significantly low compared to the typical flows in a computer network. Similarly, the fraudulent transactions are significantly rare compared to the typical transactions in a financial organization. Although such anomalies occur very infrequently, their impact can be devastating for an individual, an organization, or a community. Hence, detecting anomalies is considered to be extremely important and has been extensively studied in a broad spectrum of research areas in different domains, e.g., intrusion detection in cyber security [1], fault detection in safety critical systems (e.g., space craft) [2], spam detection in social networks [3, 4], malignant tumor detection in MRI image [5], fraud detection for credit cards [6], insurance or health care etc. Despite the fact that there exist many algorithms for anomaly mining [7], they are often insufficient or incompatible to address the huge multitude and magnitude of the impending threats that we are confronting in different spheres of our digital life. In this dissertation, we have designed, implemented, and tested several anomaly detection algorithms to overcome this problem.

Designing a single anomaly mining approach which is efficient as well as robust

to a wide range of datasets from diverse application domains is a challenging task. This has paved a way for the emergence of ensemble learning in anomaly mining. Ensemble methods utilize multiple algorithms to obtain better performance than the constituent algorithms alone and produce more robust results [8]. Thanks to these advantages, a large body of research has been devoted to ensemble learning in classification [9–11] and clustering [12, 13]. On the other hand, building effective ensembles for anomaly detection has proven to be a challenging task [14, 15]. A key challenge is the lack of ground-truth; which makes it hard to measure detector accuracy and to accordingly select accurate detectors to combine, unlike in classification. Moreover, there exist no objective or 'fitness' functions for anomaly mining, unlike in clustering. Moreover, real world data contains multiple and diverse types of input modalities, e.g., images are associated with captions and tags, review networks are associated with text reviews and ratings, videos contain both visual and audio signals, online posts contain urls and texts etc. It is hard to ignore the difference of the input modalities as they are characterized by very different statistical properties. This requires designing useful models by combining the different modalities into a joint representation which further increase the challenge of anomaly mining.

Therefore, in this dissertation, we focus on designing unsupervised ensemble and multimodal learning approaches for anomaly detection with novel applications in a variety of important domains including event detection in temporal networks (e.g., computer networks, social networks etc.), spam detection in online review networks, and outlier detection in multidimensional point datasets (e.g., disease, satellite, sensor datasets etc.). Our major contributions are as follows:

- *Designing selective anomaly ensemble approach* [16, 17]
  We develop SELECT, one of the first selective ensemble approaches for anomaly detection. As the name implies, the key property of this ensemble is its selection mechanism which carefully decides which results to combine from multiple heterogeneous base detectors in the ensemble. Our approach employs two novel unsupervised selection strategies that we design to choose the base detector as well as consensus results to combine. We apply SELECT on the event detection problem in temporal graphs as well as outlier detection problem in multi-dimensional point data (no-graph).

- *Designing sequential outlier ensemble approach* [18]
  We study the feasibility of bias-variance reduction under the unsupervised setting, and design a sequential ensemble model CARE, to reduce both bias and variance for outlier detection. Specifically, we use two aggregation phases, (i) we combine the results of feature-bagged base detectors using weighted aggregation, and (ii) the result of the current iteration is aggregated with the combined result from the previous iterations cumulatively. These two phase

aggregations in each iteration aim to reduce the variance. Furthermore, we remove the top outliers in successive iterations with an aim to reduce the bias. Furthermore, we improve CARE by incorporating an isolation based base detector and design iCARE which gives better performance both in terms of accuracy and running time.

- *Designing multimodal learning approach for collective opinion spam detection* [19]
  We design a multimodal learning approach SpEagle with an application to opinion spam detection in online review network. SpEagle utilizes clues from all of metadata (text, timestamp, rating) as well as relational data (review network), and combine them collectively under a unified framework to spot spam users, fake reviews, as well as targeted products. Moreover, SpEagle can seamlessly integrate labels on any subset of objects (user, review, and/or product) when available, without any changes in the algorithm. We also designed a lighter version of SpEagle to provide significant speed-up. We demonstrate the effectiveness and scalability of SpEagle on three real-world review datasets from Yelp.com with filtered (spam) and recommended (nonspam) reviews, where it significantly outperforms several baselines and state-of-the-art methods. To the best of our knowledge, this is the largest scale quantitative evaluation performed to date for the opinion spam problem.

- *Collective opinion spam detection using active inference* [20]
  According to our knowledge, this is the first work where we address the problem of the opinion spam detection with active inference. Our goal is to achieve improved performance of SpEagle over random selection within a small budget to label important nodes. Our key insight is to select nodes that (i) exhibit high uncertainty, (ii) reside in a dense region, and (iii) are closeby to other uncertain nodes in the network. Based on this insight, we design a utility measure, called Expected UnCertainty Reach (EUCR), and pick the node with the highest EUCR score at every step iteratively.

1.1 Overview

The rest of this thesis is organized as follows:

- In Chapter 2, we describe our motivation in designing a selective ensemble learning. Here, we present the SELECT approach for event detection in dynamic graphs as well as outlier detection in multidimensional data. We provide extensive evaluation of SELECT on thirteen real-world datasets and compare our results with existing ensembles.

- In Chapter 3, we present sequential ensemble approaches CARE and iCARE along with the justification of bias-variance reduction under an unsupervised framework. We provide a large-scale evaluation of our sequential method on twenty one real-world datasets and compare our results with state-of-the-art baseline approaches and ensembles.

- In Chapter 4, we present a multimodal learning approach SpEagle and apply this approach on collective opinion spam detection. Here, we provide a comprehensive evaluation of SpEagle on three real-world review datasets from Yelp.com and compare the performance with existing state-of-the-art opinion spam models.

- In Chapter 5, we address the problem of active inference for opinion spam detection. Here, we present a new utility measure EUCR to select important nodes for labeling with in a small budget. We adapt existing active inference approaches for opinion spam detection. Finally, we present evaluation of our approach on two large real-world datasets from Yelp.com and compare with the adapted active inference approaches.

- Finally, in Chapter 6, we summarize our contributions in anomaly mining literature and present some exciting future research directions.

Chapter 2

Selective Ensemble Learning for Anomaly Mining

Ensemble methods utilize multiple algorithms to obtain better performance than the constituent algorithms alone and produce more robust results [8]. Thanks to these advantages, a large body of research has been devoted to ensemble learning in classification [9–11, 21–23] and clustering [12, 13, 24–26]. On the other hand, building effective ensembles for anomaly detection has proven to be a challenging task [14, 15]. A key challenge is the lack of ground-truth; which makes it hard to measure detector accuracy and to accordingly select accurate detectors to combine, unlike in classification. Moreover, there exist no objective or 'fitness' functions for anomaly mining, unlike in clustering.

Existing attempts for anomaly ensembles either combine outcomes from all the constituent detectors [27–31], or induce diversity among their detectors to increase the chance that they make independent errors [32, 33]. However, as our prior work [34] suggests, neither of these strategies would work well in the presence of inaccurate detectors. In particular, combining all, including inaccurate results would deteriorate the overall ensemble performance. Similarly, diversity-based ensembles would combine inaccurate results for the sake of diversity. Moreover, using weighted aggregation approach to combine the constituent detectors as proposed by Klementiev et al. [35] also get hurt by the inaccurate detectors which we show in our experiments.

In this work, we tap into the gap between anomaly mining and ensemble methods, and propose SELECT, one of the first *selective* ensemble approaches for anomaly detection. As the name implies, the key property of our ensemble is its selection mechanism which carefully decides which results to combine from multiple different methods in the ensemble—as such, we find that "less is more". We summarize our

contributions as follows.

- We identify and study the problem of building selective anomaly ensembles in a fully unsupervised fashion.
- We propose SELECT, a new ensemble approach for anomaly detection, which utilizes not only multiple heterogeneous detectors, but also various consensus methods under a unified ensemble framework.
- SELECT employs two novel unsupervised selection strategies that we design to choose the detector/consensus results to combine, which render the ensemble not only more robust but improve its performance further over its non-selective counterpart.
- Our ensemble approach is general and flexible. It does not rely on specific data types (e.g., time series or point data), and allows other detectors and consensus methods to be incorporated.
- We provide theoretical evidence for our SELECT approach to achieve better accuracy compared to the base detectors and other baseline approaches.

We apply our ensemble approach to the event detection problem in temporal graphs as well as outlier detection problem in multi-dimensional point data (no-graph), where SELECT utilizes five heterogeneous event/outlier detection algorithms and seven different consensus methods. Extensive evaluation on datasets with ground truth shows that SELECT outperforms the average individual detector, the full ensemble that naively combines all results, the diversity-based ensemble in [32], as well as the weighted ensemble approach in [35].

2.1 Related Work

Event detection in temporal data and outlier detection in multi-dimensional point data are fundamental research problems that find numerous applications in the real world. As such, a large body of research has focused on building effective techniques for these problems. For information on such techniques we refer the reader to detailed surveys [7, 36, 37] and devote this section to discuss related work on ensemble learning, which is the main focus of our work.

Ensemble techniques leverage multiple different methods to obtain better performance than the individual methods in the ensemble [10]. This is achieved by combining the strengths of accurate methods and alleviating the weaknesses of the less accurate ones. For example, boosting [38] and stacking [39] directly integrate accuracy estimation within their iterative ensemble learning. Others assign expertise weights proportional to the accuracy of independent learners and externally

combine their results [40–43]. Ensembles are also known to produce more robust results. For example, bootstrap aggregating (or bagging) tends to reduce problems related to over-fitting to the training data [44].

Thanks to these advantages, ensemble learning has spurned a large body of work devoted to the study of ensemble classification and clustering [8,9,11–13,21–26]. On the other hand, building effective ensembles for anomaly detection has remained to be a challenging task, due to lack of ground truth and inherent objective functions [14, 15]. As such, there exist only a handful of mostly recent works on building anomaly ensembles [27–34].

Feature bagging [30] is the earliest work formalizing an outlier ensemble. It uses the same base algorithm (i.e., LOF [45]) on different feature subsets and employs a rank based merging to create the final consensus. Feature bagging is better calibrated by [28, 29] which convert the outlier scores to probability estimates and use score based merging. Different from those, [27, 31–34] utilize heterogeneous detectors (e.g., LOF [45], LOCI [46], k-distance, etc.).

Specifically, [27, 34] unify and combine results from all detectors, [31] uses accuracy on synthetically generated datasets to assign weights to the detectors, [35] calculates relative weights for the ranklists returned by the base detectors in an unsupervised way and [32,33] select the most diverse set of results to combine. Most of these methods are designed for outlier detection in clouds of data points. There exist very few anomaly ensemble approaches for data represented as graphs [34, 47]. In [47] Jing et al. proposed a weighted ensemble approach by iteratively maximizing the probabilistic consensus among the output of the base detectors. In our previous work [34] we combine all the results from the base detectors uniformly which reveals the fact that too much diversity among the base detectors hurts the final ensemble. Similar problem can occur in data fusion where web data from different sources are combined to build a knowledge base. Too much noise in several individual sources hurts the knowledge base. Srivastava et al. [48] provide a greedy approach to select the correlated data sources for data fusion to remove the erroneous data sources.

2.2 Background and Preliminaries

Event Detection Problem

Temporal graphs change dynamically over time in which new nodes and edges arrive or existing nodes and edges disappear. Many dynamic systems can be modeled as temporal graphs, such as computer, trading, transaction, and communication networks.

In this work, we consider temporal anomalies as events. Here, temporal anomalies are those time points at which the graph structure changes significantly. Event detection in temporal graph data is the task of finding the points in time at which the graph structure notably differs from its past. These change points may correspond to significant events; such as critical state changes, anomalies, faults, intrusion, etc. depending on the application domain. Formally, the problem can be stated as follows.

Given a sequence of graphs $\{G_1, G_2, \ldots, G_t, \ldots, G_T\}$;

Find time points t' s.t. $G_{t'}$ differs significantly from $G_{t'-1}$.

Outlier Detection Problem

A well known characterization of an outlier is given by Hawkins as, "an observation which deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism" [49]. A popular formulation of outlier detection is to find unusual points in multi-dimensional data by their distance to the neighboring points. Based on this notion there exist two most famous approaches for outlier detection (i) distance based, and (ii) density based methods. Specifically, distance based outlier detection problem is to find data points which are far from the rest of the data and density based methods find the points which reside in a lower density region compared to its nearest neighbors. Formally, the problem can be stated as follows.

Given a multi dimensional data D with n individual points and d dimensions;

Find points which are far from the rest of the data or reside in a lower density region.

2.2.1 Motivation for Ensembles

Several different methods have been proposed for the above problems, survey of which are given in [7, 36]. To date, however, there exists no single method that has been shown to outperform all the others. The lack of a winner technique is not a freak occurrence. In fact, it is unlikely that a given method could perform consistently well on different data of varying nature. Further, different techniques may identify different classes or types of anomalies depending on their particular formulation. This suggests that effectively *combining* the results from various different detection methods (detectors from here onwards) could help improve the detection performance.

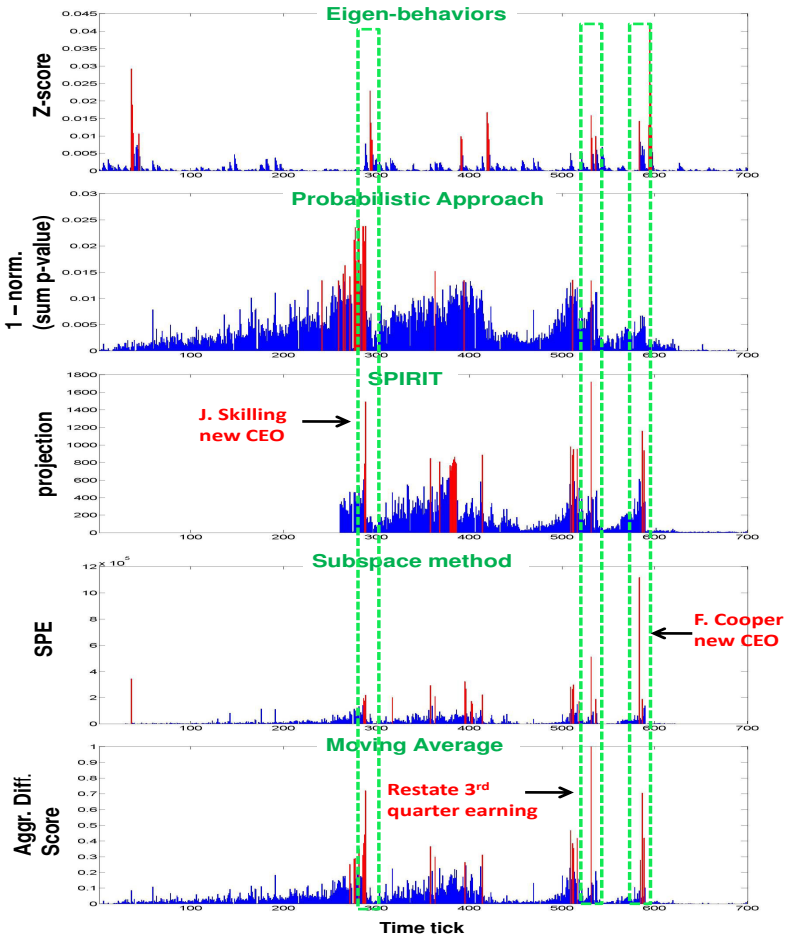


Figure 2.1: Anomaly scores from five detectors (rows) for the Enron Inc. time line. Red bars depict top 20 anomalous time points.

2.2.2 Motivation for Selective Ensembles

Ensembles are expected to perform superior to their average constituent detector, however a naive ensemble that trusts results from *all* detectors may not work well. The reason is, some methods may not be as effective as desired depending on the nature of the data in hand, and fail to identify the anomalies of interest. As a result, combining accurate results with inaccurate ones may deteriorate the overall ensemble performance [34]. This suggests that *selecting* which detectors to assemble is a critical aspect of building effective and robust ensembles—which implies that "less is more".

To illustrate the motivation for (selective) ensemble building further, consider the event detection example in Figure 2.1. The rows show the anomaly scores as-

signed by five different detectors to time points in the Enron Inc.'s time line. Notice that the scores are of varying nature and scale, due to different formulations of the detectors. We realize that the detectors mostly agree on the events that they detect; e.g., 'J. Skilling new CEO'. On the other hand, they assign different magnitude of anomalousness to the time points; e.g., the top anomaly of methods varies. These suggest that combining the outcomes could help build improved ranking of the anomalies. Next notice the result provided by "Probabilistic Approach" which, while identifying one major event also detected by other detectors, fails to provide a reliable ranking for the rest; e.g., it scores many other time points higher than 'F. Cooper new CEO'. As such, including this detector in the ensemble is likely to deteriorate the overall performance.

In summary, inspired by the success of classification and clustering ensembles and driven by the limited work on anomaly ensembles, we aim to systematically combine the strengths of accurate detectors while alleviating the weaknesses of the less accurate ones to build selective ensembles for anomaly mining. While we build ensembles for the event and outlier detection problems in this work, our approach is general and can directly be employed on a collection of detection methods for other anomaly mining problems.

2.3 SELECT: selective ensemble learning for anomaly detection

2.3.1 Overview

SELECT takes the input data, (i) for event detection a sequence of graphs $\{G_1, \ldots, G_t, \ldots, G_T\}$, and outputs a rank list R of objects (in this case time points $1 \leq t \leq T$), and (ii) for outlier detection d dimensional point data in D, and outputs a rank list of those data points, ranked from most to least anomalous.

The main steps of SELECT are given in Algorithm 1. Step 1 employs (five) different anomaly detection algorithms as base detectors of the ensemble. Each detector has a specific and different measure to score the individual objects (time/point data) by anomalousness. As such, the ensemble embodies heterogeneous detectors. As motivated earlier, Step 2 selects a subset of the detector results to assemble through a proposed selection strategy. Step 3 then combines the selected results into a consensus. Besides several different anomaly detection algorithms, there also exist various different consensus finding approaches. In spirit of building ensembles, SELECT also leverages (seven) different consensus techniques to create intermediate aggregate results. Similar to Step 2, Step 4 then selects a subset of the consensus

results to assemble. Finally, Step 5 combines this subset of results into the final rank list of objects using inverse rank aggregation (Section 2.3.3).

Algorithm 1: SELECT

Input: Data: graph sequence $\{G_1, \ldots, G_t, \ldots, G_T\}$
Output: Rank list of objects (time/point data) by anomaly
1: Obtain results from (5) base detectors
2: Select set E of detectors to assemble
3: Combine E by (7) consensus techniques
4: Select set C of consensus results to assemble
5: Combine C into final rank list

Different from prior works, (*i*) SELECT is a *two-phase* ensemble that not only leverages multiple detectors but also multiple consensus techniques, and (*ii*) it employs novel strategies to carefully select the ensemble components to assemble without any supervision, which outperform naive (no selection) and diversity-based selection (Section 2.5). Moreover, (*iii*) SELECT is the first ensemble method for event detection in temporal graphs, although the same general framework as presented in Algorithm 1 can be deployed for other anomaly mining tasks, e.g. outlier detection, where the base detectors are replaced with a set of algorithms for the particular task at hand. As such we also utilize SELECT for building outlier ensemble with multi-dimensional point data.

Next we fill in the details on the three main components of the proposed SELECT ensemble. In particular, we describe the base detectors (Section 2.3.2), consensus techniques (Section 2.3.3), and the selection strategies (Section 2.3.4).

2.3.2 Base Detectors

In this work SELECT employs five base detectors (Algorithm 1, Line 1) in the Anomaly Ensemble. SELECT is a flexible approach, as such one can easily expand the ensemble with other base detectors. There exists various approaches for outlier detection [50] based on different aspects of outliers, or designed for distinct applications which require detection of domain specific outliers. In our work, we are interested about *unsupervised* outlier detection approaches that assign outlierness scores to the individual instances in the data, as such, allow ranking of instances based on outlierness.

There are a number of well known unsupervised approaches, e.g., "distance based" and "density based" methods for outlier detection. Distance based methods [51,52] and its variants are mostly based on k nearest neighbor (kNN) distances

between the instances, trying to find the *global* outliers far from the rest of the data. On the other hand, density based methods [45, 46] and its variants try to find the *local* outliers which are located in a lower density region compared to their k nearest neighbors.

In this work, for outlier ensemble with no-graph settings SELECT employs two distance based approaches (i) AvgKNN (average k nearest neighbor distance of individual instances is used as outlierness score), (ii) LDOF [52], and three density based approaches (iii) LOF [45], (iv) LOCI [46], and (v) LoOP [53]. For brevity we skip the detailed description of these well established outlier detection approaches.

Moreover, there exist various methods for the event detection problem in temporal graphs [36]. SELECT utilizes five base detectors for event detection, e.g., (1) eigen-behavior based event detection (EBED) from our prior work [54], (2) probabilistic time series anomaly detection (PTSAD) we developed recently [34], (3) Streaming Pattern DIscoveRy in multIple Time-Series (SPIRIT) by Papadimitriou *et al.* [55], (4) anomalous subspace based event detection (ASED) by Lakhina *et al.* [56], and (5) moving-average based event detection (MAED).

Event detection methods extract graph-centric features (e.g., degree) for all nodes over time and detect events in multi-variate time series. We provide brief descriptions of the methods in the following subsections.

Eigen Behavior based Event Detection (EBED)

The multi-variate time series contain the feature values of each node over time and can be represented as a $n \times t$ data matrix, for n nodes and t time points. EBED [54] defines sliding time windows of length w over the series and computes the principal left singular vector of each $n \times w$ matrix W. This vector is the same as the principal eigenvector of WW^T and is always positive due to the Perron-Frobenius theorem [57]. Each eigenvector $u(t)$ is treated as the "eigen-behavior" of the system during time window t, the entries of which are interpreted as the "activity" of each node.

To score the time points, EBED computes the similarity between eigen-behavior $u(t)$ and a summary of past eigen-behaviors $r(t)$, where $r(t)$ is the arithmetic average of $u(t')$'s for $t' < t$. The anomalousness score of time point t is then $Z = 1 - u(t) \cdot r(t) \in [0, 1]$, where high value of Z indicates a change point. For each anomalous time point $\bar{t}$, EBED performs attribution by computing the relative change $\frac{|u_i(\bar{t}) - r_i(\bar{t})|}{u_i(\bar{t})}$ of each node i at $\bar{t}$. The higher the relative change, the more anomalous the node is.

Probabilistic Time Series Anomaly Detection (PTSAD)

A common approach to time series anomaly detection is to probabilistically model a given series and detect anomalous time points based on their likelihood under the model. PTSAD models each series with four different parametric models and performs model selection to identify the best fit for each series. Our first model is the *Poisson*, which is used often for fitting count data. However, Poisson is not sufficient for sparse series with many zeros. Since real-world data is frequently characterized by over-dispersion and excess number of zeros, we employ a second model called *Zero-Inflated Poisson* (ZIP) [58] to account for data sparsity.

We further look for simpler models which fit data with many zeros and employ the *Hurdle models* [59]. Rather than using a single but complex distribution, Hurdle models assume that the data is generated by two simple, separate processes; (i) the hurdle and (ii) the count processes. The hurdle process determines whether there exists activity at a given time point and in case of activity the count process determines the actual (positive) counts. For the hurdle process, we employ two different models. First is the independent *Bernoulli* and the second is the first order *Markov* model which better captures the dependencies, where an activity influences the probability of subsequent activities. For the count process, we use the *Zero-Truncated Poisson* (ZTP) [60].

Overall we model each time series with four different models: Poisson, ZIP, Bernoulli+ZTP and Markov+ZTP. We then employ Vuong's likelihood ratio test [61] to select the best model for individual series. Note that the best-fit model for each series may be different.

To score the time points, we perform a single-sided test to compute a p-value for each value x in a given series; i.e., $P(X \geq x) = 1 - cdf_H(x) + pdf_H(x)$, where H is the best-fit model for the series. The lower the p-value, the more anomalous the time point is. We then aggregate all the p-values from all the series per time point by taking the normalized sum of the p-values and inverting them to obtain scores $\in [0, 1]$ (s.t. higher is more anomalous). For each anomalous time point $\bar{t}$, attribution is done by sorting the nodes (i.e., the series) based on their p-values at $\bar{t}$.

Streaming Pattern DIscoveRy in multIple Time-Series (SPIRIT)

SPIRIT [55] can incrementally capture correlations, discover trends, and dynamically detect change points in multi-variate time series. The main idea is to represent the underlying trends of a large number of numerical streams with a few hidden variables, where the hidden variables are the *projections* of the observed streams onto

the principal direction vectors (eigenvectors). These discovered trends are exploited for detecting change points in the series.

The algorithm starts with a specific number of hidden variables that capture the main trends of the data. Whenever the main trends change, new hidden variables are introduced or several of existing ones are discarded to capture the change. SPIRIT can further quantify the change in the individual time series for attribution through their *participation weights*, which are the entries in the principal direction vectors. For further details on the algorithm, we refer the reader to the original paper by Papadimitriou *et al.* [55].

Anomalous Subspace based Event Detection (ASED)

ASED [56] is based on the separation of high-dimensional space occupied by the time series into two disjoint subspaces, the normal and the anomalous subspaces. Principal Component Analysis is used to separate the high-dimensional space, where the major principal components capture the most variance of the data and hence, construct the normal subspace and the minor principal components capture the anomalous subspace. The projection of the time series data onto these two subspaces reflect the normal and anomalous behavior. To score the time points, ASED uses the *squared prediction error* (SPE) of the residuals in the anomalous subspace. The residual values associated with individual series at the anomalous time points are used to measure the anomalousness of nodes for attribution. For the specifics of the algorithm, we refer to the original paper by Lakhina *et al.* [56].

Moving Average based Event Detection (MAED)

MAED is a simple approach that calculates the moving average μ_t and the moving standard deviation σ_t of each time series corresponding to each node by extending the time window one point at a time. If the value at a specific time point is more than three moving standard deviations away from the mean, then the point is considered as anomalous and assigned a non-zero score. The anomalousness score is the difference between the original value and $(\mu_t + 3\sigma_t)$ at t. To score the time points collectively, MAED aggregates their scores across all the series. For each anomalous time point $\bar{t}$, attribution is done by sorting the nodes (i.e., the series) based on the individual scores they assign to $\bar{t}$.

2.3.3 Consensus Finding

Our ensemble consists of heterogeneous detectors. That is, the detectors employ different anomaly scoring functions and hence their scores may vary in range and interpretation (see Figure 2.1). Unifying these various outputs to find a consensus among detectors is an essential step toward building an ensemble.

A number of different consensus finding approaches have been proposed in the literature, which can be categorized into two, as rank based and score based aggregation methods. Without choosing one over the other, we utilize seven well-established methods as we describe below.

Rank based consensus. Rank based methods use the anomaly scores to order the data points (time points for event detection) into a rank list. This ranking makes the algorithm outputs comparable and facilitates combining them. Merging multiple rank lists into a single ranking is known as rank aggregation, which has a rich history in theory of social choice [62] and information retrieval [63]. SELECT employs three rank based consensus methods. The first method is *Kemeny-Young* [64] which is a voting technique that uses preferential ballot and pair-wise comparison counts to combine multiple rank lists, in which the detectors are treated as voters and the points as the candidates they vote for. The second method *Robust Rank Aggregation* (RRA) [65] utilizes order statistics to compute the probability that a given ordering of ranks for a point across detectors is generated by the null model where the ranks are sampled from a uniform distribution. The final ranking is done based on this probability, where more anomalous points receive a lower probability. The steps of *Robust Rank Aggregation* are given in Algorithm 2.

Given a set of anomaly rank lists R, we first calculate the normalized rank of each data point by dividing its rank by the length of the rank list. For each data point, we get the normalized rank vector ($\mathbf{r} \in sR$) $\mathbf{r} = [r_{(1)}, \ldots, r_{(m)}]$, such that $r_{(1)} \leq \ldots \leq r_{(m)}$, where $r_{(l)}$ denotes the normalized rank of a data point in list $l \in R$. We also store and return the sorted indices in S_{sort} (for its further use in SELECT described in Section 2.3.4). We then compute the order statistics based on this sorted normalized rank vectors to calculate the final aggregated rank list. Specifically, for each ordered list l in a given $\mathbf{r}$, we compute how probable it is to obtain $\hat{r}_{(l)} \leq r_{(l)}$ when the ranks ($\hat{r}$) are generated from a uniform null distribution. This probability ($p_{l,m}(\mathbf{r})$) can be expressed as a binomial probability (in step 17) since at least l normalized rankings drawn uniformly from $[0, 1]$ must be in the range $[0, r_{(l)}]$. The accurate lists rank the anomalies at the top, and hence yield low normalized ranks $r_{(l)}$, so the probability is expected to drop with the ordering ($l \in 1, \ldots, m$). We also store and return these probabilities in $pVals$ (for its further use in SELECT described in Section 2.3.4). As the number of accurate detectors is not known, we define the final score $\rho(\mathbf{r})$ in step 20 for the normalized rank vector

Algorithm 2: RobustRankAggregation

Input: $R :=$ set of anomaly rank lists, $O :=$ target anomalies
Output: $fR :=$ aggregated final rank list
$pVals :=$ probability matrix for normalized rank vectors
$S_{sort} :=$ sorted index matrix for normalized rank vector

1: $nR := \emptyset$, $m = length(R)$/*total item in rank list*/
2: /* calculate normalized rank vector */
3: **for each column** $l \in R$ **do**
4: /* Rank() finds the rank of items in the rank list l */
5: $nR := nR \cup Rank(l)/m$
6: **end for**
7: $sR := \emptyset$, $S_{sort} := \emptyset$
8: **for each row** $l \in nR$ **do**
9: $[sl, ind] = sort(l)$/* ascending order */
10: $sR := sR \cup sl$
11: $S_{sort} = S_{sort} \cup ind$
12: **end for**
13: $pVals := \emptyset$
14: **for each row** $\mathbf{r} \in sR$ **do**
15: $\beta = zeros(1, m)$
16: **for** $l := 1 \ldots m$ **do**
17: $p_{l,m}(\mathbf{r}) := \sum_{t=l}^{m} \binom{m}{t} r_{(l)}^{t} (1 - r_{(l)})^{m-t}$
18: $\beta(1, l) := \beta(1, l) + p_{l,m}(\mathbf{r})$
19: **end for**
20: $\rho(\mathbf{r}) = min(beta)$
21: $pVals := pVals \cup \beta$
22: **end for**
23: $fR = sort(\rho)$/*ascending order*/
24: **if** $O \neq \emptyset$ **then**
25: $S_{sort} := S_{sort}(O, :)$
26: $pVals := pVals(O, :)$
27: **end if**
28: **return** fR, S_{sort}, $pVals$

$\mathbf{r}$ as the minimum of p-values. Finally, we order the data points in fR according to this ρ values, where lower values means more anomalous.

The third approach is based on *Inverse Rank* aggregation, in which we score each point by $\frac{1}{r_i}$ where r_i denotes its rank by detector i and average these scores across detectors based on which we sort the points into a final rank list.

Score based consensus. Rank-based aggregation provides a crude ordering of the data points, as it ignores the actual anomaly scores and their spacing. For instance, quite different rankings can yield equal performance in binary decision. Score-based aggregation approaches tackle the calibration of different anomaly scores and unify them within a shared range. SELECT employs two score based consensus methods. *Mixture Modeling* [28] converts the anomaly scores into probabilities by modeling them as sampled from a mixture of exponential (for inliers) and Gaussian (for outliers) distributions. We use an expectation maximization (EM) algorithm to minimize the negative log likelihood function of the mixture model to estimate the parameters. We calculate the final posterior probability with Bayes rule which represents the probability of anomalousness of the data points. *Mixture Modeling* also provides a binary decision (*class*) for the data points, where point with probability greater than 0.5 gets class 1 (for outliers) and 0 (for inliers) otherwise. *Unification* [29] also converts the scores into probability estimates through regularization, normalization, and Gaussian scaling steps. The probabilities are then comparable across detectors, which we aggregate by both max and avg. This yields four score based methods.

2.3.4 Methodology

Given different base detectors and various consensus methods, the final task remains to utilize them under a unified ensemble framework. In this section, we discuss our proposed approach for building anomaly ensembles. As motivated earlier in Section 2.2.2, carefully selecting which detectors to assemble in Step 2 may help prevent the final ensemble from going astray, provided that some base detectors may fail to reliably identify the anomalies of interest to a given application. Similarly, pruning away consensus results that may be noisy in Step 4 could help reach a stronger final consensus. In anomaly mining, however, it is challenging to identify the components with inferior results given the lack of ground truth to estimate their generalization errors externally. In this section, we present two orthogonal selection strategies that leverage internal clues across detectors or consensuses and work in a fully unsupervised fashion: (i) a vertical strategy that exploits correlations among the results, and (ii) a horizontal strategy that uses order statistics to filter out far-off results.

Strategy I: Vertical Selection

Our first approach to selecting the ensemble components is through correlation analysis among the score lists from different methods, based on which we successively enhance the ensemble one list at a time (hence vertical). The work flow of the

vertical selection strategy is given in Algorithm 3.

Algorithm 3: Vertical Selection

Input: $S :=$ set of anomaly score lists
Output: $E :=$ ensemble set of selected lists

1: $P := \emptyset$
2: /* convert scores to probability estimates */
3: **for each** $s \in S$ **do**
4: $\quad P := P \cup Unification(s)$
5: **end for**
6: $target := avg(P)$;;; /*target vector*/
7: $r :=$ ranklist after sorting $target$ in descending order
8: $E := \emptyset$
9: sort P by weighted Pearson (wP) correlation to $target$
10: /* in descending order, weights:$\frac{1}{r}$ */
11: $l := fetchFirst(P)$, ;
;
$E := E \cup l$
12: **while** $P \neq \emptyset$ **do**
13: $\quad p := avg(E)$;
;
/*current prediction of E*/
14: $\quad$ sort P by wP correlation to p ;
/*descending order*/
15: $\quad l := fetchFirst(P)$
16: $\quad$ **if** $wP(avg(E \cup l), target) > wP(p, target)$ **then**
17: $\quad\quad E := E \cup l$;;; /*select list*/
18: $\quad$ **end if**
19: **end while**
20: **return** E

Given a set of anomaly score lists S, we first unify the scores by converting them to probability estimates using *Unification* [29]. Then we average the probability scores across lists to construct a $target$ vector, which we treat as the "pseudo ground-truth" (Lines 1-6).

We initialize the ensemble E with the list $l \in S$ that has the highest weighted Pearson correlation to $target$. In computing the correlation, the weights we use for the list elements are equal to $\frac{1}{r}$, where r is the rank of an element in $target$ when sorted in descending order, i.e., the more anomalous elements receive higher weight (Lines 7-11).

Next we sort the remaining lists $S\backslash l$ in descending order by their correlation to the current "prediction" of the ensemble, which is defined as the average probability of lists in the ensemble. We test whether adding the top list to the ensemble would increase the correlation of the prediction to *target*. If the correlation improves by this addition, we update the ensemble and reorder the remaining lists by their correlation to the updated prediction, otherwise we discard the list. As such, a list gets either included or discarded at each iteration until all lists are processed (Lines 12-19).

Strategy II: Horizontal Selection

We are interested in finding the data points that are ranked high in a set of accurate rank lists (from either base detectors or consensus methods), ignoring a (small) fraction of inaccurate rank lists. Thus, we also present an element-based (hence horizontal) approach for selecting ensemble components.

To identify the accurate lists, this strategy focuses on the anomalous elements. It assumes that the normalized ranks (defined in Section 2.3.3) of the anomalies should come from a distribution skewed toward zero as the accurate lists are considered to have the anomalies at high positions. Based on this, lists in which the anomalies are not ranked sufficiently high (i.e., have large normalized ranks) are considered to be inaccurate and voted for being discarded. The work flow of the horizontal selection strategy is given in Algorithm 4.

Similar to the vertical strategy we first identify a "pseudo ground truth", in this case a list of anomalies. In particular, we use *Mixture Modeling* [28] to convert each score list in S into probability estimates by modeling them as sampled from a mixture of exponential (for inliers) and Gaussian (for outliers) distributions. We then generate binary lists from the probability estimates in which outliers are denoted by 1 (for probabilities > 0.5), and inliers by 0 (otherwise). We then employ majority voting across these binary lists to obtain a final set of target anomalies O (Lines 1-7).

Given that S contains m lists, we construct a normalized rank vector $\mathbf{r} = [r_{(1)}, \ldots, r_{(m)}]$ for each anomaly $o \in O$, such that $r_{(1)} \leq \ldots \leq r_{(m)}$, where $r_{(l)}$ denotes the rank of o in list $l \in S$ normalized by the total number of elements in l. Following similar ideas to *Robust Rank Aggregation* [65] (in Section 2.3.3), we then compute order statistics based on these sorted normalized rank lists to identify the lists (inaccurate ones) that provide statistically large ranks for each anomaly.

Specifically, for each ordered list l in a given $\mathbf{r}$, we compute how probable it is to obtain $\hat{r}_{(l)} \leq r_{(l)}$ when the ranks $\hat{r}$ are generated by a uniform null distribution. We denote the probability that $\hat{r}_{(l)} \leq r_{(l)}$ by $p_{l,m}(\mathbf{r})$. Under the uniform null model,

Algorithm 4: Horizontal Selection

Input: $S :=$ set of anomaly score lists
Output: $E :=$ ensemble set of selected lists

1: $M := \emptyset$,;
   $R := \emptyset$,;
   $F := \emptyset$,;
   $E := \emptyset$
2: **for each** $l \in S$ **do**
3: /* label score lists with 1 (outliers) & 0 (inliers) */
4: $class := MixtureModel(l)$, ;
   ;
   $M := M \cup class$
5: $R := R \cup ranklist(l)$
6: **end for**
7: $O := majorityVoting(M)$;;; /*target anomalies*/
8: $[S_{sort}, pVals] := RobustRankAggregation(R, O)$
9: **for each** $o \in O$ **do**
10: $m_{ind} := \min(pVals(o, :))$
11: $F := F \cup S_{sort}(o, (m_{ind} + 1) : end)$
12: **end for**
13: **for each** $l \in S$ **do**
14: $count :=$ number of occurrences of l in F
15: **end for**
16: Cluster non-zero *count*s into two clusters, C_l and C_h
17: $E := S \setminus \{s \in C_h\}$;
/* discard high-*count* lists */
18: **return** E

the probability that $\hat{r}(l)$ is smaller or equal to $r_{(l)}$ can be expressed as a binomial probability since at least l normalized rankings drawn uniformly from $[0, 1]$ must be in the range $[0, r_{(l)}]$.

$$p_{l,m}(\mathbf{r}) = \sum_{t=l}^{m} \binom{m}{t} r_{(l)}^{t} (1 - r_{(l)})^{m-t},$$

For a sequence of accurate lists that rank the anomalies at the top, and hence that yield low normalized ranks $r_{(l)}$, this probability is expected to drop with the ordering, i.e., for increasing $l \in \{1 \dots m\}$. As with increasing ordering the probability of drawing more normalized ranks uniformly from $[0, 1]$ to be in a small range $[0, r_{(l)}]$ gets small. An example sequence of p probabilities (y-axis) are shown in Figure 2.2

for an anomaly based on 20 score lists. The lists are sorted by their normalized ranks of the anomaly on the x-axis. The figure suggests that the 5 lists at the end of the ordering are likely inaccurate, as the ranks of the given anomaly in those lists are larger than what is expected based on the ranks in the other lists.

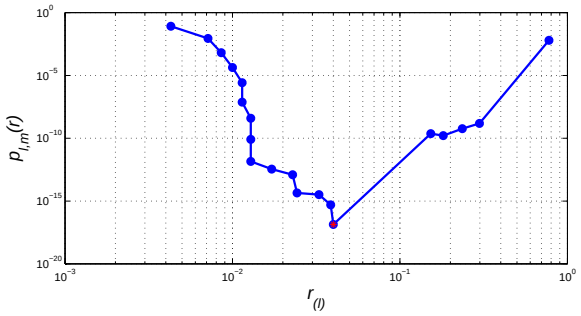


Figure 2.2: Normalized rank $r_{(l)}$ vs. probability p that $\hat{r}_{(l)} \leq r_{(l)}$, where $\hat{r}$ are drawn uniformly at random from $[0, 1]$.

Based on this intuition, we count the frequency that each list l is ordered *after* the list with $\min_{l=1,\ldots,m} p_{l,m}(\mathbf{r})$ among all the normalized rank lists $\mathbf{r}$ of the target anomalies (Lines 8-15). We then group these counts into two clusters[1] and discard the lists in the cluster with the higher average count (Lines 16-17). This way we eliminate the lists with larger counts, but retain the lists that appear inaccurate only a few times which may be a result of the inherent uncertainty or noise in which we construct the target anomaly set.

2.3.5 Existing/Alternative Ensemble Learning Approaches

In this section, we discuss three alternative existing approaches for building anomaly ensembles, which differ in whether and how they select their ensemble components. We compare to these methods in the experiments (Section 2.5).

Full ensemble

The full ensemble [34] selects all the detector results (Step 2 of Alg.1) and later all the consensus results (Step 4 of Alg.1) to aggregate at both phases of SELECT. As such, it is a naive approach that is prone to obtain inferior results in the presence of inaccurate detectors.

[1]We cluster the counts by k-means clustering with $k = 2$, where the centroids are initialized with the smallest and largest counts, respectively.

Diversity-based ensemble

In classification, two basic conditions for an ensemble to improve over the constituent classifiers are that the base classifiers are (i) accurate (better than random), and (ii) diverse (making uncorrelated errors) [8,11]. Achieving better-than-random accuracy in supervised learning is not hard, and several studies have shown that ensembles tend to yield better results when there is a significant diversity among the models [66, 67].

Following on these insights, Schubert *et al.* proposed a diversity-based ensemble [32], which is similar to our vertical selection in Alg. 3. The main distinction is the ascending ordering in Lines 9 and 14, which yields a diversity-favored, in contrast to a correlation-favored, selection.[2]

Unlike classification ensembles, however, it is not realistic for anomaly ensembles to assume that all the detectors will be reasonably accurate (i.e., better than random), as some may fail to spot the (type of) anomalies in the given data. In the existence of inaccurate detectors, the diversity-based approach would likely yield inferior results as it is prone to selecting inaccurate detectors for the sake of diversity. As we show in our experiments, too much diversity is in fact bound to limit accuracy for event and outlier detection ensembles.

Unsupervised Learning Algorithm for Rank Aggregation (ULARA)

In selective ensemble approaches, base detectors and consensus approaches are selected in an unsupervised way to generate the final result. In doing so the algorithms which are not selected are discarded and do not contribute to the final result. An alternative way to using binary selection criteria is estimating weights for detectors/consensus results and applying a weighted rank aggregation technique to combine the results. [35] proposed an unsupervised learning algorithm (called ULARA) for this kind of rank aggregation, which adaptively learns a parameterized linear combination of ranklists to optimize the relative influence of individual detectors on the final ranking by learning relative weights w_i for the individual ranklists (where, $\sum_{i=1}^{n} w_i = 1$). Their approach is guided by the principle that the relative contribution of an individual ranklist to the final ranking should be determined by its tendency to agree with other ranklists in the pool. Those ranklists that agree with the majority are given large relative weights and those that disagree are given small relative weights. Agreement is measured by the total variance from the average ranking of individual data points. As a result, the goal is to assign weights such

[2]There are other differences between our vertical selection (Algorithm 3) and the diversity-based ensemble in [32], such as the construction of the pseudo ground truth and the choice of weights in correlation computation.

that the total weighted variance is minimized. ULARA has two different ways to estimate the detector weights, one based on additive and another based on exponential weight updates. In evaluation, we experiment with both of them and report the better performance for each dataset.

2.4 Theoretical Foundations

In this section we present the theoretical underpinnings of our proposed anomaly ensemble. Although, classification and anomaly detection problems are significantly different, the theoretical foundation of both the problems can be explained with bias-variance tradeoff. We explain the theoretical analysis for our anomaly ensemble in light of the theoretical foundations provided by Aggarwal *et al.* [68] for outlier ensemble in terms of well known ideas from classification. In Section 2.4.1 we describe the bias-variance trade-off for anomaly detection and in Section 2.4.2 we present evidence for error reduction by reducing bias-variance for SELECT.

2.4.1 Bias-Variance Tradeoff in Anomaly Detection

The bias-variance tradeoff is often explained in the context of supervised learning, e.g., classification, as quantification of bias-variance requires labeled data. Although, anomaly detection problems lack the existence of ground truth (hence solved using unsupervised approaches), this bias-variance tradeoff can be quantified by treating the dependent variable (actual labels) as unobserved.

Unlike classification, most anomaly detection algorithms output "anomalousness" scores for the data points. We can consider these anomaly detection algorithms as two class classification problems having a majority class (normal points) and a rare class (anomalous points) by converting the anomalousness scores to class labels. The points which achieve scores above a threshold are considered as anomalies and get label 1 (label 0 for normal points below threshold). Deciding this threshold is a difficult task for heterogeneous detectors as they provide scores with different scaling and in different ranges. As such there exist unification approaches [28, 53] which convert these anomalousness scores to probability estimates to make them comparable with out changing the ranking of the data points.

Now that unsupervised anomaly detection problem looks similar like a classification problem with only unobserved actual labels, we can explain the bias-variance tradeoff for anomaly detection using ideas from classification. The expected error of anomaly detection can be split into two main components reducible error and irreducible error (i.e., noise). This reducible error can be minimized to maximize

the accuracy of the detector. Furthermore, the reducible error can be decomposed into (i) error due to squared bias, and (ii) error due to variance. However, there is a tradeoff while minimizing both these sources of errors.

Bias of a detector is the amount by which the expected output of the detector differs from the true unobserved value, over the training data. On the other hand, variance of a detector is the amount by which the output of a detector over one training set differs from the expected output of the detector over all the training sets. The tradeoff between bias and variance can be viewed as, (i) a detector which has low bias is very flexible in fitting data well and it will fit each training set differently providing high variance, and (ii) inflexible detectors will have low variance and might provide high bias. Our goal is to improve the accuracy as much as possible by reducing both bias and variance using selective anomaly ensemble approach SELECT.

2.4.2 Bias-Variance Reduction in Anomaly Ensemble

Most classification ensemble generalize directly to anomaly ensemble for variance reduction, but controlled bias reduction is rather difficult due to lack of ground truth. It is evident from classification ensemble literature that combining results from multiple heterogeneous base algorithms will decrease the overall variance of the ensemble [68] which is also true for anomaly ensemble. On the other hand, this combination does not provide enough ground for reducing bias in anomaly ensemble. Moreover, our SELECT approach is designed based on the assumption that, there exist inaccurate detectors which are able to hurt the overall ensemble if combined with the accurate ones.

In this work, we present two selective approaches SelectV and SelectH which discard these inaccurate detectors. For both the algorithms we utilize pseudo-ground truth which can be viewed as a low-bias output because it averages the outputs for SelectV and takes majority voting for SelectH across the diverse detectors, each of which might have biases in different directions. By eliminating the detectors which do not agree with the pseudo-ground truth for SelectV and provide inaccurate ranking (compared to others) of the target anomalies (pseudo-ground truth) in SelectH, we are effectively eliminating the detectors which have high bias. Furthermore, we are using SELECT in two phases to reduce the bias. Therefore, by carefully selecting detectors and combining their outputs in two phases we are reducing both bias and variance, and thus improving accuracy.

The reason why the state-of-the-art approaches (Full, DivE, ULARA) might fail to achieve better accuracy than SELECT can be described with bias-variance reduction. Full combines all the base detectors results including the ones with high bias and thus hurt the final ensemble. Although ULARA calculates relative weights based

on the agreement between the detectors, it fails to totally discard the ones with high bias. For the sake of diversity DivE selects the more diverse detectors and thus end up selecting the ones with high bias, reducing the overall accuracy.

In selecting the detectors, SelectV utilizes the correlation between the ranklists provided by the base detectors. Therefore, SelectV considers all the data points to decide which detectors to select and thus affected by the majority inliers class. On the other hand, SelectH considers only the target anomalies to decide which detectors to select, as in this approach we emphasize on the anomalies being misclassified by the detectors. As a result, SelectV is sometimes prone to discarding accurate detectors and selecting inaccurate ones. Section 2.5 provides results justifying the above explanation.

We consider our SELECT approach to be a heuristic one as it is not guaranteed to provide optimal solution for different datasets. As such it can behave unpredictably for pathological datasets (see Section 2.5). Bias reduction in unsupervised learning, e.g., anomaly detection, is a hard problem and using heuristic method is quite reasonable to improve accuracy by achieving immediate goals.

2.5 Evaluation

We evaluate our selective ensemble approach on the event detection problem using five real-world datasets, both previously used as well as newly collected by us, including email communications, news corpora, and social media. For four of these datasets we compiled ground truths for the temporal anomalies, for which we present quantitative results. We use the remaining data for illustrating case studies. Furthermore, we evaluate SELECT on the outlier detection problem using seven real-world datasets from UCI machine learning repository.[3]

We compare the performance of SELECT with vertical selection (SelectV), and horizontal selection (SelectH) to that of individual detectors, the full ensemble with no selection (Full), the diversity-based ensemble (DivE) by [32], and weighted ensemble approach (ULARA) by [35]. This makes ours one of the few works that quantitatively compares and contrasts anomaly ensembles at a scale that includes as many datasets with ground truth.

In a nutshell, our results illustrate that (*i*) base detectors do not always all produce accurate results, (*ii*) ensemble approach alleviates the shortcomings of the inaccurate detectors, (*iii*) a careful selection of ensemble components increases the overall performance, and (*iv*) introducing noisy results decreases overall ensemble accuracy where the diversity-based ensemble is affected the most.

[3]http://archive.ics.uci.edu/ml/datasets.html

2.5.1 Dataset Description

Temporal Graph Datasets

In the following we describe the five real-world temporal graph datasets and one simulated cyber network dataset (Dataset 3) we used in this work. Our datasets are collected from various domains. Five datasets contain ground truth events, and the last dataset is used for illustrating case studies. All our datasets can be found at `http://shebuti.com/SelectiveAnomalyEnsemble/`, where we also share the source code for `SELECT`.

Dataset 1: EnronInc. We use four years (1999–2002) of Enron email communications. In the temporal graphs, the nodes represent email addresses and directed edges depict sent/received relations. Enron email network contains a total of $80,884$ nodes. We analyze the data with daily sample rate skipping the weekends (700 time points). The ground truth captures the major events in the company's history, such as CEO changes, revenue losses, restatements of earnings, etc.

Dataset 2: RealityMining Reality Mining is comprised of communication and proximity data of 97 faculty, student, and staff at MIT recorded continuously via pre-installed software on their mobile devices over 50 weeks [69]. From the raw data we built sequences of weekly temporal graphs for three types of relations; voice calls, short messages, and bluetooth scans. For voice call and short message graphs a directed edge denotes an incoming/outgoing call or message, and for bluetooth graphs an edge depicts physical proximity between two subjects. The ground truth captures semester breaks, exam and sponsor weeks, and holidays.

Dataset 3: CyberChallenge This data, provided by NGAS R&D Park, contains IP-IP network traffic flow over ten days between 125 hosts. We analyze this data using 10 minute sample rate to build our temporal graph sequence (1304 time points). The ground truth contains three major changes in the state of the network, and a total of three associated IPs which we use in evaluating characterization performance.

Dataset 4: TwitterSecurity We collect tweet samples using the Twitter Streaming API for four months (May 12–Aug 1, 2014). We filter the tweets containing Department of Homeland Security keywords related to terrorism or domestic security.[4] After named entity extraction and resolution (including URLs, hashtags, @ mentions), we build entity-entity co-mention temporal graphs on daily basis (80 time ticks). We compile the ground truth to include major world news of 2014, such as the Turkey mine accident, Boko Haram kidnapping school girls, killings during Yemen raids, etc.

[4]http://www.huffingtonpost.com/2012/02/24/homeland-security-manual_n_1299908.html

Dataset 5: TwitterWorldCup Our Twitter collection also spans the World Cup 2014 season (June 12–July 13). This time, we filter the tweets by popular/official World Cup hashtags, such as `#worldcup`, `#fifa`, `#brazil`, etc. Similar to TwitterSecurity, we construct entity-entity co-mention temporal graphs on 5 minute sample rate (8640 time points). The ground truth contains the goals, penalties, and injuries in all the matches that involve at least one of the renowned teams (specifically, at least one of Brazil, Germany, Argentina, Netherlands, Spain, France).

Dataset 6: NYTNews This corpus contains all of the published articles in New York Times over 7.5 years (Jan 2000–July 2007) (available from https://catalog.ldc.upenn.edu/LDC2008T19). The named entities (people, places, organizations) are hand-annotated by human editors. We construct weekly temporal graphs (390 time points) in which each node corresponds to a named entity and edges depict co-mention relations in the articles. The data contains around $320,000$ entities, however no ground truth events.

Table 2.1: Summary of multi-dimensional point datasets

dataset	Instances	Attributes	% of outliers
WBC	378	30	5.6
Glass	214	9	4.2
Lympho	148	18	4.1
Cardio	1831	21	9.6
Musk	3062	166	3.2
Thyroid	3772	6	2.5
Letter	1600	32	6.25

Multi-dimensional point Datasets

In Table 2.1 we provide the summary of seven real-world datasets that we utilize in outlier ensemble from UCI Machine Learning Repository [70]. Originally, these are classification datasets, as such, further preprocessing was required to adapt them in outlier detection problem having a rare class (outliers) and a majority class (inliers). Appendix A contains the detailed description of these datasets and all of them are available at http://odds.cs.stonybrook.edu/#table1. WBC, Glass, Lympho, Cardio, Musk and Thyroid datasets are used in [68] and the Letter dataset is introduced in [71].

Table 2.2: Significance of accuracy results compared to random ensembles with same number of selected components as SELECTfor event detection. The accuracy of the alterntive approaches (Full, DivE, and ULARA) are also given in parentheses. All the approaches presented here incorporate *10 base components*. These results show that (i) SELECT is superior to existing methods, and (ii) it selects significantly more important (i.e., accurate) detectors to combine.

	Accuracy	significance
EnronInc. (10 comp.) (Full: 0.7082, DivE: 0.6276, ULARA: 0.3652)		
(i) RandE (3/10, 3/7)	0.4804 (μ)	0.1757 (σ)
SelectV	0.7125	$= \mu + 1.3210\sigma$
(ii) RandE (5/10, 6/7)	0.5509 (μ)	0.1406 (σ)
SelectH	**0.7920**	$= \mu + 1.7148\sigma$
RM-VoiceCall (10 comp.) (Full: 0.7302, DivE: 0.8724, ULARA: 0.8125)		
(i) RandE (2/10, 1/7)	0.7370 (μ)	0.1551 (σ)
SelectV	0.8370	$= \mu + 0.6447\sigma$
(ii) RandE (8/10, 6/7)	0.7653 (μ)	0.0714 (σ)
SelectH	**0.9045**	$= \mu + 1.9496\sigma$
RM-Bluetooth (10 comp.) (Full: 0.8398, DivE: 0.7735, ULARA: 0.8437)		
(i) RandE (4/10, 1/7)	0.8269 (μ)	0.1129 (σ)
SelectV	**0.9193**	$= \mu + 0.8184\sigma$
(ii) RandE (8/10, 6/7)	0.8410 (μ)	0.0322 (σ)
SelectH	0.8886	$= \mu + 1.4783\sigma$
RM-SMS (10 comp.) (Full: 0.9092, DivE: 0.8598, ULARA: 0.7937)		
(i) RandE (4/10, 1/7)	0.8328 (μ)	0.0978 (σ)
SelectV	**0.9283**	$= \mu + 0.9765\sigma$
(ii) RandE (8/10, 6/7)	0.8976 (μ)	0.0620 (σ)
SelectH	0.9217	$= \mu + 0.3887\sigma$
CyberChallenge (10 comp.) (Full: **1.0000**, DivE: **1.0000**, ULARA: 0.9167)		
(i) RandE (6/10, 4/7)	0.8831 (μ)	0.1380 (σ)
SelectV	**1.0000**	$= \mu + 0.8471\sigma$
(ii) RandE (10/10, 6/7)	0.9405 (μ)	0.0407 (σ)
SelectH	**1.0000**	$= \mu + 1.4619\sigma$
TwitterSecurity (10 comp.) (Full: 0.5200, DivE: 0.4800, ULARA: 0.5733)		
(i) RandE (4/10, 1/7)	0.5068 (μ)	0.0755 (σ)
SelectV	0.5467	$= \mu + 0.5285\sigma$
(ii) RandE (9/10, 3/7)	0.5198 (μ)	0.0538 (σ)
SelectH	**0.5867**	$= \mu + 1.2435\sigma$

Table 2.3: Significance of accuracy results compared to random ensembles with same number of selected components as SELECTfor event detection. The accuracy of the alterntive approaches (Full, DivE, and ULARA) are also given in parentheses. All the approaches presented here incorporate *20 base components*. These results show that (i) SELECT is superior to existing methods, and (ii) it selects significantly more important (i.e., accurate) detectors to combine.

	Accuracy	significance
EnronInc. (20 comp.) (Full: 0.5420, DivE: 0.4697, ULARA: 0.2961)		
(i) RandE (4/20, 2/7)	0.4047 (μ)	0.1732 (σ)
SelectV	0.7018	$= \mu + 1.7154\sigma$
(ii) RandE (15/20, 6/7)	0.5707 (μ)	0.0864 (σ)
SelectH	**0.7798**	$= \mu + 2.4201\sigma$
RM-VoiceCall (20 comp.) (Full: 0.8011, DivE: 0.8335, ULARA: 0.8250)		
(i) RandE (2/20, 2/7)	0.7752 (μ)	0.1494 (σ)
SelectV	0.8847	$= \mu + 0.7329\sigma$
(ii) RandE (17/20, 6/7)	0.8187 (μ)	0.0497 (σ)
SelectH	**0.8949**	$= \mu + 1.5332\sigma$
RM-SMS (20 comp.) (Full: 0.9542, DivE: 0.8749, ULARA: 0.7312)		
(i) RandE (2/20, 1/7)	0.7685 (μ)	0.1521 (σ)
SelectV	0.9294	$= \mu + 1.0579\sigma$
(ii) RandE (17/20, 5/7)	0.9217 (μ)	0.0296 (σ)
SelectH	**0.9621**	$= \mu + 1.3649\sigma$
CyberChallenge (20 comp.) (Full: 0.6325, DivE: 0.6667, ULARA: 0.4549)		
(i) RandE (5/20, 7/7)	0.5975 (μ)	0.2599 (σ)
SelectV	0.3575	$= \mu - 0.9234\sigma$
(ii) RandE (20/20 + 6/7)	0.6128 (μ)	0.0881 (σ)
SelectH	**0.7500**	$= \mu + 1.5573\sigma$

2.5.2 Event Detection Performance

Next we quantitatively evaluate the ensemble methods on detection accuracy. The final result output by each ensemble is a rank list, based on which we create the precision-recall (PR) plot for a given ground truth. We report the area under the PR plot, namely *average precision*, as the measure of accuracy.

In Table 2.2 and Table 2.3 we report the summary of results for different datasets containing the average precision values for SELECT and other existing ensemble approaches (Full, DivE, and ULARA) to which we compare SELECT. Here, Table 2.2 represents the approaches with 10 base components and Table 2.3 rep-

resents the approaches with 20 base components. To investigate the significance of the selections made by SELECT ensembles, we compare them to ensembles that randomly select the same number of components to assemble at each phase. In Table 2.2 and Table 2.3 we also report the average and standard deviation of accuracies achieved by 100 such random ensembles, denoted by RandE, and the gain achieved by SelectV and SelectH over their respective random ensembles. We note that SELECT ensembles provide superior results to RandE, Full, DivE, and ULARA. Moreover, SelectH appears to be a better strategy than SelectV, where it either provides the best result (8/10 from both Tables) or achieves comparable accuracy when SelectV is the winner. Selecting results based on diversity turns out to be a poor strategy for anomaly ensembles as DivE yields even worse results than the Full ensemble (6/10 from both Tables). Putting relative weight depending on the agreement between base detector results does not even help according to our evaluation, as such ULARA ensemble yields poor results even worse than DivE (8/10 from both Tables).

Table 2.4: Accuracy of ensembles for EnronInc. (features: weighted in-/out-degree). $*$ depicts selected detector/consensus results.

		Full	DivE	ULARA(w_i)	SelectV	SelectH
Base Algorithms	EBED (win)	0.1313	$*$	▬	$*$	
	PTSAD (win)	0.1462	$*$	▪		
	SPIRIT (win)	0.7032	$*$	▬		$*$
	ASED (win)	0.5470	$*$	▪	$*$	$*$
	MAED (win)	0.6670		▪		$*$
	EBED (wout)	0.2846	$*$	▬		
	PTSAD (wout)	0.2118	$*$	▪		
	SPIRIT (wout)	0.4563	$*$	▬		$*$
	ASED (wout)	0.0580	$*$	▪		
	MAED (wout)	0.7328		▪	$*$	$*$
Consensus	Inverse Rank	$*$ 0.6829	$*$ 0.5660	–	0.6738	$*$ 0.8291
	Kemeny-Young	$*$ 0.4086	$*$ 0.3703	–	$*$ 0.6586	$*$ 0.6334
	RRA	$*$ 0.6178	0.4871	–	0.5686	$*$ 0.6590
	Uni (avg)	$*$ 0.5292	$*$ 0.5511	–	$*$ 0.6375	$*$ 0.6207
	Uni (max)	$*$ 0.3333	$*$ 0.3187	–	0.4314	$*$ 0.7353
	MM (avg)	$*$ 0.7513	$*$ 0.5726	–	$*$ 0.7663	$*$ 0.7530
	MM (max)	$*$ 0.0218	$*$ 0.0218	–	0.2108	0.0224
Final Ensemble		0.7082	0.6276	0.3652	0.7125	**0.7920**

Table 2.4 shows the accuracies for all five ensemble methods on EnronInc., along with the accuracies of the base detectors and consensus methods. In Table 2.4 the black bars for ULARA are the representatives of weights where the length

of the bars are proportional to the relative weights assigned to corresponding detectors, we denote them as *weight bars* for ULARA. Also ULARA is a single phase ensemble approach, as such there is no second phase results. We note that some detectors yield quite low accuracy (e.g., ASED (wout)) on this dataset. Further, MM (max) consensus provides low accuracy across ensembles no matter which detector results are combined. SELECT ensembles successfully filter out relatively inferior results and achieve higher accuracy. SelectV ensemble provides sparser selection than SelectH ensemble, but SelectH provides better accuracy than SelectV, which indicates that SelectV possibly missing some valuable detectors. We also note that ULARA andDivE yield lower performance than all, including Full. The weight bars indicate that ULARA is putting high weights to relatively inaccurate detectors.

We show the final anomaly scores of the time points provided by SelectH on EnronInc. for visual analysis in Figure 2.3. The figure also depicts the ground truth events by vertical (red) lines, which we note to align well with the time points with high scores.

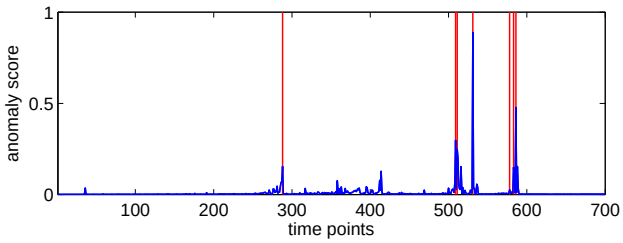


Figure 2.3: Anomaly scores of time points by SelectH on EnronInc. align well with ground truth (vertical red lines).

Table 2.4 shows results when we use weighted node in-/out-degree features on the directed Enron graphs to construct the input time series for the base detectors. As such, the ensembles utilize 10 components in the first phase. We also build the ensembles using 20 components where we include the unweighted in-/out-degree features. Table 2.5 gives all the accuracy results, selections made and weight bars for ULARA, a summary of which is provided in Table 2.3. We notice that the unweighted graph features are less informative and yield lower accuracies across detectors on average. This affects the performance of Full, DivE, and ULARA, where the accuracies drop significantly, specially for ULARA. On the other hand, SELECT ensembles are able to achieve comparable accuracies with increased significance under the additional noisy input.

Thus far, we used the exact time points of the events to compute precision and recall. In practice, some time delay in detecting an event is often tolerable. Therefore, we also compute the detection accuracy when delay is allowed; e.g., for delay 2, detecting an event that occurred at t within time window $[t-2, t+2]$ is

Table 2.5: Accuracy of ensembles for EnronInc. (directed) (20 components) (features: weighted in-/out-degree and unweighted in-/out-degree). $*$ depicts selected detector/consensus results.

		Full	DivE	ULARA(w_i)	SelectV	SelectH
Base Algorithms	EBED (win)	0.1313	$*$			$*$
	PTSAD (win)	0.1462	$*$			$*$
	SPIRIT (win)	0.7032	$*$			$*$
	ASED (win)	0.5470	$*$		$*$	$*$
	MAED (win)	0.6670				$*$
	EBED (wout)	0.2846	$*$			
	PTSAD (wout)	0.2118	$*$			$*$
	SPIRIT (wout)	0.4563				$*$
	ASED (wout)	0.0580	$*$			
	MAED (wout)	0.7328			$*$	$*$
	EBED (uin)	0.0892	$*$			
	PTSAD (uin)	0.1607	$*$			$*$
	SPIRIT (uin)	0.3996	$*$			$*$
	ASED (uin)	0.1395			$*$	$*$
	MAED (uin)	0.4439			$*$	$*$
	EBED (uout)	0.0225	$*$			
	PTSAD (uout)	0.2546				$*$
	SPIRIT (uout)	0.1012	$*$			$*$
	ASED (uout)	0.0870	$*$			
	MAED (uout)	0.4181				$*$
Consensus	Inverse Rank	$*$ 0.7121	$*$ 0.5660	–	0.6577	$*$ 0.7496
	Kemeny-Young	$*$ 0.3033	$*$ 0.2495	–	0.5361	$*$ 0.5066
	RRA	$*$ 0.5948	$*$ 0.5348	–	0.4948	$*$ 0.5774
	Uni (avg)	$*$ 0.4838	$*$ 0.4325	–	$*$ 0.6047	$*$ 0.5336
	Uni (max)	$*$ 0.3020	$*$ 0.2242	–	0.6633	$*$ 0.4280
	MM (avg)	$*$ 0.5673	$*$ 0.4662	–	0.6761	$*$ 0.7217
	MM (max)	$*$ 0.0216	$*$ 0.0216	–	$*$ 0.5355	0.0222
Final Ensemble		0.5420	0.4697	0.2961	0.7018	**0.7798**

counted as accurate. Figure 2.4 shows the accuracy for 0 to 5 time point delays (days) for EnronInc., where delay 0 is the same as exact detection. We notice that SELECT ensembles and Full can detect almost all the events within 5 days before or after each event occurs.

Next we analyze the results for RealityMining. Similar to EnronInc., we build the ensembles using both 10 and 20 components for the directed Voice Call and

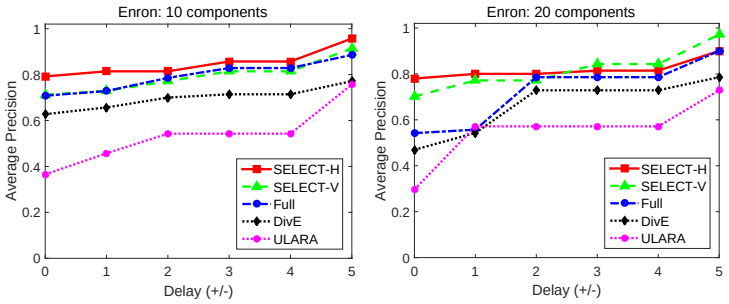


Figure 2.4: EnronInc. average precision vs. detection delay using (left) 10 components and (right) 20 components.

Table 2.6: Accuracy of ensembles for RealityMining Voice Call (directed) (10 components) (features: weighted in-/out-degree). $*$: selected detector/consensus results.

		Full	DivE	ULARA(w_i)	SelectV	SelectH
Base Algorithms	EBED (win)	0.3508	$*$	▬▬		
	PTSAD (win)	0.6284		▬		$*$
	SPIRIT (win)	0.8309	$*$	▬		$*$
	ASED (win)	0.9437		▬	$*$	$*$
	MAED (win)	0.8809	$*$	▬▬		$*$
	EBED (wout)	0.4122	$*$	▬▬		
	PTSAD (wout)	0.6273		▬		$*$
	SPIRIT (wout)	0.7346		▬	$*$	$*$
	ASED (wout)	0.9500		▬		$*$
	MAED (wout)	0.8758		▬▬		$*$
Consensus	Inverse Rank	$*$ 0.7544	0.6169	–	0.8880	$*$ 0.8222
	Kemeny-Young	$*$ 0.8221	$*$ 0.7708	–	0.8619	$*$ 0.9309
	RRA	$*$ 0.8154	0.5936	–	0.8901	$*$ 0.9416
	Uni (avg)	$*$ 0.7798	$*$ 0.6413	–	$*$ 0.8370	$*$ 0.9098
	Uni (max)	$*$ 0.6704	0.5757	–	0.7786	$*$ 0.7833
	MM (avg)	$*$ 0.9190	$*$ 0.9162	–	0.8835	$*$ 0.9183
	MM (max)	$*$ 0.4380	$*$ 0.8934	–	0.7569	0.4380
Final Ensemble		0.7302	0.8724	0.8125	0.8370	**0.9045**

SMS graphs. Bluetooth graphs are undirected, as they capture (symmetric) proximity of devices, for which we build ensembles with 10 components using weighted and unweighted degree features. All the details on detector and consensus accuracies, weight bars for ULARA as well as selections made by DivE, SelectV and SelectH are given in Table 2.6 and Table 2.7 for Voice Call 10 and 20 components, Table 2.8 and Table 2.9 for SMS 20 and 10 components, and Table 2.10 for Bluetooth 10 com-

Table 2.7: Accuracy of ensembles for RealityMining Voice Call (directed) (20 components) (features: weighted in-/out-degree and unweighted in-/out-degree) ∗: selected detector/consensus results.

		Full	DivE	ULARA(w_i)	SelectV	SelectH
Base Algorithms	EBED (win)	0.3508	∗			
	PTSAD (win)	0.6284				∗
	SPIRIT (win)	0.8309			∗	∗
	ASED (win)	0.9437			∗	∗
	MAED (win)	0.8809	∗			∗
	EBED (wout)	0.4122	∗			
	PTSAD (wout)	0.6273				∗
	SPIRIT (wout)	0.7346				∗
	ASED (wout)	0.9500				∗
	MAED (wout)	0.8758				∗
	EBED (uin)	0.4173				
	PTSAD (uin)	0.8636	∗			∗
	SPIRIT (uin)	0.8313				∗
	ASED (uin)	0.9191				∗
	MAED (uin)	0.8706	∗			∗
	EBED (uout)	0.4800				∗
	PTSAD (uout)	0.8665				∗
	SPIRIT (uout)	0.7480				∗
	ASED (uout)	0.9229	∗			∗
	MAED (uout)	0.9115				∗
Consensus	Inverse Rank	∗ 0.8035	0.7952	–	0.9240	∗ 0.8681
	Kemeny-Young	∗ 0.9064	0.9018	–	0.9076	∗ 0.9158
	RRA	∗ 0.8866	∗ 0.7771	–	0.9013	∗ 0.9311
	Uni (avg)	∗ 0.8598	0.9192	–	∗ 0.8448	∗ 0.9102
	Uni (max)	∗ 0.6844	∗ 0.6863	–	0.8517	∗ 0.7611
	MM (avg)	∗ 0.9321	∗ 0.9083	–	∗ 0.8312	∗ 0.9134
	MM (max)	∗ 0.4380	∗ 0.8858	–	0.8015	0.4380
Final Ensemble		0.8011	0.8335	0.8250	0.8847	**0.8949**

ponents. We provide the summary of results in Table 2.2 and Table 2.3. We note that SELECT ensembles provide superior results to Full, DivE, and ULARA. Specifically, for Voice Call and SMS with 20 components SelectH performs better than all other detectors, and for SMS with 10 components and Bluetooth SelectV performs better than all. These tables also show that the accuracy of the final ensemble with SELECT is better than the individual base detectors in most of the cases.

Table 2.8: Accuracy of ensembles for RealityMining SMS (directed) (20 components) (features: weighted in-/out-degree and unweighted in-/out-degree). $*$: selected detector/consensus results.

		Full	DivE	ULARA(w_i)	SelectV	SelectH
Base Algorithms	EBED (win)	0.6117	*	▬		*
	PTSAD (win)	0.7003		▪		*
	SPIRIT (win)	0.9256		▪		*
	ASED (win)	0.6338	*	▬		*
	MAED (win)	0.9002		▬		*
	EBED (wout)	0.5595		▬		*
	PTSAD (wout)	0.7023		▪		
	SPIRIT (wout)	0.8656		▪		*
	ASED (wout)	0.9102		▪		*
	MAED (wout)	0.9259		▬	*	*
	EBED (uin)	0.4407	*	▬		
	PTSAD (uin)	0.7809	*	▪		*
	SPIRIT (uin)	0.7841		▪		*
	ASED (uin)	0.6248	*	▬		*
	MAED (uin)	0.8297	*	▬		*
	EBED (uout)	0.3246		▬		
	PTSAD (uout)	0.9157		▪	*	*
	SPIRIT (uout)	0.8744		▪		*
	ASED (uout)	0.9150		▪		*
	MAED (uout)	0.8005		▬		*
Consensus	Inverse Rank	$*$ 0.9135	0.6751	–	0.9634	$*$ 0.9230
	Kemeny-Young	$*$ 0.9286	$*$ 0.7567	–	0.9094	$*$ 0.9325
	RRA	$*$ 0.9568	0.6465	–	0.9418	$*$ 0.9583
	Uni (avg)	$*$ 0.8791	0.6499	–	$*$ 0.9294	$*$ 0.9156
	Uni (max)	$*$ 0.7173	$*$ 0.6696	–	0.9342	0.8650
	MM (avg)	$*$ 0.9107	$*$ 0.8942	–	0.8519	$*$ 0.9138
	MM (max)	$*$ 0.8895	$*$ 0.8480	–	0.9307	0.8895
Final Ensemble		0.9542	0.8749	0.7312	0.9294	**0.9621**

Figure 2.5 illustrates the accuracy-delay plots which show that SELECT ensembles for Bluetooth and SMS detect almost all the events within a week before or after they occur, while the changes in Voice Call are relatively less reflective of the changes in the school year calendar.

Finally, we present the event detection result using Twitter. Table 2.11 contains accuracy details for detecting world news on TwitterSecurity, a summary of

Table 2.9: Accuracy of ensembles for RealityMining SMS (directed) (10 components) (features: weighted in-/out-degree). $*$: selected detector/consensus results.

		Full	DivE	ULARA(w_i)	SelectV	SelectH
Base Algorithms	EBED (win)	0.6117	*			
	PTSAD (win)	0.7003				*
	SPIRIT (win)	0.9256			*	*
	ASED (win)	0.6338	*			*
	MAED (win)	0.9002	*			*
	EBED (wout)	0.5595	*			
	PTSAD (wout)	0.7023			*	*
	SPIRIT (wout)	0.8656	*			*
	ASED (wout)	0.9102			*	*
	MAED (wout)	0.9259			*	*
Consensus	Inverse Rank	$*$ 0.8309	0.8174	–	0.8933	$*$ 0.8044
	Kemeny-Young	$*$ 0.9491	$*$ 0.8779	–	0.9511	$*$ 0.9386
	RRA	$*$ 0.8761	$*$ 0.8424	–	0.9578	$*$ 0.9516
	Uni (avg)	$*$ 0.8531	0.8247	–	$*$ 0.9283	$*$ 0.8684
	Uni (max)	$*$ 0.8205	$*$ 0.7632	–	0.8829	$*$ 0.8678
	MM (avg)	$*$ 0.9276	$*$ 0.9487	–	0.9492	$*$ 0.9084
	MM (max)	$*$ 0.8907	$*$ 0.8577	–	0.9410	0.9011
Final Ensemble		0.9092	0.8598	0.7937	**0.9283**	0.9217

Table 2.10: Accuracy of ensembles for RealityMining Bluetooth (undirected) (10 components) (feature: weighted and unweighted degree). $*$: selected detector/consensus results.

		Full	DivE	ULARA(w_i)	SelectV	SelectH
Base Algorithms	EBED (wdeg)	0.4363	*			
	PTSAD (wdeg)	0.5820	*			*
	SPIRIT (wdeg)	0.9499	*			*
	ASED (wdeg)	0.8601			*	*
	MAED (wdeg)	0.8359	*			*
	EBED (udeg)	0.4966	*			
	PTSAD (udeg)	0.8694			*	*
	SPIRIT (udeg)	0.9162			*	*
	ASED (udeg)	0.7662			*	*
	MAED (udeg)	0.8788	*			*
Consensus	Inverse Rank	$*$ 0.8646	$*$ 0.8255	–	0.8790	$*$ 0.8538
	Kemeny-Young	$*$ 0.9534	0.9169	–	0.9698	$*$ 0.9361
	RRA	$*$ 0.9413	0.8318	–	0.9693	$*$ 0.9684
	Uni (avg)	$*$ 0.9071	0.8654	–	$*$ 0.9193	$*$ 0.9225
	Uni (max)	$*$ 0.6973	$*$ 0.6122	–	0.8270	$*$ 0.7126
	MM (avg)	$*$ 0.9407	$*$ 0.9340	–	0.8596	$*$ 0.8892
	MM (max)	$*$ 0.6461	$*$ 0.6374	–	0.8830	0.6461
Final Ensemble		0.8398	0.7735	0.8437	**0.9193**	0.8886

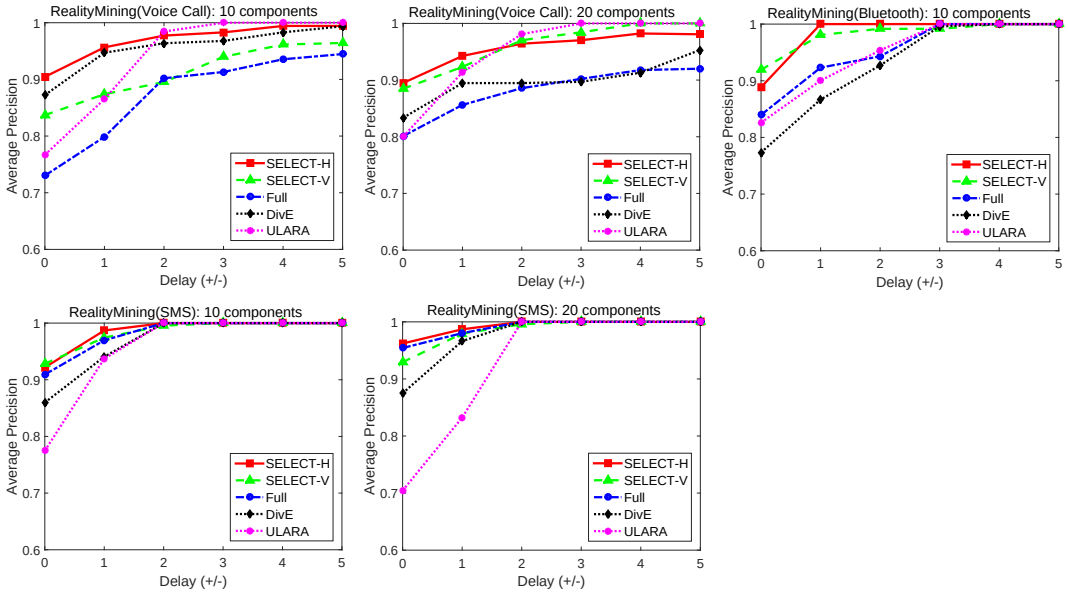


Figure 2.5: RealityMining average precision vs. detection delay for (left to right) Voice Call (10 comp.), Voice Call (20 comp.), Bluetooth (10 comp.), SMS (10 comp.), and SMS (20 comp.).

which is included in Table 2.2. Results are in agreement with prior ones, where SelectH outperforms the other ensembles. This further becomes evident in Figure 2.6 (left), where SelectH can detect all the ground truth events within 3 days delay. The detail results for CyberChallenge dataset are presented in Appendix B.

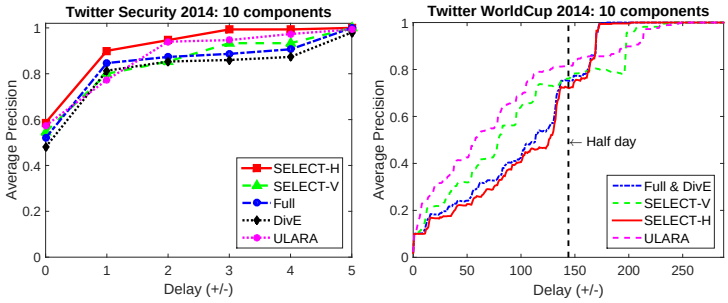


Figure 2.6: Average precision vs. detection delay for (left) Twitter Security and (right) Twitter WorldCup 2014.

The detection dynamics change when TwitterWorldCup is analyzed. The events in this data such as goals and injuries are quite instantaneous (recall the 4 goals in 6 minutes by Germany against Brazil), where we use a sample rate of 5 minutes. Moreover, such events are likely to be reflected on Twitter with some delay by social media users. As such, it is extremely hard to pinpoint the exact

Table 2.11: Accuracy of ensembles for TwitterSecurity (undirected) (10 components) (features: weighted and unweighted degree). ∗: selected detector/consensus results.

		Full	DivE	ULARA(w_i)	SelectV	SelectH
Base Algorithms	EBED (wdeg)	0.4000	∗	▬		∗
	PTSAD (wdeg)	0.5400	∗	▬		∗
	SPIRIT (wdeg)	0.4467	∗	▬	∗	∗
	ASED (wdeg)	0.6200	∗	▬	∗	∗
	MAED (wdeg)	0.4933		▬		∗
	EBED (udeg)	0.4133	∗	▬	∗	∗
	PTSAD (udeg)	0.5467		▬	∗	∗
	SPIRIT (udeg)	0.3867	∗	▬		∗
	ASED (udeg)	0.5400	∗	▬		∗
	MAED (udeg)	0.4533	∗	▬		
Consensus	Inverse Rank	∗ 0.4467	0.4267	–	0.5133	∗ 0.4667
	Kemeny-Young	∗ 0.5667	0.5333	–	0.5333	0.5800
	RRA	∗ 0.5867	∗ 0.5333	–	0.5467	∗ 0.5933
	Uni (avg)	∗ 0.5600	0.5000	–	∗ 0.5467	∗ 0.6000
	Uni (max)	∗ 0.4533	∗ 0.4400	–	0.5800	0.4533
	MM (avg)	∗ 0.5333	∗ 0.5667	–	0.5267	0.5600
	MM (max)	∗ 0.3667	∗ 0.3667	–	0.5533	0.5733
Final Ensemble		0.5200	0.4800	0.5733	0.5467	**0.5867**

time of the events by the ensembles. As we notice in Figure 2.6 (right), the initial accuracies at zero delay are quite low. When delay is allowed for up to 288 time points (i.e., one day), the accuracies incline to a reasonable level within half a day delay. In addition, all the detector and consensus results seem to contain signals in this case where most of them are selected by the ensembles, hence comparable accuracies. In fact, DivE selects all of them and performs the same as Full. Here, ULARA and SelectV perform quite closely.

2.5.3 Noise Analysis

Provided that selecting which results to combine would especially be beneficial in the presence of inaccurate detectors, we design experiments where we introduce increasing number of noisy results into our ensembles. In particular, we create noisy results by randomly shuffling the rank lists output by the base detectors and treat them as additional detector results. Figure 2.7 shows accuracies (avg.'ed over 10 independent runs) on all of our datasets for 10 component ensembles . Results using

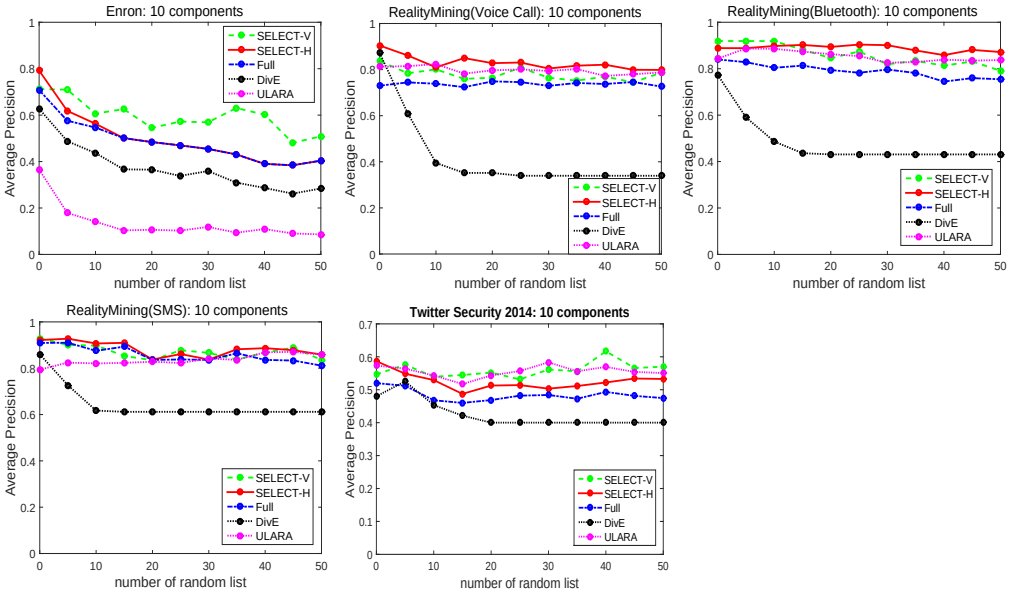


Figure 2.7: Ensemble accuracies drop when increasing number of random results are added for Enron, Voice Call, Bluetooth, SMS, and TwitterSecurity with 10 components (from top to bottom, left to right). Note the decrease is most prominent for DivE.

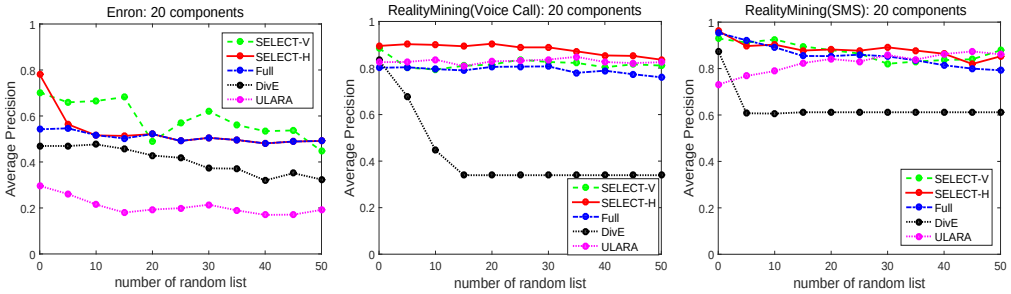


Figure 2.8: Analysis of accuracy when increasing number of random base results are introduced for ensembles with 20 components for Enron, Voice Call, and SMS from left to right. Decline in accuracy under noise is most prominent for DivE.

20 components are similar, and provided in Figure 2.8. We notice that SELECT ensembles provide the most stable and effective performance under increasing number of noisy results. More importantly, these results show that DivE degenerates quite fast in the presence of noise, i.e., when the assumption that all results are reasonably accurate fails to hold. We note that ULARA remains stable in the presence of noise for RealityMining and TwitterSecurity, but degrade in performance for EnronInc.

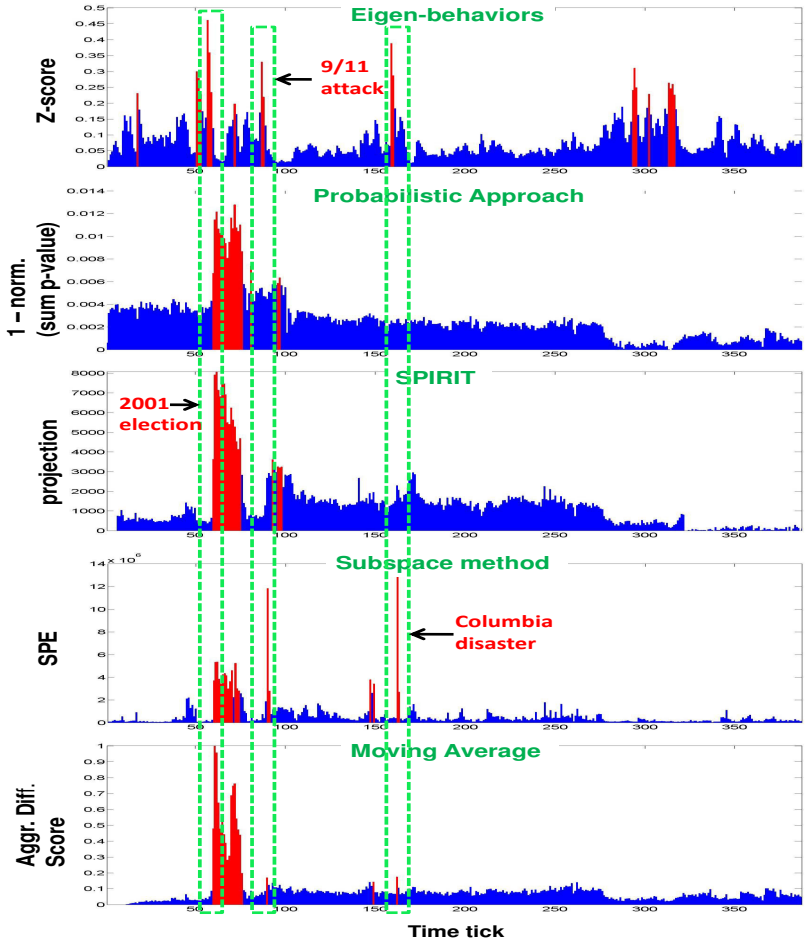


Figure 2.9: Anomaly scores from five base detectors (rows) for NYT news corpus. red bars: top 20 anomalous time points per detector, green boxes: top 3 events by the final ensemble.

2.5.4 Case Studies

In this section we evaluate our ensemble approach qualitatively using the NYTNews corpus dataset, for which we do not have a compiled list of ground truth events. Figure 2.9 shows the anomaly scores for the 2000-2007 time line, provided by the five base detectors using weighted degree feature (we have demonstrated a similar figure for EnronInc. in Figure 2.1 for additional qualitative analysis).

Top three events by SelectH are marked within boxes in the figure, and corresponds to major events such as the 2001 elections, 9/11 WTC attacks, and the 2003 Columbia Space Shuttle disaster. SelectH also ranks entities by association to a detected event for attribution. We note that for the Columbia disaster, NASA

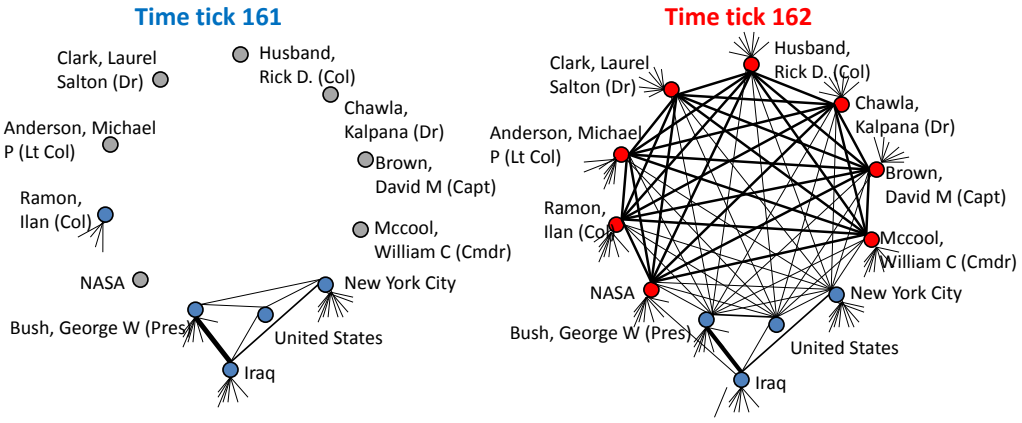


Figure 2.10: During 2003 Columbia disaster a clique of NASA and the seven killed astronauts emerges from time tick 161 to 162.

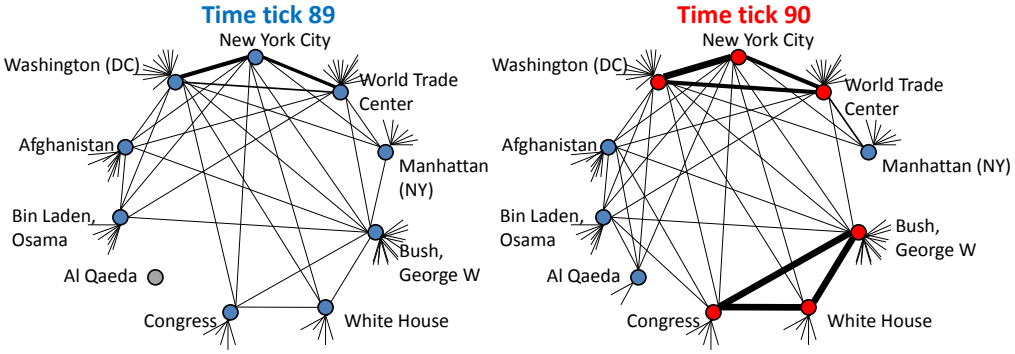


Figure 2.11: During 2001, 9/11 terrorist attacks in WTC heavy links emerge between top ranked entities from time tick 89 to 90.

and the seven astronauts killed in the explosion rank at the top. The visualization of the change in Figure 2.10 shows that a heavy clique with high degree nodes emerges in the graph structure at the time of the event. We also note that for the 9/11 WTC terrorist attacks in 2001, heavily linked entities e.g. World Trade Center, New York City, Washington (DC), George W. Bush, White House, Congress are rank at the top. The visualization of the change in Figure 2.11 shows that the heavy links emerge between the top ranked entities in the graph right after the event has occurred.

Table 2.12: Significance of accuracy results compared to random ensembles with same number of selected components as SELECT for outlier detection. The accuracy of the alterntive approaches (Full, DivE, and ULARA) are also given in parentheses. These results show that (i) SELECT is superior to existing methods, and (ii) it selects significantly more important (i.e., accurate) detectors to combine.

	Accuracy	significance
WBC (25 comp.) (Full: 0.3283, DivE: 0.2416, ULARA: 0.2439)		
(i) RandE (19/25, 6/7)	0.3209 (μ)	0.0091 (σ)
SelectV	0.2950	$= \mu - 2.8462\sigma$
(ii) RandE (21/25, 6/7)	0.3256 (μ)	0.0093 (σ)
SelectH	**0.3840**	$= \mu + 6.2796\sigma$
Glass (25 comp.) (Full: 0.2134, DivE: 0.1326, ULARA: 0.2022)		
(i) RandE (19/25, 3/7)	0.2032 (μ)	0.0224 (σ)
SelectV	0.2079	$= \mu + 0.2098\sigma$
(ii) RandE (21/25, 5/7)	0.2136 (μ)	0.0060 (σ)
SelectH	**0.2194**	$= \mu + 0.9667\sigma$
Lympho (25 comp.) (Full: 0.7287, DivE: 0.7593, ULARA: 0.7121)		
(i) RandE (15/25, 1/7)	0.6199 (μ)	0.1334 (σ)
SelectV	0.6347	$= \mu + 0.1109\sigma$
(ii) RandE (24/25, 6/7)	0.7488 (μ)	0.0291 (σ)
SelectH	**0.7843**	$= \mu + 1.2199\sigma$
Musk (25 comp.) (Full: 0.1195, DivE: 0.0981, ULARA: 0.1106)		
(i) RandE (14/25, 6/7)	0.1228 (μ)	0.0233 (σ)
SelectV	**0.1355**	$= \mu + 0.5451\sigma$
(ii) RandE (18/25, 5/7)	0.1162 (μ)	0.0239 (σ)
SelectH	0.1138	$= \mu - 0.1004\sigma$
Cardio (25 comp.) (Full: 0.3202, DivE: 0.2932, ULARA: 0.2331)		
(i) RandE (11/25, 6/7)	0.2709 (μ)	0.0185 (σ)
SelectV	0.2767	$= \mu + 0.3135\sigma$
(ii) RandE (19/25, 6/7)	0.3193 (μ)	0.0185 (σ)
SelectH	**0.4389**	$= \mu + 6.4649\sigma$
Thyroid (25 comp.) (Full: 0.0815, DivE: 0.0773, ULARA: 0.0925)		
(i) RandE (1/25, 0/7)	0.1445 (μ)	0.1036 (σ)
SelectV	**0.2342**	$= \mu + 0.9765\sigma$
(ii) RandE (20/25, 4/7)	0.1054 (μ)	0.0213 (σ)
SelectH	0.1412	$= \mu + 1.6808\sigma$
Letter (25 comp.) (Full: 0.4292, DivE: 0.4335, ULARA: 0.4298)		
(i) RandE (25/25, 6/7)	0.4303 (μ)	0.0043 (σ)
SelectV	0.4286	$= \mu - 0.3953\sigma$
(ii) RandE (18/25, 6/7)	0.4489 (μ)	0.0110 (σ)
SelectH	**0.5504**	$= \mu + 9.2273\sigma$

2.5.5 Outlier Detection Performance

Next we quantitatively evaluate the ensemble methods on outlier detection accuracy on 7 real-world datasets. Here, we use *average precision* as the measure of accuracy. We utilize different values of the parameter k (number of nearest neighbors) for the individual based detectors to provide 25 base components for the outlier ensembles. For Cardio dataset we use $k = 5, 10, 50, 100, 500$, and for all the other datasets we use $k = 5, 10, 15, 20, 25$.

In Table 2.12 we report the summary of results for different multi-dimensional point datasets containing the average precision values for SELECT and other existing ensemble approaches (Full, DivE and ULARA) to which we compare SELECT. Similar to the event detection results we further investigate the significance of SELECT and provide the results for random ensemble (RandE) where we randomly select the same number of components asSELECT to assemble at each phase. We note that either SelectH or SelectV provides superior results to RandE, Full, DivE, and ULARA. Moreover, SelectH appears to be a better strategy than SelectV, as in most cases it provides the best result(in Table 2.12). Whereas, in some cases SelectV fall short to even beat Full, DivE, and ULARA. For brevity we skip the detailed accuracy tables and noise analysis plots as provided for event detection results.

2.6 Summary of Contributions

In this work we design SELECT, a new selective ensemble approach for anomaly mining, and applied it to the event detection problem in temporal graphs and outlier detection problem in multi-dimensional point data (no-graph). SELECT is a two-phase approach that combines multiple detector results and then multiple consensuses, respectively. Motivated by our earlier observations [34] that inaccurate detectors may deteriorate overall ensemble accuracy, we designed two unsupervised selection strategies, SelectV and SelectH, which carefully choose which detector/consensus outcomes to assemble. We compared SELECT to Full, the ensemble that combines all results, DivE, an existing ensemble [32] that combines diverse, i.e., least correlated results, and ULARA, a weighted rank aggregation approach [35].

Our quantitative evaluation for both event and outlier ensemble on real-world datasets with ground truth show that building selective ensembles is effective in boosting detection performance. SelectH appears to be a better strategy than SelectV, where it either provides the best result (8/10 in Table 2.2, Table 2.3 and 5/7 in Table 2.12) or achieves comparable accuracy when SelectV is the winner. Selecting results based on diversity turns out to be a poor strategy for anomaly ensembles as DivE yields even worse results than the Full ensemble (6/10 in Table 2.2, Table 2.3

and 5/7 in Table 2.12). Noise analysis for event detection further corroborates the fact that DivE selects inaccurate/noisy results for the sake of diversity and declines in accuracy much faster than the rest. Table 2.2, Table 2.3 and 2.12 also show that ULARA is worse than Full in 3/6 cases, 3/4 cases , and in 5/7 cases respectively, suggesting no clear winner. In comparison, SelectV and SelectH respectively outperform Full in 8/10 and 9/10 cases for event detection, 6/7 and 2/7 cases for outlier detection. This suggests that while the Full ensemble is inferior in the presence of inaccurate detectors, as a selective ensemble SELECT, specifically SelectH is superior to existing approaches like DivE and ULARA.

Future work will investigate how to go beyond binary selection and estimate weights for the detector/consensus results. We will continue to enhance SELECT with other detectors and consensus methods. Finally, all source code of our methods and datasets used in this work are made available at `http://shebuti.com/SelectiveAnomalyEnsemble/`.

Chapter 3

Sequential Ensemble Learning for Outlier Detection

As a significant subject, outlier detection is widely researched in the literature. There exist various approaches for outlier detection such as density based methods [53, 72, 73] and distance based methods [52, 74], which find unusual points by the distance to their k nearest neighbors (kNNs). However, each of these methods can only focus on some specific kinds of outliers based on the data sets collected from different application domains. There exists no known algorithm that could detect all types of outliers that appear in a wide variety of domains. As a result, ensemble learning for outlier detection has become a popular research area more recently [16, 68, 75], which aims to put together multiple detectors so as to leverage the "strength of the many".

In contrast to outlier detection ensembles, classification ensembles have been studied for decades. The explosive growth of classification ensemble models provides the new opportunity to design effective methods for other machine learning tasks including outlier mining. One can categorize ensemble methods into two kinds. The first one is the parallel ensemble, where base learners are created independent of each other and their results are combined to get the final outcome; while the second one is the sequential ensemble, where base learners are created over iterations and have dependency among them. Specifically, several outlier ensembles are proposed based on two seminal works of classification ensembles: (*i*) the parallel ensemble Bagging [44], which creates base components from different subsamples of training datasets parallelly, and (*ii*) the sequential ensemble AdaBoost [76], which creates base components iteratively. Among those, some try to induce diversity among the base detectors [30, 68, 75], and others selectively combine outcomes from the candidate detectors [16, 77].

Existing outlier ensembles have several limitations, most importantly they avoid discussing the theoretical aspects of outlier detection. Recently, Aggarwal *et al.* [68] argue that although they appear to be very different problems, classification and outlier detection share quite similar theoretical underpinnings in terms of the bias-variance trade-off. Specifically, one can consider the outlier detection problem as a binary classification task where the labels are unobserved, the inliers being the majority class and the outliers the minority class, and the error of a detector can be decomposed into bias and variance terms in a similar way. In existing outlier ensembles, various parallel frameworks combining multiple detector outcomes are designed to reduce variance only, most of which are incapable of overcoming the presence of inaccurate base detectors. On the other hand, it remains challenging to reduce bias in a controlled way for outlier detection or remove inaccurate detectors due to the lack of ground truth to validate the results during the intermediate steps. There exist some successful heuristic approaches to reduce bias. One such commonly used approach is to remove outliers in successive iterations [14] to build more robust outlier models iteratively.

In this work, we study the feasibility of bias-variance reduction under the unsupervised setting, and propose a sequential ensemble model called Cumulative Agreement Rates Ensemble (CARE), to reduce both bias and variance for outlier detection. Specifically, each iteration in the sequential ensemble consists of two aggregation phases: (1) in the first phase, we combine the results of feature-bagged base detectors using weighted aggregation, where weights are estimated in an unsupervised way through the Agreement Rates (AR) method by [78], and (2) in the second phase, the result of the current iteration is aggregated with the combined result from the previous iterations cumulatively. These two phase aggregations in each iteration aim to reduce the variance. Furthermore, we use the combined result from the previous iterations to improve the next iteration by removing the top (i.e., most obvious) outliers and perform a variable probability sampling to create the data model to be used for the next iteration. The removal of top outliers in successive iterations aims to reduce the bias.

To the best of our knowledge, this is the first work focusing on reducing both bias and variance for unsupervised outlier detection. In general, this work offers the following contributions:

- We design a new approach which incorporates weighted aggregation of feature-bagged base detectors, where weights are estimated in an unsupervised fashion (Section 3.3.3 and 3.3.3).
- We devise a sequential ensemble over the weighted combination, which cumulatively aggregates the results from multiple iterations until a stopping condition

is met (Section 3.3.3 and 3.3.3).

- We provide a new sampling approach called Filtered Variable Probability Sampling (FVPS) which utilizes the result from the previous iteration to filter the top outliers, and uses variable probability sampling to select points from the original data to create the data model for the next iteration (Section 3.3.3).
- Our sequential ensemble is designed to reduce both bias and variance and improves the overall result. Moreover, we provide experiments with synthetic datasets to support this claim (Section 3.4).
- We further design iCARE which is an isolation based CARE approach to improve both speed and performance (Section 3.7).

We evaluate our method on twenty one different real-world datasets, majority of which are from the UCI machine learning repository [70]. Our results show that CARE outperforms the baseline (i.e., non-ensemble) detectors in most cases and remains close to the baselines in cases where it falls shorter. We also compare CARE with the existing state-of-the-art outlier ensembles [68, 75, 79]. Similarly, it provides significant improvement when it is the winner, and performs close otherwise (Section 3.8).

3.1 Related Work

3.1.1 Ensemble Models

Ensemble models for classification have been extensively studied in the literature for several decades. In 1996, Breiman [44] presented a parallel ensemble, today well-known as *bagging*, which consists of multiple predictor components trained on samples of the original dataset, to infer the label using a plurality vote, enhancing the model through variance reduction. However, bagging omitted the bias term and could only reduce the variance. To make up this deficit, a new sequential ensemble known as *boosting* was devised by Freund *et al.* [76]. Their proposed AdaBoost algorithm assigned larger weights for the misclassified instances to advance the given base classification algorithm and combined weighted sum of multiple weak learners into a boosted classifier to reduce both bias and variance.

After these two important seminal works, a proliferation of ensemble methods followed, aiming to explain and improve over the original methods. A two-stage process called adaptive bagging [80] was proposed to perform bias and variance reduction respectively. It generated an intermediate output in the first stage and

the altered first stage output was adopted as new input of the second stage that is bagging. Sun *et al.* [81] analyzed the influence of adding a cost term to AdaBoost and offered the cost-sensitive boosting algorithm for imbalanced data classification.

Our proposed outlier detection approach CARE uses similar insights as in AdaBoost; by sequentially updating the data model on which data points are scored for outlierness as well as by combining multiple detector outcomes parallelly to reduce bias and variance, respectively. Since the above methods can be learned in supervised settings while most anomaly detection tasks provide no labels, our method differs from the existing classifier ensembles in that we focus on reducing both bias and variance in a fully unsupervised setting, which has not been studied before.

3.1.2 Outlier Ensembles

Outlier ensemble learning, as a rarely explored area, mainly tries to reduce the variance through the combination of different base detectors. A parallel approach called feature bagging, proposed by Lazarevic and Kumar [30], built an ensemble based on randomly selected feature subsets from original features to detect outliers in high-dimensional and noisy datasets. Inspired by random forests [82], Liu *et al.* [79] employed different subsamples of training data to establish an ensemble of trees to isolate outliers on the basis of the path length from the root to the leaves.

In recent years, with more attention focusing on outlier ensembles, several work discussed the theories and emphasized the crucial aspects of ensemble model construction. Aggarwal [83] and Zimek [15] talked about algorithmic patterns, categorization, and the important building blocks of outlier ensembles such as model combination and diversity of base models. Zimek *et al.* [75] analytically and experimentally studied the subsampling technique and improved results through building an ensemble on top of several subsamples without mentioning the subsample selection. Aggarwal and Sathe [68] deduced the bias-variance trade-off theory from classification to outlier detection, clarified some misconceptions about the existing subsampling methods, and proposed more effective subsampling and feature bagging approaches. On base detector combination, Rayana and Akoglu [16] presented unsupervised strategies to select a subset of trusted detectors while omitting inaccurate ones in an unsupervised way.

Unlike existing outlier ensembles that solely employ a sequential or parallel framework, our proposed method CARE incorporates both of these building blocks to reduce both bias and variance. These two phases respectively involve (i) successively eliminating outliers from the original dataset to build a better data model on which outlierness is estimated (sequentially), and (ii) combining the results from individual base detectors and across iterations (parallelly).

3.2 Background and Preliminaries

3.2.1 Bias-Variance Trade-off for Outlier Detection

The bias-variance trade-off is often explained in the context of supervised learning, e.g., classification, as quantification of bias-variance requires labeled data. Although outlier detection problems are solved using unsupervised approaches due to the lack of ground truth, the bias-variance trade-off can be quantified similarly by treating the dependent variable (actual labels) as unobserved [68].

Unlike classification, most outlier detection algorithms output "outlierness" scores for the data points. We can consider the outlier detection problem as a binary classification task having a majority class (inliers) and a minority class (outliers) by converting the outlierness scores to class labels. The points with scores above a threshold are considered as outliers with label 1 (label 0 for inliers below threshold). After converting the unsupervised outlier detection problem to a classification task with only unobserved labels, we can explain the bias-variance trade-off for outlier detection using ideas from classification. Specifically, the expected error of outlier detection can be split into two main components: reducible error and irreducible error (i.e., error due to noise). The reducible error can be minimized to maximize the accuracy of the detector. Furthermore, the reducible error can be decomposed into (i) error due to squared bias, and (ii) error due to variance. However, there is a trade-off while minimizing both these sources of errors.

Bias of a detector is the amount by which the expected output of the detector differs from the actual label (unobserved) over the training data. While variance of a detector is the expected amount by which the output of a detector over one training set differs from the expected output of the detector over all the training sets. Simply put, the trade-off between bias and variance can be viewed as, (i) a detector with low bias is very flexible in fitting the data well and fits each training set differently with high variance, and (ii) an inflexible detector fits each training set almost similarly yielding high bias but low variance.

3.2.2 Motivation for Sequential Ensembles

Our goal in this work is to improve outlier detection by reducing both bias and variance, and in return decreasing the reducible error. It is evident from the classification ensemble literature that combining results from multiple base algorithms decreases the overall variance of the ensemble [30,44,68], which is also true for outlier ensembles. On the other hand, this combination does not provide any evidence for reducing bias, as controlled bias reduction is rather difficult due to lack of ground

truth. However, there exist some successful heuristic approaches for reducing bias by removing outliers iteratively to build a more robust successive outlier detector. This iterative approach can be considered as a sequential ensemble. The basic idea here is that the outliers interfere with the creation of the normal data model, and the removal of points with high outlierness scores will be beneficial for the outlier model to produce an output close to the actual (unobserved) labels in expectation.

3.3 Methodology

3.3.1 Overview

CARE takes the d-dimensional data, a value for k (nearest neighbor count), and a value for $MAXITER$ as input and outputs an outlierness score list **fs** and a rank list **r** (ranked based on most to least outlierness) of all the data points. In the experiments we use $k = 5$, which is compatible with the state-of-the-art methods [68, 75]. Moreover, parameter k in subsampling methods is scaled by the inverse

Algorithm 5: CARE Outlier Detection Ensemble

Input: d-dimensional Data D, NN count $k = 5$, $MAXITER = 15$
Output: Score list (**fs**) and rank list (**r**) of points

1: $S = D$ (initially); $E = \emptyset$; $iter = 0$
2: **while** $iter \leq MAXITER$ **do**
3: Obtain results from (b) feature-bagged base detectors (D, S, k) [Section 3.3.2]
4: Calculate pairwise agreement rates a_A for all base detector pairs in set A
5: Estimate detector errors **e** ($b \times 1$) based on a_A [Section 3.3.3]
6: Compute detector weights using estimated errors [Section 3.3.3]
7: Compute pruned weighted outlierness scores of data points to get combined scores (**ws**) [Section 3.3.3]
8: $E = E \cup \mathbf{ws}$
9: $\mathbf{fs} = average(E)$
10: Generate new data sample S from D using FVPS (w/o replacement) on **fs** [Section 3.3.3]
11: $\mathbf{r} = sort(\mathbf{fs})$ (descending order)
12: **if** *stopping condition* is TRUE **then**
13: *break* [Section 3.3.3]
14: **end if**
15: $iter = iter + 1$
16: **end while**

of various subsample sizes, as such large k is not required (the smaller the subsample, the larger the relative neighborhood per point becomes for a fixed k). As for $MAXITER$, we set it to 15, a relatively small value. We assume that our approach improves the base detectors over iterations, and the results are stabilized after only a few iterations and the algorithm stops following the stopping criterion.

The main steps of CARE are given in Algorithm 5. Step 3 creates the feature-bagged outlier detectors as base detectors of the ensemble. For the first iteration the sample set S contains the whole data D as shown in step 1. For each base detector, $q \in [d/2, d-1]$ features are selected randomly to create the corresponding feature bag. We create b ($= 100$) feature-bagged base detectors. Motivated by Platanios *et al.* [78], step 4 calculates the pairwise agreements a_A for all possible pairs of base detectors and step 5 estimates the error rates of the individual base detectors in an unsupervised way using a_A. Step 6 calculates weights for the base detectors using their corresponding error rates. Step 7 combines the outlierness scores from the different base detectors with weighted average combination to get final outlierness scores **ws**. Step 8 stores the outlierness scores in E at each iteration and step 9 calculates the final outlierness scores **fs** by averaging the results of all previous iterations as well as the current iteration. Based on **fs**, step 10 generates the new data sample S (where, $|S| < |D|$) using the FVPS approach w/o replacement (see Section 3.3.3) and step 11 generates the ranked list **r** of instances from most to least outlierness. We repeat steps 3-16 until the stopping condition at step 12 is met or upto the given maximum iteration $MAXITER$. As the algorithm suggests the complexity of CARE is quadratic in the size of the dataset n.

Unlike existing ensemble techniques, CARE incorporates a two-phase aggregation approach in each iteration; first, it combines the results from the individual base detectors (parallel) and second, it cumulatively aggregates the results from multiple iterations (sequential).

Next we describe the main components of our proposed CARE in detail. In particular, we describe the base detectors in Section 3.3.2 and consensus approaches in Section 3.3.3.

3.3.2 Base Detectors

There exist various approaches for outlier detection based on different aspects of outliers, designed for distinct applications to detect domain-specific outliers. In our work, we are interested in *unsupervised* outlier detection approaches that assign outlierness scores to the individual instances in the data, as such, allow ranking of instances based on outlierness.

kNN based Outlier Detectors

There are a number of well-known unsupervised approaches, e.g., "distance based" and "density based" methods for outlier detection. Distance based methods [51, 52] and their variants try to find the *global* outliers far from the rest of the data based on k nearest neighbor (kNN) distances of the data points. On the other hand, density based methods [53, 72, 73] and their variants try to find the *local* outliers which are located in a lower density region compared to their k nearest neighbors.

In this work, we create two versions of CARE: (1) using the distance based approach AvgKNN (average k nearest neighbor distance of individual data point is used as outlierness score), and (2) using the most popular density-based approach LOF [72]. We note that CARE is flexible to accommodate any other nearest neighbor based outlier detection algorithm as well.

Feature Bagging

Feature bagging is commonly used in classification ensembles for dimensionality reduction as well as for variance reduction. Like classification ensembles, feature bagging can also be incorporated in outlier ensembles in order to explore multiple subspaces of the data to induce diverse base detectors for high-dimensional outlier detection. As such, in this work we incorporate feature bagging to create multiple base detectors and combine their results to detect outliers with a goal to improve the detection performance by reducing variance. Given a d-dimensional dataset D, for each base detector (either LOF or AvgKNN), we randomly select $q \in [d/2, d-1]$ features to create b ($= 100$) different feature-bagged base detectors.

3.3.3 Consensus Approaches

Unlike classification, building an effective ensemble for outlier detection is a challenging task due to the lack of ground truth, which makes it difficult to measure the detector accuracy and combine the results from accurate detectors. Most of the existing approaches either combine outcomes of all the base detectors [30, 34] (hurting the ensemble in presence of poor detectors), or selectively incorporate accurate base detectors in an unsupervised fashion discarding the poor ones [15, 16]. However, the definition of a poor detector varies across different application domains, as some selective approaches are useful for certain applications but not as useful for others. Therefore, in this work we go beyond binary selection and estimate weights for individual base detectors to aggregate their results with a weighted combination. Furthermore, we cumulatively combine the weighted aggregation results for multiple

iterations until a stopping condition is satisfied. In the following two sections, we describe the error estimation and weighted aggregation of the base detectors. In Section 3.3.3 the sequential aggregation approach is described and in Section 3.3.3 we introduce a stopping condition for our iterative CARE approach.

Error Estimation

Platanios *et al.* [78] proposed an *unsupervised* approach called Agreement Rates (AR) to estimate errors of multiple classifiers. Motivated by [78], we adapt the unsupervised error estimation of the individual outlier detectors in our work. This estimation is based on the agreement rates for all possible pairs of base detectors in A. Outlier detection can be considered as a binary classification problem with a majority class (inliers = 0) and a minority class (outliers = 1). However, most existing outlier detection algorithms provide outlierness scores for the data points, and not $\{0, 1\}$ labels for them. In order to adapt the AR approach, we calculate the agreement rates for all possible pairs of detectors in A, for which $\{0, 1\}$ labels are needed for the data points. We use Cantelli's inequality [84] to estimate a threshold th_i $(i = 1 \dots b)$ with confidence level at 20% to find a cutoff point between inliers $(= 0)$ and outliers $(= 1)$ for each base detector to get a binary list of class labels.

After estimating the class labels, we calculate the agreement rates. As inliers are the majority class, if we take into account all the data points in calculating the agreement rates, it is likely that most values would be large as most detectors often agree on a large number of inliers. Our main goal is to find agreement based on the outliers detected by the base detectors, and ignore a large number of inliers. Therefore, we take the union of all outliers $(= 1)$ across different base detectors to obtain U. Set U contains the important data points (detected as outliers), which we use to calculate the agreement rates for the detector pairs in A.

In the following sections we denote the base detectors as $f_i \in F$ $(i = 1 \dots b, |F| = b)$, input data as D, and class labels as Y. The error event E_A of a set of detectors in A is defined as an event when all the detectors make an error:

$$E_A = \bigcap_{i \in A} [f_i(D) \neq Y] \ , \tag{3.1}$$

where $\bigcap$ denotes set intersection. The error rate of a set of detectors in A is then defined as the probability that all detectors in A make an error together and is denoted as

$$e_A = \mathbb{P}(E_A) \ . \tag{3.2}$$

The agreement rate of two detectors is the probability that both make an error or

neither makes an error. As such, the pairwise agreement rate equation in terms of error rates for the sets in $A : |A| = 2$ can be written as

$$\begin{aligned} a_{\{i,j\}} &= \mathbb{P}(E_{\{i\}} \cap E_{\{j\}}) + \mathbb{P}(\bar{E}_{\{i\}} \cap \bar{E}_{\{j\}}) \\ &= 1 - e_{\{i\}} - e_{\{j\}} + 2e_{\{i,j\}} \ , \quad \forall \{i,j\} \in A : i \neq j \ , \end{aligned} \tag{3.3}$$

where $\bar{\cdot}$ denotes the set complement. On the other hand, the agreement rates for the set of detectors in $A : |A| = 2$ can be directly calculated from the detector output and set U (defined earlier) as follows:

$$a_A = \frac{1}{|U|} \sum_{u=1}^{|U|} \mathbb{I}\{f_i(D_u) = f_j(D_u)\}, \forall \{i,j\} \in A : i \neq j \ . \tag{3.4}$$

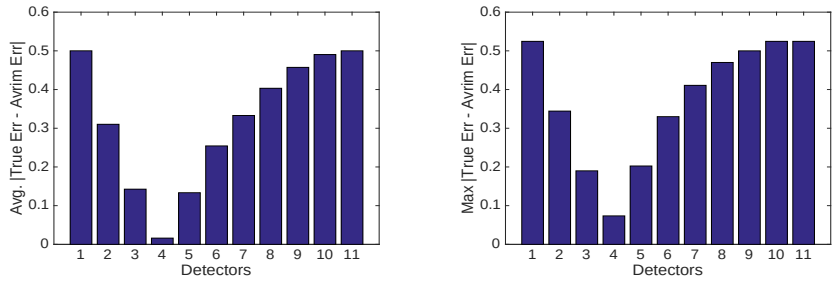
Provided that one can easily compute the pairwise agreement rates $a_{\{i,j\}}$'s, which can be written in terms of the (unknown) individual and pairwise error rates of the detectors, we can cast the error rate estimation as a constrained optimization problem where the agreement equations in (3.3) form constraints that must be satisfied as follows:

$$\begin{aligned} \textbf{min.} \quad & \sum_{\hat{A}:|\hat{A}|\leq 2} e_{\hat{A}}^2 + \epsilon_{\hat{A}} \\ \textbf{s.t.} \quad & a_A = 1 - e_{\{i\}} - e_{\{j\}} + 2e_{\{i,j\}} \ , \ \forall \{i,j\} \in A \\ & 0 \leq e_{\hat{A}} < 0.5 + \epsilon_{\hat{A}} \ , \\ & 0 \leq \epsilon_{\hat{A}} \end{aligned} \tag{3.5}$$

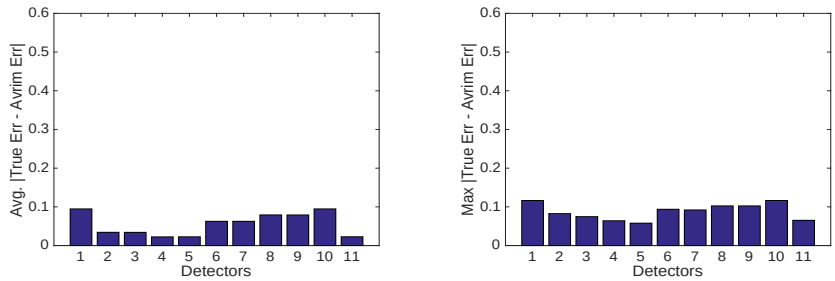
where $\hat{A}$ contains individual as well as pairs of detectors (i.e., $\hat{A} = F \cup A$) and $\epsilon_{\hat{A}}$'s denote the slack variables.

In their AR approach, Platanios *et al.* assume that the error rates should be strictly < 0.5. Different from theirs, we allow the error rates to be above 0.5, for which we introduce a slack variable $\epsilon_{\hat{A}} \geq 0$ in constraints $0 \leq e_{\hat{A}} \leq 0.5 + \epsilon_{\hat{A}}$. In real-world settings, it is possible to have poor base detectors having large errors (i.e., worse than random). Then the question becomes whether the presence of poor detectors (with error ≥ 0.5) hampers the overall estimation of the errors. To answer this question we have designed experiments with synthetic datasets mimicking the real datasets having 1000 data points in total, where 10% of them are outliers and 11 base detectors with different true error rates. We generate multiple (= 100) snapshots of the synthetic datasets randomly, to analyze results. Figure 3.1 shows the average and maximum difference between the true error and estimated error of different base detectors. In Figure 3.1 (a), 6/11 detectors have true error ≥ 0.5, as a result the average and maximum differences are larger as compared to Figure 3.1 (b) and (c), where in (b) none and in (c) only 2/11 detectors have errors ≥ 0.5. We

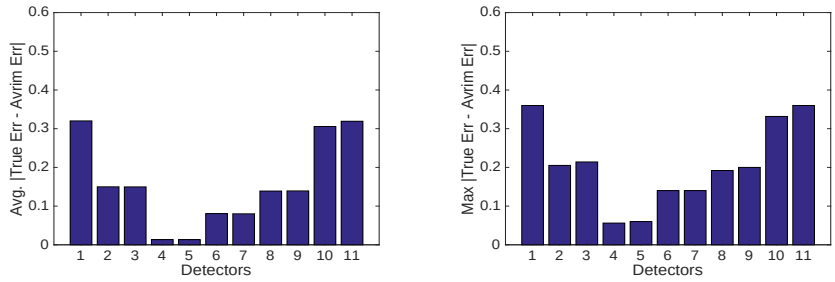
conclude that if all the detectors are good (better than random) or only a few are bad, the optimization in (3.5) estimates meaningful error rates.



(a) $e = [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]$



(b) $e = [0.0, 0.0, 0.1, 0.1, 0.2, 0.2, 0.2, 0.3, 0.3, 0.4, 0.4]$



(c) $e = [0.0, 0.1, 0.1, 0.2, 0.2, 0.3, 0.3, 0.4, 0.4, 0.8, 0.9]$

Figure 3.1: Average (left) and Maximum (right) difference (across datasets) between the true error and the estimated error of different base detectors (e represents true error rates of base detectors). Notice that the differences are high with the presence of many bad detectors, but low otherwise.

Although the above constrained optimization approach estimates error rates of individual as well as of all possible pairs of base detectors, we only utilize the error rates of the individual detectors to calculate their corresponding weights for aggregation, which we describe next.

Weighted Aggregation

Most commonly used aggregation functions in outlier ensembles are *average* and *maximum*. In most cases, average is preferred over maximum as the latter overestimates the absolute scores. On the other hand, averaging might dilute the final scores with the presence of poor detectors. In CARE, we propose to use *weighted aggregation* to improve the ensemble.

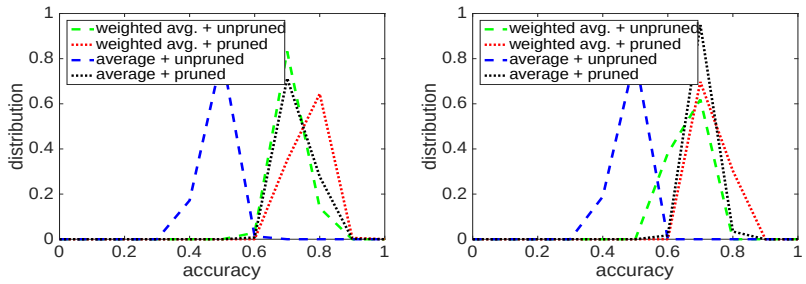
We calculate the weights of the base detectors from their estimated error rates as described in the previous section, such that the weights are positive and inversely proportional to the corresponding errors. Inspired by AdaBoost [76], we employ the error rates of individual detectors to calculate their corresponding weights using the following equation:

$$w_i = \frac{1}{2}\log\left(\frac{2}{e_i} - 1\right)\ ,\ i = 1 \dots b \tag{3.6}$$

where $w_i \geq 0$ is the weight of detector i with estimated error $e_i \in [0, 1]$, for $i = 1 \dots b$. Moreover, as we assume that in real-world settings the base detector pool will have poor (i.e., worse than random) detectors, we also incorporate a pruning strategy where we discard the detectors with error $e_i \geq 0.5$. To support the weighted aggregation strategy with pruning, we provide experimental results with synthetic datasets to compare average vs. weighted aggregation as well as pruned vs. un-pruned selection. In Figure 3.2, we show the distributions of the final ensemble accuracies with different consensus approaches across 1000 samples of a synthetic dataset. Each synthetic data has 1000 points and 4 detectors with true errors $e = [0.2, 0.4, 0.6, 0.8]$ having 5% (left) and 10% (right) of outliers respectively. Here, we calculate the weights and prune the detectors using these estimated errors. We can see from both figures that weighted consensus is better than averaging and pruning is better than un-pruned aggregation as the red curve is more skewed towards the higher accuracy values than the others.

After pruning p detectors with error $e_i \geq 0.5$, we combine the outlierness scores from the base detectors using weighted aggregation. In order to do weighted aggregation, we need to unify the outlierness scores, as different base detectors employ different feature sets, hence provide scores with varying range and scale. To standardize, we use Gaussian Scaling [85] to convert the outlierness scores of AvgKNN or LOF into probability estimates Pr_i $(i = 1 \dots b - p) \in [0, 1]$. We calculate the final outlierness score $ws(x)$ of a data point x using the weighted average of the probability estimates as follows:

$$ws(x) = \frac{\sum_{i=1}^{b-p} w_i \times Pr_i(x)}{\sum_{i=1}^{b-p} w_i} \tag{3.7}$$



5% outliers 10% outliers

Figure 3.2: Distribution of accuracies with different consensus approaches. Notice the distribution with pruned weighted aggregation (red curve) is skewed towards higher accuracies.

Above, $\sum_{i=1}^{b-p} w_i$ is used to normalize the outlierness scores. The final scores can be used to sort the data points from most to least outlierness to produce a ranked list $\mathbf{r}$. Again, we use Cantelli's inequality on $\mathbf{fs}$ to calculate a threshold with confidence level 20% to find the final cutoff point between outliers and inliers.

Thus far, we described steps 3–7 of Algorithm 5. Next we describe the iterative nature of our sequential ensemble.

Sequential Ensemble

With the weighted aggregation combining multiple feature-bagged base detectors we aim to reduce variance, but our additional goal is to reduce bias. In outlier detection, it is hard to reduce bias in a controlled way, but there exist some successful heuristics to reduce bias. One commonly used approach is to remove outliers in successive iterations [14] in order to build more robust outlier models iteratively. This is a type of sequential ensemble. The basic idea is that the outliers interfere with the creation of a model of normal data, and the removal of points with high outlier scores is beneficial for the model in the following iteration.

As such, we adopt a sequential ensemble approach in CARE where we use the result from the previous iteration to improve the next. In particular, we select a subsample S from the original data D (where $|S| < |D|$) to use it as a *new data model based on which we calculate the outlierness scores* for all the data points in D. For example, when we need the average kNN distance of a data point $x \in D$, we calculate the distance to its k-nearest neighbors $N_i \in S$. The goal is to construct S that includes as few of the true outliers as possible, such that it serves as a more reliable data model. To do so, we design a sampling approach which we call Filtered Variable Probability Sampling (FVPS). Following are the steps of the FVPS:

- Discard top T outliers detected in previous step from D, where T is the number of outliers selected using Cantelli's inequality [84] on final outlierness scores **fs** (threshold is selected at 20% confidence level to find the cutoff point between outliers and inliers).
- Select l uniformly at random between $\min\{1 - \frac{T}{n}, \frac{50}{n}\}$ and $\max\{1 - \frac{T}{n}, \frac{1000}{n}\}$, where n is the number of points in the original dataset.
- Build sub-sample S (where $|S| = l \times (n - T)$) by sampling from D' (outliers-discarded) based on the probability of the points being normal (i.e., $(1 - \mathbf{fs})$).

In step 1 of FVPS, we obtain D' by filtering the outliers detected in the previous step to reduce bias and improve the outlier ensemble iteratively. Here, we choose confidence level 20% to get a larger T in order to remove as many outliers as possible. Even though this step might remove some inliers, those should not effect the model as they would have lower probability of being normal points to be removed in the first place. Inspired by Aggarwal and Sathe [68], we adopt variable sampling to select a sample size in step 2. The variable sampling approach has an effect over the parameter choice of the outlier detectors (i.e., k). Varying the subsample size at fixed k effectively varies the percentile value of k in the subsample for different iterations. For some datasets smaller value of k is better, for others larger is better. However, there is no known suitable approach to estimate the correct value of k for a dataset. Therefore, in CARE we select a small value of k (e.g., 5) and employ variable sampling to incorporate the illusion of using different k values in different iterations, which introduces diverse detectors iteratively. After deciding the sample size in step 2, we use probability sampling to create the data model S in step 3. Here, we choose a point from D' to include in S based on its probability of being normal. As a result, we expect to have less interference from the outliers in S as it is mostly built with normal points.

FVPS introduces diverse detectors based on different S in each iteration, hence, we aggregate (e.g. cumulative average) the outlierness scores **ws** over the iterations to compute final scores **fs** to further reduce variance and improve the sequential ensemble (step 9 in Alg. 5). Note that **fs** is also what FVPS uses for discarding outliers and sampling set S. This sequential ensemble requires a sopping point at which the iteration stops and returns the final cumulative result. In the following section, we introduce a stopping criterion for the sequential ensemble.

Stopping Criterion

We need a proper stopping criterion for the sequential ensemble approach to decide where the iteration should stop and return the final result. As the whole framework is unsupervised, there is no way to use intermediate evaluation to find a stopping point. In CARE, we utilize the pairwise agreement rates a_A between all possible pairs of base detectors to find the stopping point. Experiments reveal a useful strategy: if the distribution of a_A's is skewed towards higher agreement rates, then the error estimates of the base detectors tend to be more accurate. Intuition is, it is unlikely that most pairs would have high agreement and yet agree on the wrong labels. Therefore, we propose to track the agreement rates and their distribution to decide when to stop iterating CARE.

Specifically, we use the area under the curve (*auc*) of the complementary cumulative distribution function (*ccdf*) of a_A's as the quantitative measure to decide the stopping point. The *auc* of *ccdf* is large if the distribution of a_A's is skewed towards higher agreement rates and vice versa. We assume that as CARE sequentially progresses over iterations, the base detectors improve, and hence the *auc* of *ccdf* for pairwise agreement rates gets larger. However, if at any iteration $t+1, t \in [0, MAXITER]$, the $auc(t+1)$ falls below the average by more than the standard deviation of $auc(0, \ldots, t+1)$, the sequential ensemble stops and returns the result at iteration t or otherwise iterates until $MAXITER$ and returns the final result.

3.4 Reducing Bias and Variance with CARE

According to [68], ensembles with feature-bagged base detectors and with variable sampling tend to reduce variance. In this section, we provide quantitative results through experiments on synthetic datasets to show that filtering top T outliers and probability sampling in our sequential ensemble reduce bias along with variance. To present the bias-variance reduction quantitatively, we design five procedures. For each synthetic dataset, we use a data generation model $\mathbb{M}$ to create R training datasets $D_i,\ i = 1 \ldots R$ of size $m = 210$ (200 inliers and 10 outliers) and 10 test datasets $D_j^{Test},\ j = 1 \ldots 10$ of size $n = 1000$ by randomly drawing points from $\mathbb{M}$. Bias and variance of different procedures for different values of k (i.e., # nearest neighbors) for a test data D_j^{Test} are calculated w.r.t. the training data $D_i^{'},\ i = 1 \ldots R$ sampled from D_i as follows:

$$bias = \sqrt{\frac{\sum_{x=1}^{n} (f^*(x) - \overline{f}(x))^2}{n}} \tag{3.8}$$

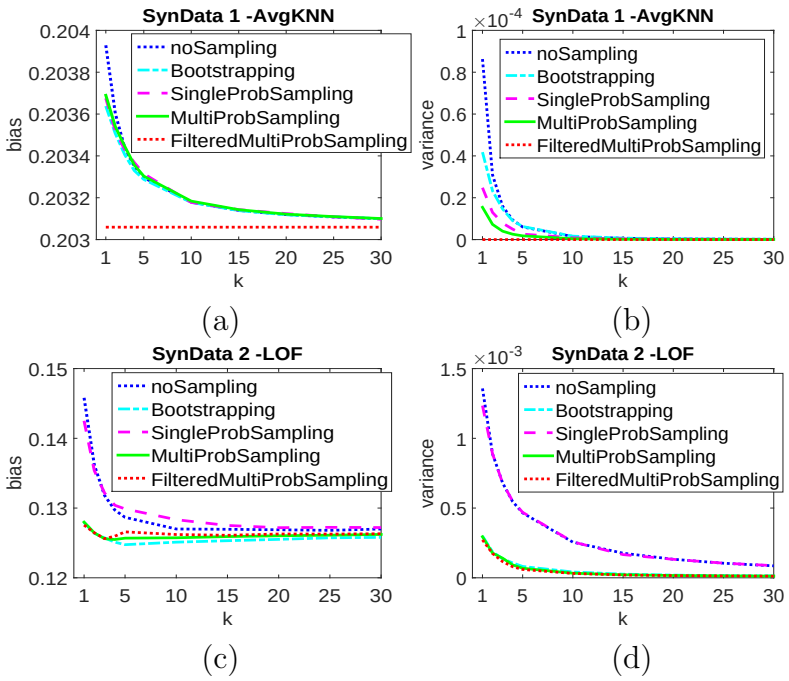


Figure 3.3: bias (left) and variance (right) vs. k (avg'ed over 10 test datasets) on two synthetic datasets. Notice that our approach (red) w/ probability sampling after top outliers being filtered reduces both bias and variance.

$$var = \frac{\sum_{x=1}^{n} \sum_{i=1}^{R} (f(x, D_i^{'}, k) - \overline{f}(x))^2}{n \times R} \tag{3.9}$$

Here, $\overline{f}(x) = \frac{\sum_{i=1}^{R} f(x, D_i^{'}, k)}{R}$, $f^*(x)$ is the actual label of data point $x \in D_j^{Test}$, and $f(x, D_i^{'}, k)$ is the normalized outlierness score of x w.r.t. sampled training set $D_i^{'}$ for k nearest neighbors. For each procedure we design a different approach for sampling $D_i^{'}$. These five different procedures are briefly described as follows:
(i) *noSampling*: $D_i^{'} = D_i$, (ii) *Bootstrapping*: sampling m times (w/ replacement) from D_i to get $D_i^{'}$, (iii) *SingleProbSampling*: probability sampling on $f(D_i, D_i, k)$ for a single iteration to get $D_i^{'}$, (iv) *MultiProbSampling*: probability sampling on $f(D_i, D_i^{'}, k)$ for multiple (i.e. 10) iterations where $D_i^{'} = D_i$ initially, and (v) *Filtered-MultiProbSampling*: filtered (top T outliers removed from D_i) probability sampling on $f(D_i, D_i^{'}, k)$ for multiple iterations (i.e. 10) where $D_i^{'} = D_i$ initially (our proposed approach).

In this section, we provide results on only two synthetic datasets (20 dimensional) for brevity, where the inliers are drawn from a mixture of Gaussian distributions and outliers are drawn from (i) power law, and (ii) uniform distribution. Figure 3.3 shows bias (left) and variance (right) vs. k, where for the top two plots

(i.e. (a), (b)) AvgKNN is used to calculate $f(x, D_i', k)$, and for the bottom two plots (i.e. (c), (d)) LOF is used. We can see from the figure that the curve for MultiProbSampling (green) is below the noSampling (blue) as well as the SingleProbSampling (magenta) curve, showing that probability sampling in multiple iterations helps to reduce both bias and variance. We also see that the FilteredMultiProbSampling (red) reduces bias further thanks to the filtering of top T outliers. Moreover, removing top outliers appears to also reduce variance as the red curve is below all the others in both (b) and (d).

3.5 Limitations of CARE

In this era of big data solutions, having a quadratic time framework (e.g., CARE) to solve anomaly mining problem is not feasible. Now-a-days, most of the applications produce massive amounts of data having millions or billions of points in high dimension. Hence, for huge datasets $O(n^2)$ solution is in-feasible. Moreover, CARE inherits the limitations associated with the base detectors LOF and AvgKNN. AvgKNN is unable to detect local as well as clustered anomalies in dense clusters, if the size of k is less than the corresponding cluster size. LOF is very much sensitive to the value of $MinPts$ (minimum number of objects) [72] and the cost of the algorithm is directly related to it. If we consider this value to be equal to the size of all the training data points, then we get better result with a slower algorithm. Although, the two phase aggregation of the base detector results reduce the error of overall framework, improved base detectors will improve CARE further.

3.6 Improved Scalability of CARE

In CARE, we utilize the proximity based base detectors and all these proximity based methods (e.g., LOF,AvgKNN) need to find k nearest neighbors by computing *Euclidean* distances between each pair of data points. Hence, the quadratic running time of the base detectors, specifically, the $O(n^2)$ nearest neighbor search utilized in LOF/AvgKNN makes CARE unacceptable for anomaly mining in massive datasets. One solution to this $O(n^2)$ nearest neighbor search approach is to use a indexing based method, e.g., kd-tree [86] to find the nearest neighbors. For low to medium dimension datasets having $n >> 2^d$, kd-tree is suitable. It has preprocessing time of $O(ndlogn)$ and query time of $O(2^d logn)$ for a single point, as analysis suggests that for k nearest neighbor search, number of nodes visited is proportional to $logn$ and number of distance calculations be approximately $k2^d$. Hence, in massive datasets where $2^d >> n$, using kd-tree is doing no better than brute-force sequential ap-

proach. Although, in CARE we are incorporating feature bagging in base detectors in the range $[d/2, d-1]$, but this range could be still high for very large d.

Therefore, to improve the scalability of CARE we incorporate ϵ-approximate nearest neighbor search (ϵ-NNS, defined in 3.1) using *locality sensitive hashing* (LSH) based on random hyperplane [87], as determining an approximate nearest neighbor should suffice for most practical purposes. We call this faster version as FastCARE. The preprocessing time of LSH is $O(nL)$, where L (e.g., = 10) is the number of hash tables and query time per point is $O(dn^{\frac{1}{1+\epsilon}})$. In CARE, we use ϵ-NNS in both LOF and AvgKNN for k nearest neighbor search. Moreover, for LOF we incorporate approximate reachability distance calculation with in a range of values for $MinPts$ (e.g. [30, 50]). Intuitively, the error introduced by the approximate approaches are reduced by the sequential ensemble framework of CARE.

Definition 3.1. ϵ-NNS: Given a set of $D(n \times d)$ points, preprocess D so that it returns a point $p \in D$ for any given query point q, such that $dist(q,p) \leq (1+\epsilon)dist(q,D)$, where $dist(q,D)$ is the distance of q to its closest point in D.

3.7 Isolation based CARE (iCARE)

According to our evaluation in Section 3.8.2, the closest competitor of CARE is isolation forest (iF), as it is superior in terms of time complexity and very close in terms of accuracy (based on results over 16 real-world datasets). However, for few datasets CARE performs much better than iF, and for few others the opposite is true. This motivates us to search for the limitations of both iF and CARE for datasets having different types of outliers, with a goal to improve CARE further. In Sectio 3.5, we described the limitations of CARE. According to [88], iF has weakness in detecting some specific types of anomalies, e.g., (i) local anomalies, (ii) anomalies with low relevant dimensions, (iii) global anomalies that exist in-between axis-parallel normal clusters, and (iv) anomalies in multimodal datasets.

To overcome these limitations of iF, Bandaragoda *et al.* designed an isolation based nearest neighbor ensemble (iNNE) [89]. In iNNE, for a dataset $D(n \times d)$, nearest neighbor distance $r : r = dist(p,q)$ (q is nearest neighbor of p) is used to isolate a sample point p in a d dimensional hypersphere, where r is the radius of the corresponding hypersphere. In the preprocessing stage of iNNE, a very small sample size ψ (e.g., = 2) is used to create $t(=25)$ random subsamples of points to isolate them in hyperspheres. The preprocessing step takes $O(t\psi^2)$. In the evaluation stage, each point $i \in D$ is evaluated against t sets of hyperspheres in iNNE to get isolation scores which are averaged to get the final outlier score. The evaluation step takes $O(nt\psi)$ time. Although iNNE is designed to overcome the limitations of iF, iNNE

itself has some limitations as follows.

- Curse of dimensionality as all the dimensions are used to create the isolation hyperspheres. Specially, when there are less number of relevant dimensions.
- Existence of interfering outliers in the initial subsamples at the preprocessing stage.
- Highly dependent on subsample size ψ.

Therefore, by addressing the limitations of CARE, iF, and iNNE, we design a new approach iCARE build on CARE with a goal to overcome all those limitations. Following are the improvements we incorporate in CARE to design iCARE,

- Replacing LOF/AvgKNNwith iNNE as base detectors, where iNNE overcomes the limitations of iF and CARE.
- The feature bagging step of CARE solves the dimensionality problem of iNNE.
- The filtered probability sampling step of CARE reduces the number of interfering anomalies in the initial subsample of iNNE.
- At the first step of iCARE ψ_0 is set to 2 to create $t(=25)$ random subsamples of isolation hyperspheres. Here, the probability of selecting an outlier in a subsample at the preprocessing stage is very low as percentage of outliers is very small compared to inliers in most of the datasets.This step also reduces the complexity of the first iteration of CARE from $O(bn^2)$ to $O(2bt(2+n))$ in iCARE, where b $(=100)$ is the number of feature bagged detectors and n is the total number of points in the data.
- The variable sampling of CARE at each iteration helps to choose different sample sizes ψ_i in sequential iterations as follows, $\psi_i = \underset{j\in[2^1,2^2,\ldots,2^9]}{\operatorname{argmin}} |j - \frac{|S_i|}{10}|$, where $|S_i|$ is the variable sample size at ith iteration. This step removes the dependency of iNNE on choosing a specific ψ based on the type of anomalies present in data which is unknown for unsupervised case.
- We change the range of variable sample size of CARE in iCARE, we select l uniformly at random between $\min\{1-\frac{T}{n}, \frac{20}{n}\}$ and $\max\{1-\frac{T}{n}, \frac{5120}{n}\}$ to align this range with the range of subsample sizes $\psi_i \in [2^1, 2^2, \ldots, 2^9]$, where n is the number of points in the original dataset and T is the number of filtered outliers.

[1]https://www.ipd.kit.edu/~muellere/HiCS/
[2]http://goo.gl/mGg8ti?gdriveurl
[3]http://yann.lecun.com/exdb/mnist/

Table 3.1: Real-world datasets used for evaluation, where d is data dimensionality, and % indicates the % of outliers.

Dataset	**#Pts** n	**Dim.** d	% **Outlier Class**
Lympho	148	18	classes 1,4 (4.1%)
WBC	278	30	21 sampled malignant class (5.6%)
Glass	214	9	class 6 (4.2%)
Vowels	1456	12	50 sampled class 1 (3.4%), classes 6,7,8, inliers
Cardio	1831	21	176 sampled pathologic (9.6%), normal inliers
Thyroid	3772	6	from [90][1] (2.5%)
Musk	3062	166	classes 213,211 (3.2%) classes j146,j147,252 inliers
Optdigits	5216	64	150 sampled digit 0 (3%)
Satimage-2	5803	36	71 sampled class 2 (1.2%)
Letter	1600	32	from [71] [2] (6.25%)
Pima	768	8	pos class (35%)
Satellite	6435	36	3 smallest classes (32%)
Breastw	683	9	malignant class (35%)
Arrhythmia	452	274	classes 3,4,5,7,8,9,14,15 (15%)
Ionosphere	351	33	bad class (36%)
Mnist[3]	7603	100	700 sampled digit 6 (9.2%), digit 0 inliers
HAR	5272	561	sitting, standing, and laying activities are down sampled as outliers (11.4%)
isolet	730	617	10 instances of class 'Y' (1.4%)
mfeat	410	649	10 instances of digit '0' (2.4%)
shuttle	49097	9	Largest Rad flow class as inliers and rest as outliers (7.0%)
mulcross	262144	4	2 outlier clusters (10.0%)

3.8 Evaluation

3.8.1 Datasets

We evaluate CARE on 21 different real-world outlier detection datasets (all available at http://odds.cs.stonybrook.edu/#table1) mostly from the UCI ML repository [70]. Table 3.1 provides the summary of the datasets used in this work. The first 9 datasets, Letter, and the following datasets are respectively obtained from [68], [71]

and [89, 91]. Detailed description of these datasets are provided in Appendix A.

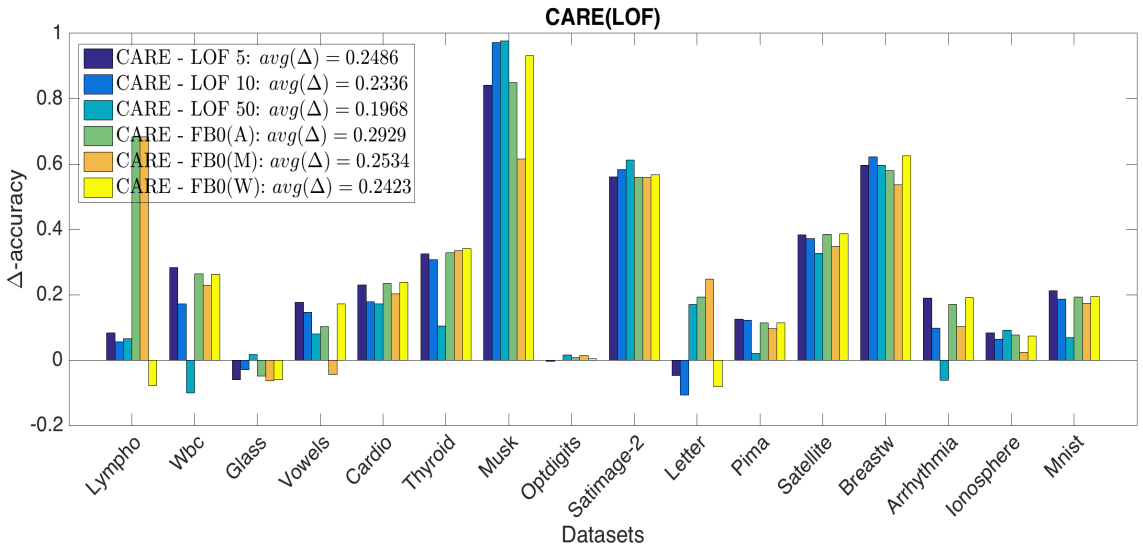


Figure 3.4: *Delta* AP values from CARE(LOF) to LOF based baseline approaches vs. datasets plot. Notice that 14/16 datasets provide improvement over most of the baseline approaches.

3.8.2 Results

We first compare CARE with baseline approaches and state-of-the-art ensembles using the first 16 datasets from Table 3.1 in the following two sections.

CARE vs state-of-the-art baselines

We compare CARE with simple LOF and AvgKNN based baseline approaches; using $k = \{5, 10, 50\}$, as well as non-sequential feature bagging (FB0) approaches with three types of aggregation; average (A), maximum (M), and weighted (W). Figure 3.4 shows the Δ Average Precision (AP: area under the precision-recall curve) values from CARE(LOF) to these six state-of-the-art baselines all using the LOF algorithm. That is, the bars depict $\text{AP}^{\text{CARE}} - \text{AP}^{\text{baseline}}$. We refer to Figure 3.5 for the original AP values that CARE(LOF) and CARE(AvgKNN) achieve on the datasets. Results show that CARE outperforms all the base detectors on 9/16 datasets, and more than half of them on 14/16 datasets. Negative Δ values are much smaller as compared to positive ones, which indicates that in cases where CARE is not better than the baselines, it remains close. In the legend of the figure, we provide the overall ΔAP values averaged across all the datasets and positive values indicate

that CARE performs better than the individual baselines on average. Similarly, Figure 3.6 contains the ΔAP values from CARE(AvgKNN) to six baselines, which this time use AvgKNN based subroutines. Again, the average Δ values (in the legend) across different datasets indicate that CARE outperforms the individual baselines on average. In cases where CARE falls shorter it often remains close to the baselines (note the relatively much smaller negative Δ's). From these two figures we also conclude that CARE(LOF) provides greater improvement over the baselines compared to CARE(AvgKNN). However, in Figure 3.5 comparing the AP values for CARE(LOF) and CARE(AvgKNN) we can see that for 10/16 datasets CARE(AvgKNN) performs significantly better.

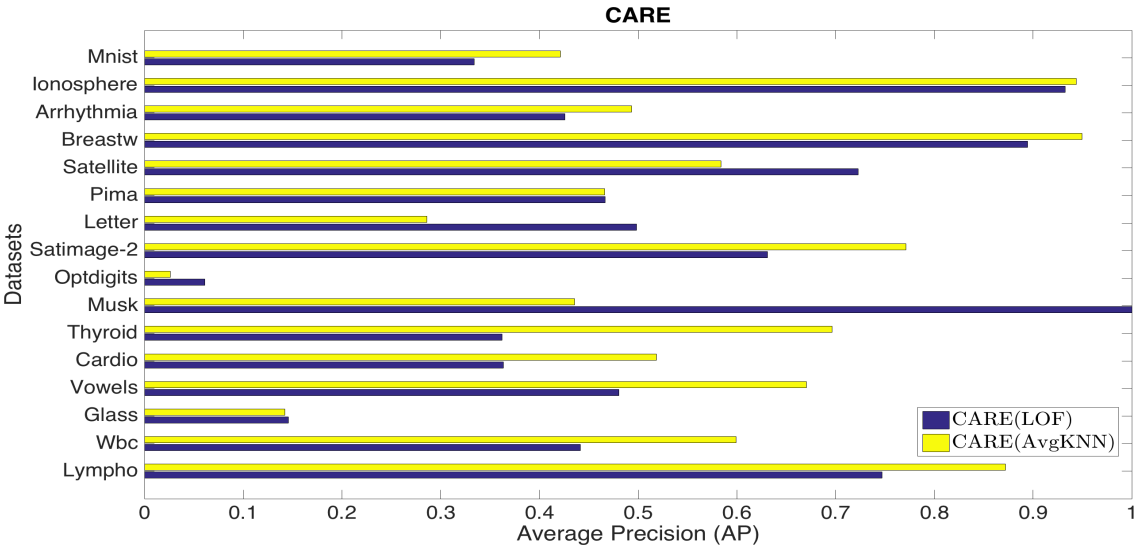


Figure 3.5: Average Precision (AP) of CARE across datasets for both LOF and AvgKNN based base detectors. CARE(AvgKNN) performs better than CARE(LOF) on 10/16 datasets.

CARE vs state-of-the-art ensembles

Next we compare CARE with the existing state-of-the-art outlier ensemble methods, including Aggarwal and Sathe's variable sampling (VS), rotated bagging (RB), and variable rotated bagging (VR) approaches [68], Zimek *et al.*'s subsampling approach [75], as well as the Isolation Forest (iF) ensemble of Liu *et al.* [79]. We employ $b = 100$ base detectors for each of these existing ensemble approaches such that they are comparable with CARE. We present the ΔAP from CARE(LOF) to these six existing state-of-the-art outlier ensembles using the LOF algorithm (except for iF) in Figure 3.7. For Zimek's subsampling approach we only present the

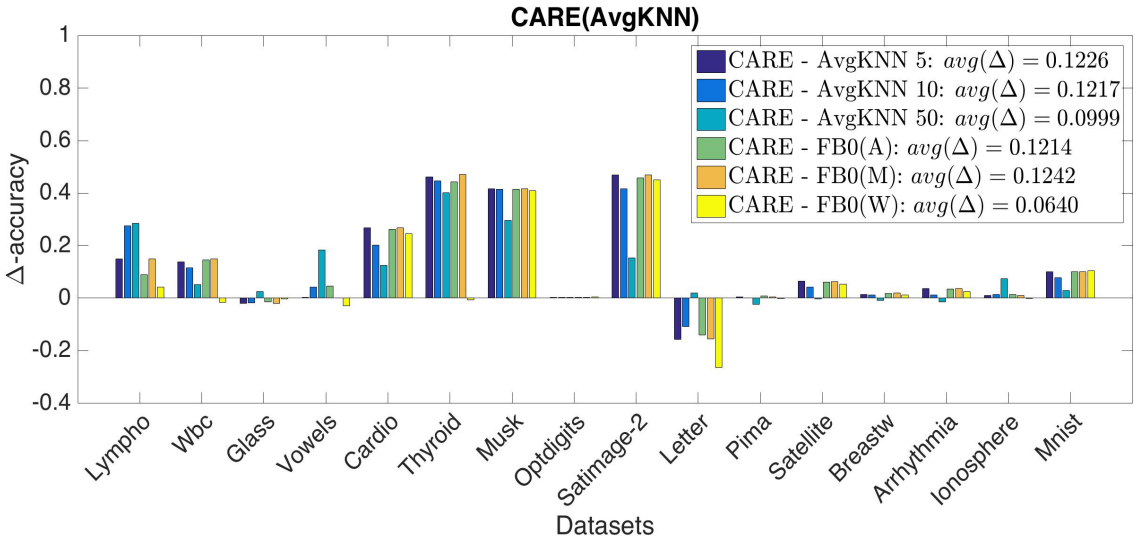


Figure 3.6: Δ AP values from CARE(AvgKNN) to AvgKNN based baselines. CARE improves over more than half of the baselines on 14/16 datasets.

results for sample sizes 10% and 50% (Z10, Z50). In Figure 3.7, we can see that the performance of CARE(LOF) and VS are close with $avg(\Delta) = 0.0038$ across all the datasets. Notice that CARE mostly improves over Z50 and RB. Although iF is little better than CARE(LOF) with $avg(\Delta) = -0.0050$, for some datasets e.g., Vowels and Letter where iF performs poorly with AP values 0.1341 and 0.0929 respectively, CARE(LOF) provides 2.6× improvement with AP value 0.4803 for Vowels, and 4.4× improvement with AP value 0.4986 for Letter. Moreover, we note that the magnitude of positive Δ values are larger than the negative ones on average. This indicates that CARE(LOF) provides major improvement in cases when it is the winner and performs similarly to existing ensembles in other cases. Finally, Figure 3.8 shows the corresponding results for CARE(AvgKNN). Positive average ΔAP values across all datasets show that CARE provides significant improvement when it outperforms an existing ensemble and falls short by a small margin in other cases. iF outperforms CARE(AvgKNN) significantly on Musk, which has a dense cluster of outliers that avoid detection by nearest neighbor distance based methods. We also find that none of the existing methods perform well on Optdigits and Glass, where further investigation is needed to understand the type of outliers that they exhibit.

Furthermore, in Figure 3.9 we present the performance in terms of AP for the above six state-of-the-art ensemble approaches along with CARE and baselines employing $k = 5$, with the LOF algorithm (left) as well as the AvgKNN algorithm (right) on two real-world datasets, i.e. Thyroid and Musk (for brevity). The boxplots in each figure are shown at varying levels of subsampling rates. In this figure

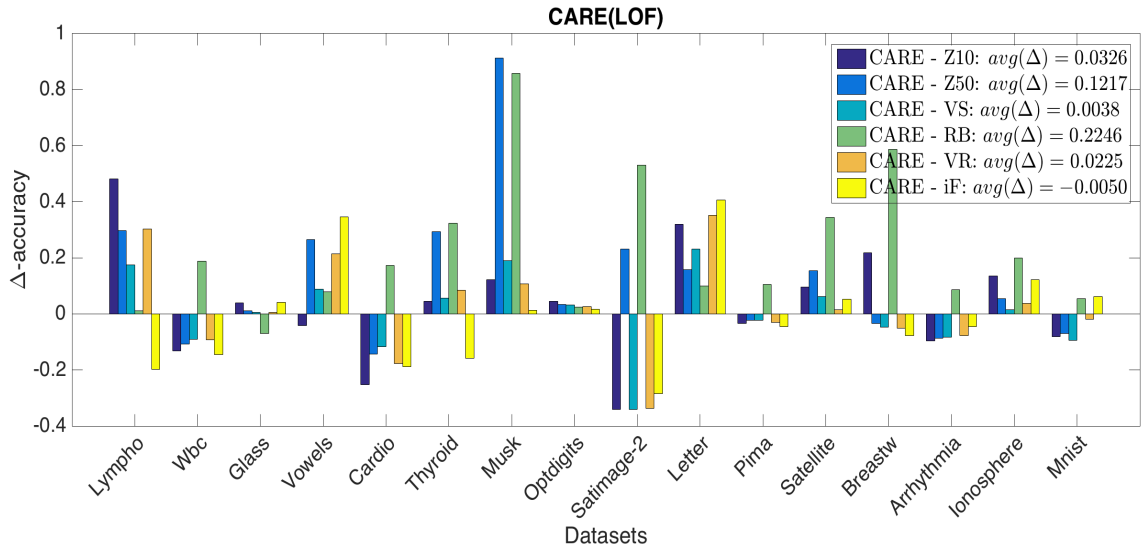


Figure 3.7: ΔAP from CARE(LOF) to LOF based state-of-the-art ensemble approaches on all the datasets. Notice that CARE outperforms existing ensembles significantly on several datasets and achieves comparable performance otherwise. $avg(\Delta)$'s in the legend denote average of ΔAP values across datasets.

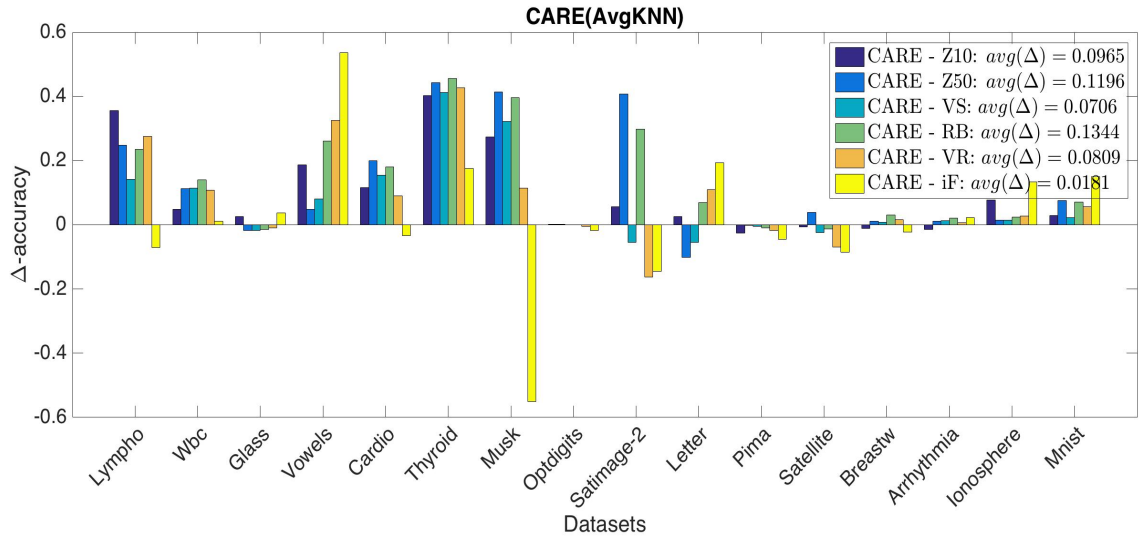


Figure 3.8: Δ AP values from CARE(AvgKNN) to AvgKNN based state-of-the-art ensemble approaches. Note the generally large positive and otherwise small negative values across datasets.

we can see that even if the baseline (LOF/AvgKNN) fails to perform well on the whole data, subsampling (both variable, e.g., CARE, VS, VR and fixed, e.g., Z10, iF) helps

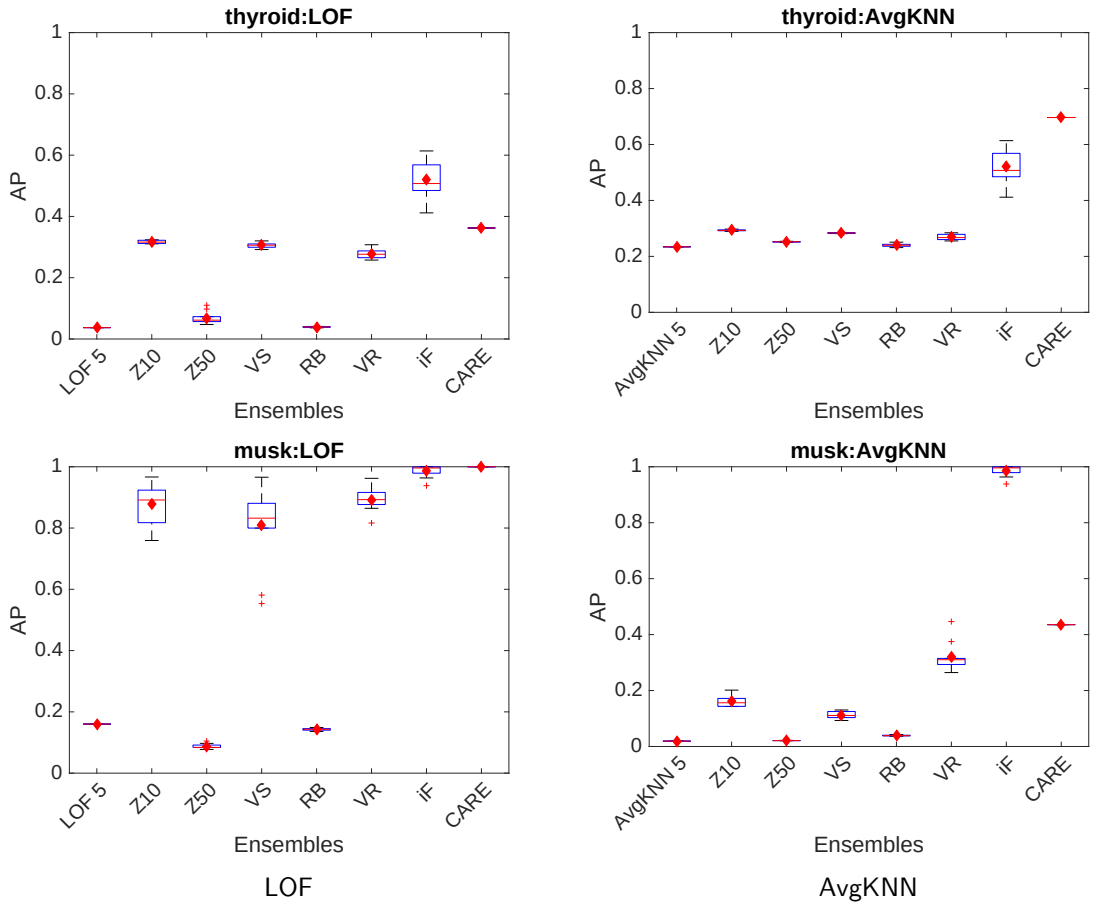


Figure 3.9: Performance of existing state-of-the-art outlier ensembles (over 10 runs) along with CARE and LOF/AvgKNN for $k = 5$ for two real-world datasets (i.e. Thyroid and Musk). Notice that CARE outperforms the baselines and existing outlier ensembles either with LOF or AvgKNN.

to improve performance, notice accuracies of Z10, VS, VR, iF, and CARE. Further investigation shows that CARE(LOF) performs better than CARE(AvgKNN) on Musk for having a dense cluster of outliers in between two bigger clusters of inliers, as such not filtering most of the outliers during FVPS approach in CARE(AvgKNN). On the other hand, CARE(AvgKNN) performs better than CARE(LOF) on Thyroid as the density of the region where outliers reside are close to that of the inliers, as such FVPS approach with filtering and probability sampling at various rates helps the AvgKNN base detectors. But for CARE(LOF) as the data model becomes sparse due to sampling, it sometimes detects inliers as unusual points and vice versa.

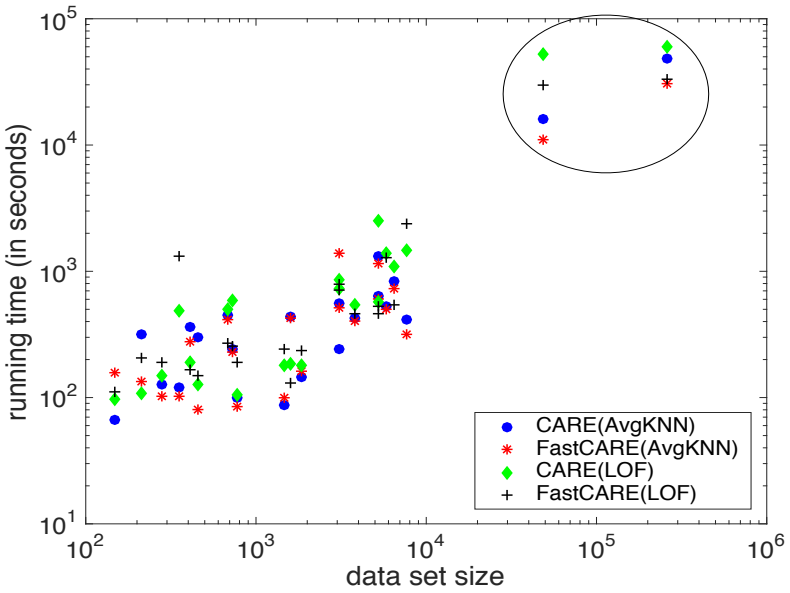


Figure 3.10: Dataset size vs running time in seconds for CARE and FastCAREacross 21 real-world datasets.

CARE vs FastCARE

We compare CARE and FastCARE with respect to both accuracy and running time over 21 real-world datasets. We calculate the time in number of seconds. In our experiments, for the large datasets, i.e., shuttle and mulcross we use kd-Tree indexing method for nearest neighbor distance calculation and approximate reachability distance for CARE(LOF) to reduce memory requirement. For the other datasets only FastCARE(LOF) uses approximate reachability distance with in a range of *MinPts* (e.g., [30 50]) values. Figure 3.10 shows that the speed is only improved for very large datasets in FastCARE. In Figure 3.11, we present the average precision values of all the datasets for CARE(LOF), CARE(AvgKNN), FastCARE(LOF), and FastCARE(AvgKNN). It shows that among 21 datasets the performance of 16 datasets drop from CARE(LOF) to FastCARE(LOF), and performance of 11 datasets drop from CARE(AvgKNN) to FastCARE(AvgKNN). Hence, we design iCARE with a goal to improve both accuracy and speed. In the following section we provide evaluation of iCARE.

iCARE vs CARE and state-of-the-art ensembles

In this section, we compare iCARE with CARE(LOF), CARE(AvgKNN) and two state-of-the-art ensembles, isolation based nearest neighbor ensemble (iNNE) [89] and

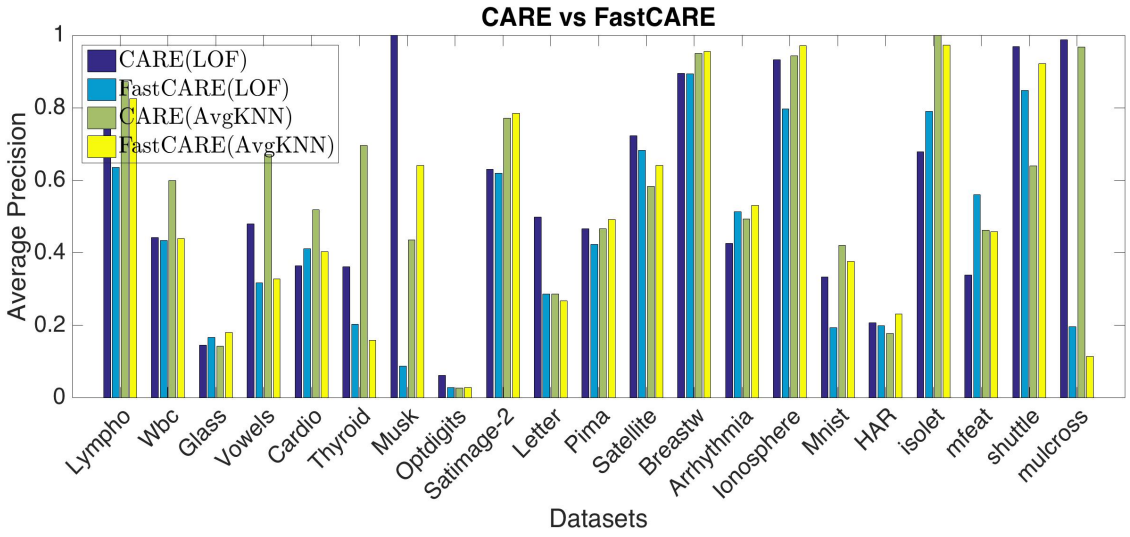


Figure 3.11: AP values of CARE(LOF/AvgKNN) and FastCARE(LOF/AvgKNN). Note CARE performs better than FastCARE for most of the datasets.

isolation forest (iF) [79]. We present the Δ Average Precision from iCARE to these four outlier ensembles in Figure 3.12. We employ $b = 100$ base detectors for each of these ensemble approaches for compatibility. For iNNE subsample size ψ is employed in the range of $[2^1, 2^2, \ldots, 2^9]$ and the best result is reported. In Figure 3.12, we can see that the performance of iCARE improves over all its competitors with positive $avg(\Delta)$ values across all datasets which means iCARE provides significant improvement when it outperforms an existing ensemble and falls short by a small margin in other cases. Among 21 datasets, only on three, e.g., Vowels, Thyroid, and Letter datasets CARE performs better than iCARE. On higher dimensional data HAR and mfeat iCARE provide large improvement over all the other methods. For Vowels, Letter, HAR and mfeat iCARE provide significant improvement over iF. Hence, we can state that iCARE performs better than CARE and other state-of-the-art ensembles in terms of accuracy. Figure 3.12 also contain the absolute AP values (below the bars) of iCARE(LOF) for all datasets. Furthermore, our goal is to make iCARE faster than CARE to be able to use for large datasets. Figure 3.13 shows the comparison of the running time in seconds between CARE(LOF/AvgKNN) and iCARE. Specifically, for large datasets, e.g., shuttle and mulcross, iCARE is significantly faster than CARE. For shuttle iCARE is 23.4× faster than CARE(LOF) and 7.2× faster than CARE(AvgKNN). Again, for mulcross iCARE is 10× faster than CARE(LOF) and 8× faster than CARE(AvgKNN).

In real-world high dimensional datasets it is hard to understand the types of outliers they contain. Hence, for further investigation on the performance of iCARE,

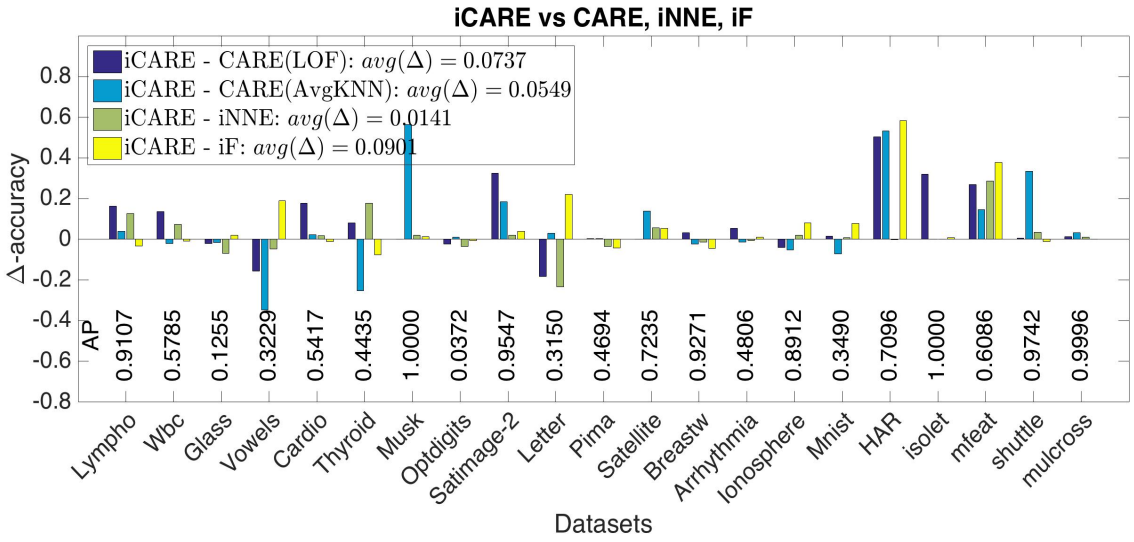


Figure 3.12: Δ AP values from iCARE to CARE(LOF), CARE(AvgKNN), iNNE, and iF approaches. Note the generally large positive and otherwise small negative values across datasets.

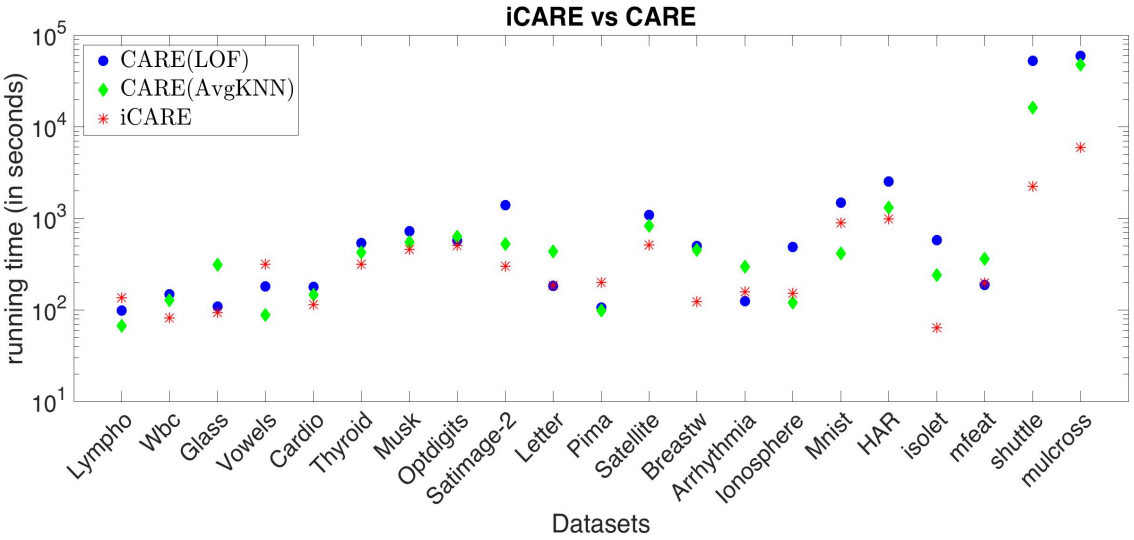


Figure 3.13: iCARE vs CARE running time in seconds for 21 datasets. Note the time gap between CARE(LOF) and iCARE for two large datasets shuttle and mulcross.

we generate two synthetic datasets having different types of outliers and compare with the existing methods in the following case studies.

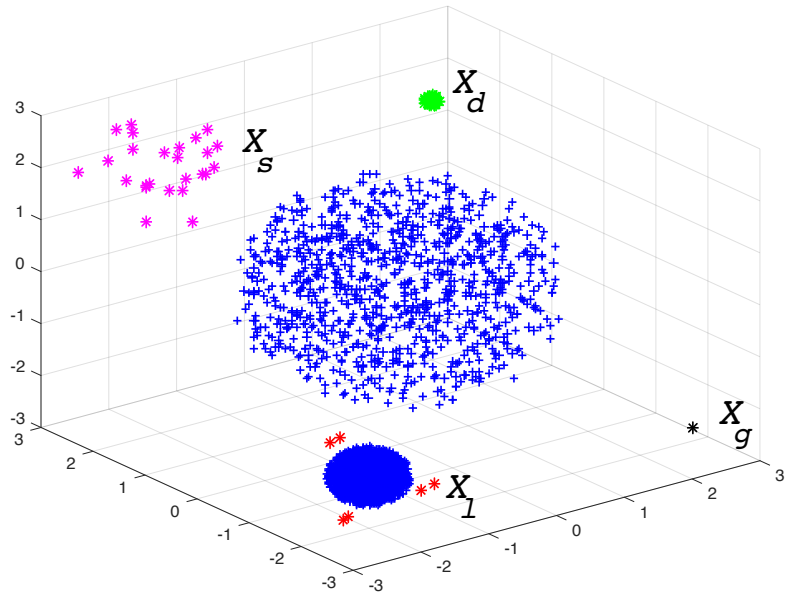


Figure 3.14: The synData1 dataset containing different types of outliers in a dataset of more than 3000 instances. (i) dense cluster: 2000 instances, (ii) sparse cluster: 1000 instances, (iii) X_d: 50 instances (a dense outlier cluster), (iv) X_s: 25 instances (a sparse outlier cluster), (v) X_l: 5 instances (local outliers) and (vi) X_g: 1 instance (a global outlier).

Case Studies:

We generate two synthetic datasets, e.g., synData1 and synData2, having local, global and clustered (both sparse and dense) outliers. synData1 is a 3 dimensional dataset whose distribution of inliers and different types of outliers are presented in Figure 3.14. In this dataset, there are two uniformly distributed normal clusters where one is sparse and the other one is dense. It also contains two clusters of outliers, where one is a dense cluster (X_d) and the other one is a sparse cluster (X_s). In addition, it has a global outlier (X_g) and six local outliers (X_l) which are place around the dense normal cluster. synData2 has the same characteristics as synData1 having different types of outliers, but in a higher dimension. synData2 has 25 dimensions in which 5 of them are irrelevant dimensions. This dataset is generated to show that both iNNE and iF are hurt by the presence of irrelevant dimensions in the data.

We evaluate iCARE, CARE(LOF), CARE(AvgKNN), iNNE and iF on this synthetic datasets and their performance is presented in Table 3.2. This table shows that for datasets with different types of outliers iCARE performs better than all of its competitors. For synData1, iF performs poorly compared to all other approaches.

Here, iF is unable to detect all the local outliers, and all the densely clustered outliers. CARE was unable to detect few densely clustered outliers and all the local outliers in both datasets. On both the datasets, iNNE has the closest performance to iCARE, where iCARE wins with the feature bagging and outlier filtering steps.

Table 3.2: AP values of iCARE, CARE, iNNE, and iFon synData1 and synData2.

datasets	Ensembles				
	iCARE	CARE(LOF)	CARE(AvgKNN)	iNNE	iF
synData1	0.9990	0.9323	0.9366	0.9753	0.3238
synData2	1.0000	0.9330	0.9287	0.9954	0.9320

3.9 Summary of Contributions

In this work, we designed CARE, a new sequential ensemble approach for outlier mining with a goal to achieve low detection error through reduced variance and bias. Two main components of CARE are its parallel and sequential building blocks. The former helps reduce variance by a weighted combination of multiple base detectors. Detector weights are derived from their error rates that are estimated through their relation to pairwise agreement rates. On the other hand, the sequential component is designed to reduce bias. It utilizes results from previous iterations and a new sampling strategy FVPS to weed out top outliers so as to construct a more robust data model based on which outlierness scores are computed. We evaluate our method on 21 real-world datasets. Extensive experiments validate that CARE provides significant improvement over the baseline methods as well as the state-of-the-art outlier ensembles when it is the winner and performs close enough otherwise. Furthermore, we improve CARE and design iCARE which provides better performance and running time for large datasets.

Chapter 4

Multimodal Learning in Collective Opinion Spam Detection

Information in the real-world comes from multiple input channels where each type of input is characterized by very distinct properties which make it difficult to ignore the fact that they are very different. Useful representations can be learned about such multimodal data by combining the multiple input modalities into a joint representation that captures the real-world notion that the data corresponds to. The need for this joint representation further increases the challenge of anomaly mining in multimodal data. In this chapter, we present a multimodal learning approach and apply it to solve opinion spam detection problem for online review network.

Online product and business reviews are increasingly valuable sources for consumers to make decisions on what to purchase, where to eat, which care provider to see, etc. They are powerful since they reflect testimonials of "real" people, unlike e.g., advertisements. Financial incentives associated with reviews, however, have created a market of (often paid) users to fabricate *fake reviews* to either unjustly hype (for promotion) or defame (under competition) a product or business, the activities of whom are called *opinion spam* [3].

The problem is surprisingly prevalent; it is estimated that more than 20% of Yelp's reviews are fake [92], following a constant growth [93], while one-third of all consumer reviews on the Internet are estimated to be fake [94]. While widespread, it is a hard and mostly open problem. The key challenge is obtaining large ground truth data to learn from, however manual labeling of reviews is extremely difficult by merely reading them, where humans are only slightly better than random [95], unlike e.g., labeling email spam. This renders supervised methods inadmissible to a large extent.

Since the seminal work of Jindal *et al.* on opinion spam [3], a variety of ap-

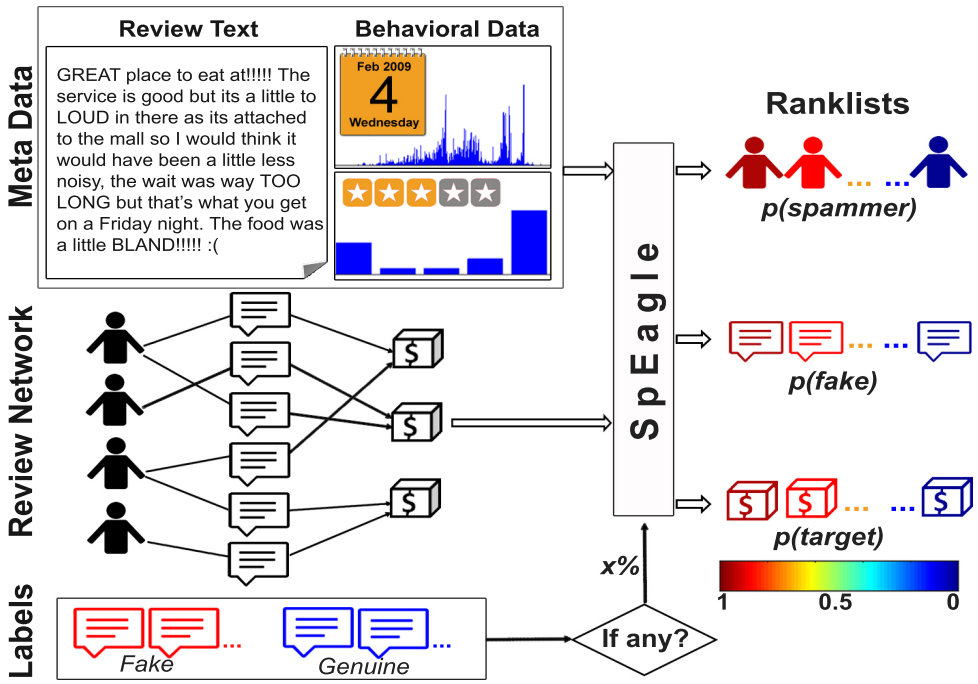


Figure 4.1: SpEagle collectively utilizes both metadata (review text, timestamp, rating) and the review network (plus available labels, if any) under a *unified* framework to rank all of users, reviews, and products by spamicity.

proaches have been proposed. At a high level, those can be categorized as linguistic approaches [95–97] that analyze the language patterns of spam vs. benign users for psycholinguistic clues of deception, behavioral approaches [3, 4, 98–102] that utilize the reviewing behaviors of users (e.g., temporal and distributional footprints), and graph-based methods [103–106] that leverage the relational ties between users, reviews, and products with minimal to no external information. Current approaches can also be grouped as those that detect fake reviews [3, 95, 96, 98, 100, 102, 105, 106], spam users [4, 101, 103, 104, 106], or spam user groups [107, 108]. (see §4.1 for details)

These have made considerable progress in understanding and spotting opinion spam, however the problem remains far from fully solved. Hence, we capitalize on a prior work [103] to design a new method, SpEagle (for Spam Eagle), that can utilize clues from all of *metadata* (text, timestamp, rating) as well as *relational data* (review network), and harness them collectively under a unified framework to spot spam users, fake reviews, as well as targeted products. Moreover, SpEagle can seamlessly integrate labels on any subset of objects (user, review, and/or product) when available, without any changes in its algorithm (see Figure 4.1). We summarize the contributions of this work as follows.

- We design SpEagle, a new approach for the opinion spam detection problem, which *ties together* relational data with metadata, i.e., it utilizes all of graph, behavioral, and review content information collectively.
- SpEagle considers the user–review–product graph to formulate the problem as a network-based classification task, in which users are labeled as spammer or benign, reviews as fake or genuine, and products as target or non-target (of spam) (§4.2.2). This formulation accepts prior knowledge on the class distribution of the nodes, which is estimated from metadata. In particular, SpEagle uses the metadata (text, timestamps, ratings) to design and extract indicative features of spam which are converted into a spam score to be used as part of class priors (§4.2.2).
- SpEagle works in a completely *unsupervised* fashion. However, it is amenable to easily leverage label information (when available). As such, we introduce a *semi-supervised* version of our method called SpEagle$^+$, which accepts as input labels for a (small) set of user, review, and/or product nodes in the graph (§4.2.2). The integration of labels is efficient and seamless; it requires *no* training on the labeled data and the main inference steps of the algorithm remain intact (§21).
- SpEagle extracts features for all user, review, and product nodes using metadata. To reduce computational overhead, we investigate the effectiveness of our features and their categories (user vs. review vs. product and text vs. behavior), and design SpLite (for SpEagle-Light), which uses only a few review features (and completely avoids feature extraction for users and products) (§21).

We evaluate our method on three real-world datasets collected from Yelp.com, containing filtered (spam) and recommended (non-spam) reviews. To the best of our knowledge, our work provides the largest scale *quantitative* evaluation to date for the opinion spam problem. Our results demonstrate that (*i*) SpEagle outperforms several baselines and state-of-the-art techniques [103, 106], (*ii*) performance can be boosted significantly by SpEagle$^+$ with only limited supervision, and (*iii*) SpLite provides significant speed-up with only a slight compromise in detection performance (§4.3).

4.1 Related Work

Opinion spam is one of the new forms of Web-based spam, and has been the focus of academic research in the last 7-8 years. We organize the various approaches to

this problem into three groups: behavior-, language-, and graph-based.

Behavior-based approaches. The approaches in this category often leverage indicative features of spam extracted from the metadata associated with user behavior (e.g., rating distribution), review content (e.g., number of capital letters), and product profile (e.g., brand and price). The seminal work by Jindal and Liu [3] use supervised learning based on 36 such features on a (pseudo) ground truth dataset, constructed by labeling the duplicate reviews in an Amazon dataset as fake reviews. Li *et al.* [100] train semi-supervised models, and use the two views from reviews and users under a co-training framework to spot fake reviews. Jindal *et al.* [99] propose rule-based discovery of unusual patterns in review data associated with the rating and brand distribution of a user's reviews. Other work that study rating based behavior of users include [98] and [101]. More recently, Mukherjee *et al.* [4] utilize reviewing behaviors of users in an unsupervised Bayesian inference framework to detect opinion spammers. Xie *et al.* [102] monitor temporal behavior of products by tracking their average rating, review count, and ratio of singleton reviewers, to spot suspicious single-time reviewers. Those spammers are particularly challenging to detect, as they provide only a single review. Besides detecting individual spammers, there has also been work on identifying spammer groups through group-level behavioral indicators of spam [107, 108].

Language-based approaches. Methods in this category focus on the characteristics of language that the opinion spammers use and how it differs from the language used in genuine reviews. This line of work is also related to studies in deception [97]. Ott *et al.* [95] learn supervised models to detect deceptive reviews based on linguistic features of reviews as well as features borrowed from studies in psychology. Amazon Mechanical Turk has been employed to crowdsource fake reviews by paying anonymous online users to write fake hotel reviews. Feng *et al.* [96] investigates syntactic stylometry for deception detection, and show that features derived from context-free-grammar parse trees improve performance over shallow lexico-syntactic features. An investigation by Mukherjee *et al.* [109] analyzed the effectiveness of linguistic and behavioral clues on a Yelp dataset with filtered and recommended reviews, and found that linguistic features are not as effective and that Yelp's filter might be using a behavioral based approach.

Graph-based approaches. Besides behavioral and linguistic clues of spam, there is also the relational association between users, reviews, and products that several methods have leveraged. Wang *et al.* [106] consider the user-review-product network and define scores for trustiness of users, honesty of reviews, and reliability of products. These scores are formulated in terms of one another, and are computed by an iterative algorithm similar to HITS [110]. Akoglu *et al.* [103] proposed a detection framework based on Markov Random Field (MRF) models on the signed bipartite network of users and products, which are connected through positive or negative

review relations (signed edges). MRFs have also been utilized in [104] to model user-user relations that capture burst-membership, and in [108] to model user-user collusion relations as well as user-attribute ownership relations. Li *et al.* [105] construct a user-IP-review graph to relate reviews that are written by the same users and from the same IPs. All of these approaches model the fake review(er) detection problem as a collective classification task on these networks, and employ algorithms such as Loopy Belief Propagation (LBP) [111] or Iterative Classification Algorithm (ICA) [112] for inference. Jiang *et al.* [113] proposed a graph-based method to identify group spammers with extremely synchronized behavior in a graph. While they evaluated their method on social networks, it can be readily used for review networks.

4.2 Methodology

In this work, we build on FraudEagle framework [103] and extend it in two main directions; (1) we expand its graph representation (structure and semantics), and (2) we incorporate meta-information into its network-only solution.

In a nutshell, we formulate the spam detection problem as a network classification task on the user-review-product network. In this task, users are to be classified as *spammer* or *benign*, products as *targeted* or *non-targeted*, and reviews as *fake* or *genuine*. To aid the network classification, we utilize additional metadata (i.e., ratings, time stamps, and review text) to extract indicative features of spam, which we incorporate into the inference procedure. Our proposed method works in a completely unsupervised fashion, however it can easily accommodate labels when available. As such, it is amenable to semi-supervised detection.

In §4.2.1, we first introduce FraudEagle for completeness and to lay out the main framework that forms the foundation for our proposed method SpEagle, described in §4.2.2.

4.2.1 FraudEagle Framework

The network representation used by FraudEagle [103] is the user–product bipartite network with signed edges. The user-product network $G = (V, E^{\pm})$ contains N user nodes $U = \{u_1, \ldots, u_N\}$ and M product nodes $P = \{p_1, \ldots, p_M\}$, $V = U \cup P$, connected through signed edges $(u_i, p_j, s) \in E^{\pm}$. The edges capture 'review' relations, where each edge sign $s \in \{+, -\}$ depicts positive or negative review.

The goal of classification on a network is to assign a label to each node. In their formulation, the domain of class labels is $\mathcal{L}_U = \{benign, spammer\}$ for users and

$\mathcal{L}_P = \{good\text{-}quality, bad\text{-}quality\}$ for products. To formally define the classification problem, the network is represented as a pairwise Markov Random Field (MRF) [114]. An MRF model consists of an undirected graph where each node i is associated with a random variable Y_i that can be in one of a finite number of states, that is, classes. In pairwise MRFs, the label of a node is assumed to be dependent only on its neighbors and independent of all the other nodes in the graph. The joint probability of labels is then written as a product of individual and pairwise factors, parameterized over the nodes and the edges, respectively:

$$P(\mathbf{y}) = \frac{1}{Z} \prod_{Y_i \in V} \phi_i(y_i) \prod_{(Y_i, Y_j, s) \in E^{\pm}} \psi_{ij}^{s}(y_i, y_j) \tag{4.1}$$

where $\mathbf{y}$ denotes an assignment of labels to all nodes, y_i refers to node i's assigned label, and Z is the normalization constant. The individual factors $\phi : \mathcal{L} \rightarrow \mathbb{R}^+$ are called *prior* (or node) potentials, and represent initial class probabilities for each node, often initialized based on prior knowledge. The pairwise factors $\psi^s : \mathcal{L}_U \times \mathcal{L}_P \rightarrow \mathbb{R}^+$ are called *compatibility* (or edge) potentials, and capture the likelihood of a node with label y_i to be connected to a node with label y_j through an edge with sign s. To reduce the number of model parameters, the edge potentials are often shared (in this case, across all edges with the same sign). In case no external information or domain knowledge is available, one can set all the priors as unbiased/uninformative (i.e., equal probability for each class). Otherwise, one can estimate them for every node separately using external sources.

In FraudEagle, one of the design goals is to avoid the requirement for additional sources to estimate priors. This also facilitates generality, as external sources may differ across domains. Therefore, the priors are all set as unbiased to $\phi_i = \{0.5, 0.5\},\ \forall i \in V$. On the other hand, the compatibility potentials are ideally estimated from labeled data. In a problem setting such as opinion spam detection however, obtaining reasonable amount of labeled data is extremely difficult due to the challenges that human annotators face. Therefore, these parameters are set based on several intuitions reflecting the modus-operandi of spammers and otherwise normal behavior of regular users: (*i*) benign users often write positive reviews to good-quality products and negative reviews to bad-quality products, in contrast (*ii*) spammers more likely write positive reviews to bad-quality products (to hype) and/or negative reviews to good-quality products (to defame, e.g., under competition), (*iii*) spammers can also write genuine-looking reviews to camouflage their mis-activities (i.e., positive reviews to good-quality and negative reviews to bad-quality products), although (*iv*) the same (e.g., writing negative reviews to good-quality products) is less likely for benign users (but can happen depending on individual experience). Under these assumptions, the following parameter settings

are used, for a small ϵ value.[1]

$\psi^{s=+}$	Product	
User	*good*	*bad*
benign	$1-\epsilon$	ϵ
spammer	2ϵ	$1-2\epsilon$

$\psi^{s=-}$	Product	
User	*good*	*bad*
benign	ϵ	$1-\epsilon$
spammer	$1-2\epsilon$	2ϵ

Given the model parameters (ϕ_i, $\forall i \in V$ and ψ^s for $s \in \{+,-\}$), the task is to infer the maximum likelihood assignment of states (class labels) to the random variables associated with the nodes, in other words, to find the $\mathbf{y}$ that maximizes the joint probability of the network as given in Eqn. (4.1). This is the inference problem which is combinatorially hard. The enumeration of all possible assignments is exponential to the network size and thus intractable for large graphs. Exact inference is known to be NP-hard for general MRFs, where instead iterative approximate algorithms such as Loopy Belief Propagation (LBP) [111] are used. We describe the details of the inference procedure in the context of our proposed method below.

4.2.2 Proposed Method SpEagle

Besides the relational information between users and products, there exist a variety of metadata in review datasets. Those include the text content of reviews, timestamps, and star ratings. Earlier work have used metadata to design features that are indicative of spam [3, 95, 100, 102, 107]. FraudEagle, on the other hand, completely excludes meta-information and solely relies on the relational structure of the data. The main motivation of this work is to bridge the relational data and the metadata to improve detection performance. In particular, we aim to leverage the metadata to estimate initial class probabilities for users, products, and reviews, which we incorporate as prior potentials of the nodes under a new MRF model.

Our motivation requires two main changes to be made in the FraudEagle framework. First, we represent the reviews as nodes inside the network. As such, we model the relational data as a user–review–product network, where each review node is connected to its corresponding user and product nodes (see Figure 4.1). The reason is that we can use metadata to estimate a "suspicion score" not only for users and products, but also for reviews. Representing reviews explicitly as nodes enables us to readily integrate this information to the formulation as prior potentials of reviews.

The second change is related to the semantics of class labels for products. The domain of class labels for products in FraudEagle is $\mathcal{L}_P = \{$*good-quality*, *bad-*

[1]Sensitivity analysis in [103] found that $\epsilon \in [0.01, 0.15]$ yields desirable and comparable results. In this work we use $\epsilon = 0.1$.

quality}. However, the metadata does not lend itself to estimating meaningful priors for these classes. Two possible ways of inferring product quality, average rating and sentiment analysis of reviews, could be misleading—both average ratings and review sentiment of products associated with fake reviewing are tempered with, and thus are not reliable. We could, on the other hand, use the metadata to perform behavioral analysis on products to characterize the likelihood that they are under manipulation (see §4.2.2). In other words, the "suspicion score" of a product estimated from metadata would not translate to its quality but to its likelihood of being a target of opinion spam. Therefore, in our formulation the products are labeled as $\mathcal{L}_P = \{non\text{-}target, target\}$. As the semantics of the network representation has changed, we also discard the sentiment, i.e. the signs on the edges and use an unsigned network $G = (V, E)$. This is because under the new setting, a targeted product can be associated with negative as well as positive fake reviews, when manipulated with an intent to defame or to hype, respectively.

The joint probability of our formulation can be written similar to Eqn. (4.1), where the node set $V = U \cup P \cup R$ now consists of three types of nodes, including the Q review nodes $R = \{r_1, \ldots, r_Q\}$, with labels from domain $\mathcal{L}_R = \{genuine, fake\}$. Moreover, the compatibility potentials are typed as ψ_{ij}^t reflecting the two types of relations in the network; the user-review edges $(u_i, r_k, t = \text{`}write\text{'}) \in E$ and the review-product edges $(r_k, p_j, t = \text{`}belong\text{'}) \in E$. In terms of setting the model parameters, we estimate the prior potentials ϕ_i from metadata for all three types of nodes, $\forall i \in V$, which we describe in §4.2.2. On the other hand, we initialize the compatibility potentials ψ_{ij}^t so as to enforce homophily [115]. In particular, we assume that all the reviews written by spammers (benign users) are fake (genuine), and that with high probability fake (genuine) reviews belong to targeted (non-targeted) products; although with some probability fake reviews may also belong to non-targeted products as part of camouflage, and similarly genuine reviews may co-exist along with fake reviews for targeted products. Nevertheless, we assume that the majority of the reviews for targeted (non-targeted) products are fake (genuine), which possibly needs to hold true for a spam campaign to be able to manipulate the average rating of a targeted product successfully. Overall, SpEagle uses the following settings.

Table 4.1: Compatibility potentials ψ^t used by SpEagle.

	User ($\psi^{t=\text{`}write\text{'}}$)		($\psi^{t=\text{`}belong\text{'}}$) **Product**	
Review	*benign*	*spammer*	*non-target*	*target*
genuine	1	0	$1-\epsilon$	ϵ
fake	0	1	ϵ	$1-\epsilon$

Next, we describe how we estimate the prior potentials from metadata for all the user, review, and product nodes. Then, we introduce the *semi-supervised* version of SpEagle and show how labels can be used if available. We provide an outline of

our algorithm and present the inference steps for computing the class assignments. Finally, we introduce a *light* version of SpEagle for computational speed-up.

Table 4.2: Features for users and products derived from metadata; categorized as behavior and text-based. H/L depicts if a High/Low value of the feature is more likely to be associated with spam.

User & Product Features			
Behavior	MNR	H	Max. number of reviews written in a day [4, 109]
	PR	H	Ratio of positive reviews (4-5 star) [109]
	NR	H	Ratio of negative reviews (1-2 star) [109]
	avgRD	H	Avg. rating deviation $avg(\|d_{i*}\|)$ of user (product) i's reviews [101, 104, 109], where $\|d_{ij}\|$ is absolute rating deviation of i's rating from j's average rating: $avg_{e_{ij} \in E_{i*}} \|d_{ij}\|$, for $d_{ij} = r_{ij} - avg_{e \in E_{*j}} r(e)$
	WRD	H	Weighted rating deviation [101], where reviews are weighed by recency: $\frac{\sum_{e_{ij} \in E_{i*}} \|d_{ij}\| w_{ij}}{\sum_{e_{ij} \in E_{i*}} w_{ij}}$, for $w_{ij} = \frac{1}{(t_{ij})^{\alpha}}$ (t_{ij} is rank order of review e_{ij} among reviews of j, $\alpha = 1.5$ is decay rate)
	BST	H	Burstiness [104, 109]—spammers are often short-term members of the site. $x_{BST}(i) = \begin{cases} 0, & \text{if } L(i) - F(i) > \tau \\ 1 - \frac{L(i) - F(i)}{\tau}, & \text{otherwise} \end{cases}$ where $L(i) - F(i)$ is number of days between last and first review of i, $\tau = 28$ days.
	ERD	L	Entropy of rating distribution of user's (product's) reviews [new]
	ETG	L	Entropy of temporal gaps Δ_t's. Given the temporal line-up of a user's (product's) reviews, each Δ_t denotes the temporal gap in days between consecutive pairs [new]
Text	RL	L	Avg. review length in number of words [109]
	ACS	H	Avg. content similarity—pairwise cosine similarity among user's (product's) reviews, where a review is represented as a bag-of-bigrams [101, 104]
	MCS	H	Max. content similarity—maximum cosine similarity among all review pairs [4, 109]

From metadata to features to priors

To estimate the prior potentials, we first extract indicative features of spam from available metadata (ratings, timestamps, review text) and then convert them to prior class probabilities. The priors are estimated for all three types of nodes. As

Table 4.3: Features for reviews derived from metadata; categorized as behavior and text-based. H/L depicts if a High/Low value of the feature is more likely to be associated with spam.

	Review Features		
Behavior	Rank	L	Rank order among all the reviews of product [3]
	RD	H	Absolute rating deviation from product's average rating [100]
	EXT	H	Extremity of rating [4]: $x_{EXT} = 1$ for ratings $\{1, 5\}$, 0 otherwise (for $\{2, 3, 4\}$)
	DEV	H	Thresholded rating deviation of review e_{ij} [4]: $$x_{DEV}(i) = \begin{cases} 1, & \textit{if } \frac{\lvert r_{ij} - avg_{e \in E_{*j}} r(e) \rvert}{4} > \beta_1 \\ 0, & \textit{otherwise} \end{cases}$$ where β_1 is learned by recursive minimal entropy partitioning
	ETF	H	Early time frame [4]—spammers often review early to increase impact. $x_{ETF}(f(e_{ij})) = 1$ if $f(e_{ij}) > \beta_2$, and 0 otherwise, where, $$f(e_{ij}) = \begin{cases} 0, & \textit{if } T(i,j) - F(j) > \delta \\ \frac{T(i,j) - F(j)}{\delta}, & \textit{otherwise} \end{cases}$$ where $T(i, j) - F(j)$ is the difference between the time of review e_{ij} and first review j, for $\delta = 7$ months, and β_2 is estimated by recursive minimal entropy partitioning
	ISR	H	Is singleton? If review is user's sole review, then $x_{ISR} = 1$, otherwise 0 [new]
Text	PCW	H	Percentage of ALL-capitals words [3, 100]
	PC	H	Percentage of capital letters [100]
	L	L	Review length in words [100]
	PP1	L	Ratio of 1st person pronouns ('I', 'my', etc.) [100]
	RES	H	Ratio of exclamation sentences containing '!' [100]
	SW	H	Ratio of subjective words (by sentiWordNet) [100]
	OW	L	Ratio of objective words (by sentiWordNet) [100]
	F	H	Frequency of review (approximated using locality sensitive hashing) [new]
	DL_u	L	Description length (information-theoretic) based on unigrams (i.e., words) [new]
	DL_b	L	Description length based on bigrams [new]

such, we compute features for users, products, and reviews separately. Most of our features have been used several times in previous work on opinion spam detection

including [3, 4, 100, 101, 104, 109], while several are introduced in this work. Table 4.2 includes brief descriptions for the features. Most of them are self-explanatory, and hence we omit detailed explanation for brevity. Instead, we provide references to prior work where they have also been used.

In particular, we show the user and product features in Table 4.2. All but one of these features can be defined for both, where we either consider all reviews of a user or all reviews of a product. One feature (BST) applies only to users, which captures the burstiness related to the age of a user, defined as the number of days between their first and the last review. Intuitively, spammers are short-term members of a review site rather than veterans, which is characterized by BST. We also show the review features in Table 4.3. Note that all the features are categorized into two—behavioral versus text-based. Text-based ones are solely derived from review content. Behavioral features are based on time stamps, ratings, distributions, ranks, etc. In the experiments, we evaluate the effectiveness of individual features as well as the feature categories.

Given a set of values $\{x_{1i}, \ldots, x_{Fi}\}$ for the F features of a node i, the next step is to combine them into a spam score $S_i \in [0, 1]$, such that the class priors can be initialized as $\{1 - S_i, S_i\}$. The features, however, may have different scales and varying distributions. To unify them into a comparable scale and interpretation, we leverage the cumulative distribution function (CDF). In particular, when we design the features, we have an understanding of whether a *high* (H) or a *low* (L) value is more suspicious for each feature. For example, high average rating deviation (avgRD) and low entropy of rating distribution (ERD) are suspicious. To quantify the extremity of a feature value x, we then use the empirical CDF to estimate the probability that the data contains a value as low or as high as x. More formally, for each feature l, $1 \leq l \leq F$, and its corresponding value x_{li}, we compute

$$f(x_{li}) = \begin{cases} 1 - P(X_l \leq x_{li}), & \textit{if high is suspicious (H)} \\ P(X_l \leq x_{li}), & \textit{otherwise (L)} \end{cases}$$

where X_l denotes a real-valued random variable associated with feature l with probability distribution P. To compute $f(\cdot)$, we use the *empirical* probability distribution of each feature over all the nodes of the given type. Overall, the features with suspiciously low or high values all receive low f values. Finally we combine these f values to compute the spam score of a node i as follows.

$$S_i = 1 - \sqrt{\frac{\sum_{l=1}^{F} f(x_{li})^2}{F}} \tag{4.2}$$

Figure 4.2 shows the CDF of example features for filtered (considered as spam)

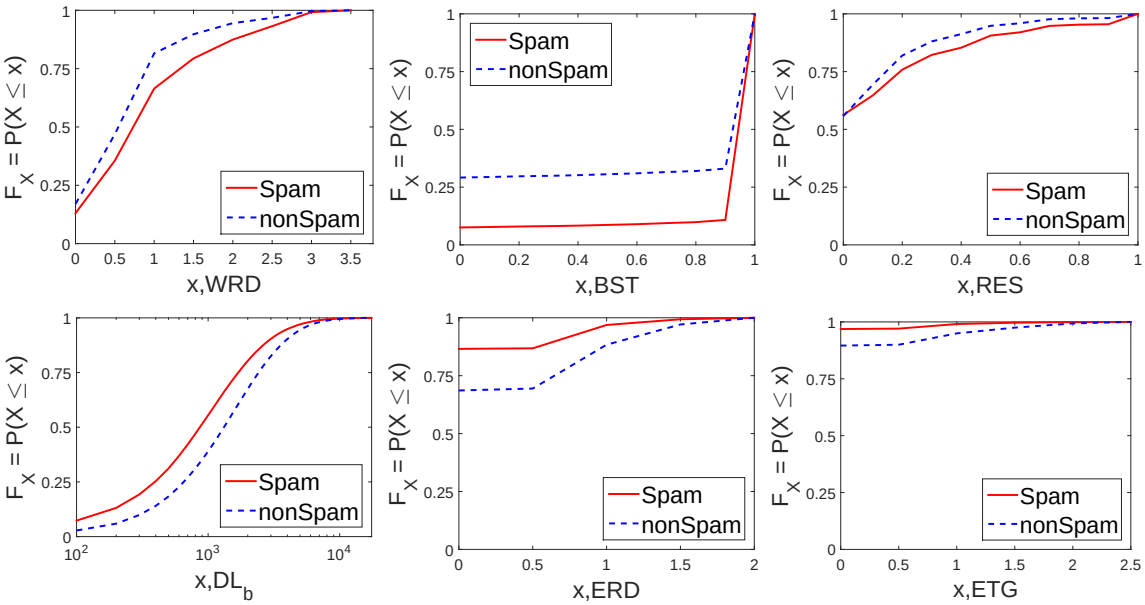


Figure 4.2: CDF distribution of example features for spam vs. non-spam review(er)s. Spam review(er)s often have (H)igher values for features in top row, and (L)ower values for those in bottom row.

versus recommended (considered as non-spam) reviews and associated reviewers in one of our Yelp datasets. Notice that spam review(er)s obtain higher values for certain features (e.g., those in top row) and lower values for some others (bottom row)—hence the (H) and (L) distinction in the $f(\cdot)$ function above. Also notice that the individual features provide only weak evidence on their own, as the CDF curves of the classes are somewhat close to each other. We aim to obtain a stronger signal by combining the multiple evidence from all the features using Eqn. (4.2).

Semi-supervised SpEagle

One of the key advantages of our formulation is that it enables seamless integration of labeled data when available. We describe two possible ways to incorporate label information. The first scenario does not involve any learning on the labeled data. Specifically, given the labels for a set of nodes (reviews, users, and/or products), we simply initiate the priors as $\{\epsilon, 1-\epsilon\}$ for those that are associated with spam (i.e., $fake$, $spammer$, or $target$), and $\{1-\epsilon, \epsilon\}$ otherwise. The priors of unlabeled nodes are estimated from metadata as given in Eqn. (4.2). The inference procedure remains the same. As this integration does not require model training or any other changes, it is extremely efficient. It is particularly suitable when the size of the labeled data is too small or unbalanced to learn from.

An alternative approach is to use the labeled data to train classifier models (one for each type of node), and initialize the prior potentials for unlabeled nodes with the class probabilities provided by the classifiers. This approach is particularly possible in our setting, where we extract features indicative of spam for each node in our network. When no labels are available, we transform these features into priors in an unsupervised fashion. In the semi-supervised scenario, one can instead use the features and the supplied labels to train models to obtain "supervised priors". Note that the class distribution of labeled data would likely be unbalanced. Most learning methods are sensitive to skewed label distribution. Therefore, one needs to either use cost-sensitive methods [116, 117] or deploy under/over-sampling techniques to balance the training data before learning [118]. Here again, the modifications are only in initializing and estimating the prior potentials of labeled and unlabeled nodes, respectively, and the inference procedure remains intact. We discuss the details of our proposed algorithm next.

The algorithm

We provide the steps of our SpEagle in Algorithm 6. As input, we take the user–review–product graph G, compatibility potential parameters ψ^t as given in Table 4.1, available metadata for feature extraction, and a set of node labels L (if any). Note that L can contain labels for a mixture of user, review, and product nodes, or can be empty. We output the corresponding class probabilities for all the unlabeled nodes.

First, we compute or initialize the prior class probabilities for all the nodes (Lines 3-10). Specifically, the priors are set to $\phi \leftarrow \{\epsilon, 1-\epsilon\}$ for those labeled nodes that belong to the spam category (i.e., $fake$, $spammer$, or $target$), and to $\phi \leftarrow \{1-\epsilon, \epsilon\}$ for nodes labeled as non-spam.[2] For unlabeled nodes, we compute their corresponding features in Table 4.2 (depending on their type), combine them into a spam score S using Eqn. (4.2), and set the priors as $\phi \leftarrow \{1-S, S\}$.

In the remainder, we follow the main steps of the Loopy Belief Propagation (LBP) algorithm [111]. LBP is based on iterative message passing between the connected nodes. We first initialize all the messages to 1 (Lines 11-13). At every iteration, a *message* $m_{i \to j}$ is then sent from each node i to each neighboring node j, where $T_i, T_j \in \{U, R, P\}$ denote the type of node i and j, respectively. The message represents the belief of i about j, i.e., what i "thinks" j's label distribution is. More formally, $m_{i \to j}$ captures the probability distribution over the class labels of j, and

[2]Our experiments showed that the alternative approach discussed in §4.2.2 produces similar results. We list the first approach in Algorithm 6 due to its efficiency, as it requires no training.

Algorithm 6: SpEagle

Input: User–Review–Product graph $G = (V, E)$, compatibility potentials ψ^t (Table 4.1), review metada (ratings, timestamps, text), labeled node set L
Output: Class probabilities for each node $i \in V \backslash L$
foreach $i \in V$ **do** // *compute/initialize priors*
    if $i \in L$ **then**
        if *i is positive (spam) class* **then** $\phi_i \leftarrow \{\epsilon, 1-\epsilon\}$
        else $\phi_i \leftarrow \{1-\epsilon, \epsilon\}$
    else
        Extract corresponding features in Table 4.2
        Compute spam score S_i using Eqn. (4.2)
        $\phi_i \leftarrow \{1-S_i, S_i\}$
foreach $(Y_i^{T_i}, Y_j^{T_j}, t) \in E$ **do** // *initialize all msg.s*
    foreach $y_j \in \mathcal{L}_{T_j}$ **do**
        $m_{i \to j}(y_j) \leftarrow 1$
repeat// *iterative message passing*
    foreach $(Y_i^{T_i}, Y_j^{T_j}, t) \in E$ **do**
        foreach $y_j \in \mathcal{L}_{T_j}$ **do**
17 $$m_{i \to j}(y_j) = \alpha \sum_{y_i \in \mathcal{L}_{T_i}} \Big(\phi_i(y_i)\ \psi_{ij}^t(y_i, y_j) \prod_{Y_k \in \mathcal{Y}_{\mathcal{N}_i} \backslash Y_j} m_{k \to i}(y_i) \Big)$$
until messages stop changing within a δ threshold
foreach $Y_i^{T_i} \in \mathcal{Y}_{V \backslash L}$ **do** // *compute final beliefs*
    foreach $y_i \in \mathcal{L}_{T_i}$ **do**
21 $$b_i(y_i) = \beta\ \phi_i(y_i) \prod_{Y_j \in \mathcal{Y}_{\mathcal{N}_i}} m_{j \to i}(y_i)$$

is computed based on the class priors of i, the compatibility potentials of the *type* of edge that connects i and j, and the messages that i receive from its neighbors excluding j. The exact expression is given in Line 17, where $\mathcal{N}_i$ denotes the set of i's neighbors and α is a normalization constant to ensure that class probabilities sum to 1. These messages are exchanged iteratively over the edges until a "consensus" is reached, i.e., the messages stop changing between consecutive iterations within a

small threshold such as $\delta = 10^{-3}$ (Lines 14-18).

When the messages stabilize, we compute the marginal probability, called the *belief* $b_i(y_i)$, of assigning each Y_i associated with a node of type $T_i \in \{U, R, P\}$ with the label y_i in label domain $\mathcal{L}_{T_i}$ (Lines 19-21). The exact expression is given in Line 21, where β is the normalization constant so that the marginal probabilities sum to 1. For classification, one can assign labels based on $\max_{y_i} b_i(y_i)$. For ranking, we sort by the probability values $b_i(y_i)$, where $y_i = spammer$ and $y_i = fake$ respectively for users and reviews.

Light-weight SpEagle

The original SpEagle computes all the features for every (unlabeled) node of each type in the graph (Line 8, Alg. 6). In the experiments we investigate the effectiveness of the features and identify a small subset of review features that produces comparable performance to using all of them. As such, we propose a light-weight version of our method, called SpLite (for SpEagle-Light), where we initialize the priors for unlabeled reviews based on the spam score computed only on those features, and use unbiased priors $\{0.5, 0.5\}$ for (unlabeled) users and products. This significantly reduces the feature extraction overhead for reviews, and completely avoids it for users and products, enabling speed-up with only slight compromise in performance. We compare the performance and running time of SpLite and SpEagle in §4.3.4.

4.3 Evaluation

We evaluate our approach *quantitative*ly on real-world datasets with near-ground-truth. In the following we first describe our datasets, evaluation metrics, and compared methods and then present the performance results.

Data description In this study, we use three datasets collected from Yelp.com, summary statistics of which are given in Table 4.4. The first dataset, named `YelpChi`, has been collected and used by [109] and contains reviews for a set of restaurants and hotels in the Chicago area. We collected two more datasets from Yelp for this work, named as `YelpNYC` and `YelpZip`. `YelpNYC` contains reviews for restaurants located in NYC. `YelpZip` is even larger, where we start with a zipcode in NY state, collect reviews for restaurants in that zipcode[3], increase the zipcode

[3]Yelp allows to search for restaurants by zipcode, e.g., http://www.yelp.com/search?cflt=restaurants&find_loc=11794

number incrementally, and repeat. The zipcodes are organized by geography, as such this process gives us reviews for restaurants in a continuous region of the U.S. map, including NJ, VT, CT, and PA.

Table 4.4: Review datasets used in this work.

Dataset	**#Reviews** (filtered %)	**#Users** (spammer %)	**#Products** (rest.&hotel)
YelpChi	67,395 (13.23%)	38,063 (20.33%)	201
YelpNYC	359,052 (10.27%)	160,225 (17.79%)	923
YelpZip	608,598 (13.22%)	260,277 (23.91%)	5,044

Yelp has a filtering algorithm in place that identifies fake/suspicious reviews and separates them into a filtered list. The filtered reviews are also made public; the Yelp page of a business shows the *recommended* reviews, while it is also possible to view the filtered/unrecommended reviews through a link at the bottom of the page. While the Yelp anti-fraud filter is not perfect (hence the "near" ground truth), it has been found to produce accurate results [119]. Our Yelp datasets contain both recommended and filtered reviews. We consider them as *genuine* and $fake$, respectively. We also separate the users into two classes; *spammers*: authors of $fake$ (filtered) reviews, and *benign*: authors with no filtered reviews. We evaluate SpEagle on both tasks: spammer detection and fake review detection.

Evaluation metrics We use four well-known ranking based metrics for our performance evaluation. We obtain the (i) precision-recall (PR) as well as (ii) ROC (true positive rate vs. FPR) curves, where the points on a curve are obtained by varying the classification threshold. We compute the area under the curve, respectively denoted as AP (average precision) and AUC. For problems such as outlier/spam/fraud detection, often the quality at the top of the ranking results are the most important. Therefore, we also inspect (iii) $precision@k$ and (iv) $NDCG@k$, for $k = \{100, 200, \ldots, 1000\}$. $Precision@k$ captures the ratio of spam(mers) in top k positions. $NDCG@k$ provides a weighted score, which favors rankings where spam(mers) are ranked closer to the top. In particular, $NDCG@k = \frac{DCG@k}{IDCG@k}$ for $DCG@k = \sum_{i=1}^{k} \frac{2^{l_i}-1}{\log_2(i+1)}$, where l_i captures the true label of item at rank i (1: spam, 0: non-spam) and $IDCG@k$ is the DCG for ideal ranking where all l_i's are 1.

Compared Methods We compare the performance of SpEagle to FraudEagle [103] as well as to the graph-based approach by [106] denoted as Wang et al., thanks to their publicly available implementations. We also consider Prior, where

we use the spam scores (for users and reviews) computed solely based on metadata as discussed in §4.2.2. PRIOR does not use the network information, and hence corresponds to SPEAGLE without LBP. We also compare to the semi-supervised SPEAGLE$^+$ with varying amount of labeled data, as well as to the computationally light-weight version SPLITE.

Table 4.5: AP and AUC performance of compared methods on all three datasets.

	User Ranking					
	AP			**AUC**		
	Y'Chi	Y'NYC	Y'Zip	Y'Chi	Y'NYC	Y'Zip
RANDOM	0.2024	0.1782	0.2392	0.5000	0.5000	0.5000
FRAUDEAGLE	0.2537	0.2233	0.3091	0.6124	0.6062	0.6175
WANG ET AL.	0.2659	0.2381	0.3306	0.6167	0.6207	0.6554
PRIOR	0.2157	0.1826	0.2550	0.5294	0.5081	0.5269
SPEAGLE	**0.3393**	**0.2680**	**0.3616**	**0.6905**	**0.6575**	**0.6710**
SPEAGLE$^+$(1%)	0.3967	0.3480	0.4245	0.7078	0.6828	0.6907
SPLITE$^+$ (1%)	0.3777	0.3331	0.4218	0.6744	0.6542	0.6784
	Review Ranking					
	AP			**AUC**		
	Y'Chi	Y'NYC	Y'Zip	Y'Chi	Y'NYC	Y'Zip
RANDOM	0.1327	0.1028	0.1321	0.5000	0.5000	0.5000
FRAUDEAGLE	0.1067	0.1122	0.1524	0.3735	0.5063	0.5326
WANG ET AL.	0.1518	0.1255	0.1803	0.5062	0.5415	0.5982
PRIOR	0.2241	0.1789	0.2352	0.6707	0.6705	0.6838
SPEAGLE	**0.3236**	**0.2460**	**0.3319**	**0.7887**	**0.7695**	**0.7942**
SPEAGLE$^+$(1%)	0.3352	0.2757	0.3545	0.7951	0.7829	0.8040
SPLITE$^+$ (1%)	0.3124	0.2550	0.3448	0.7693	0.7631	0.7923

4.3.1 Detection results

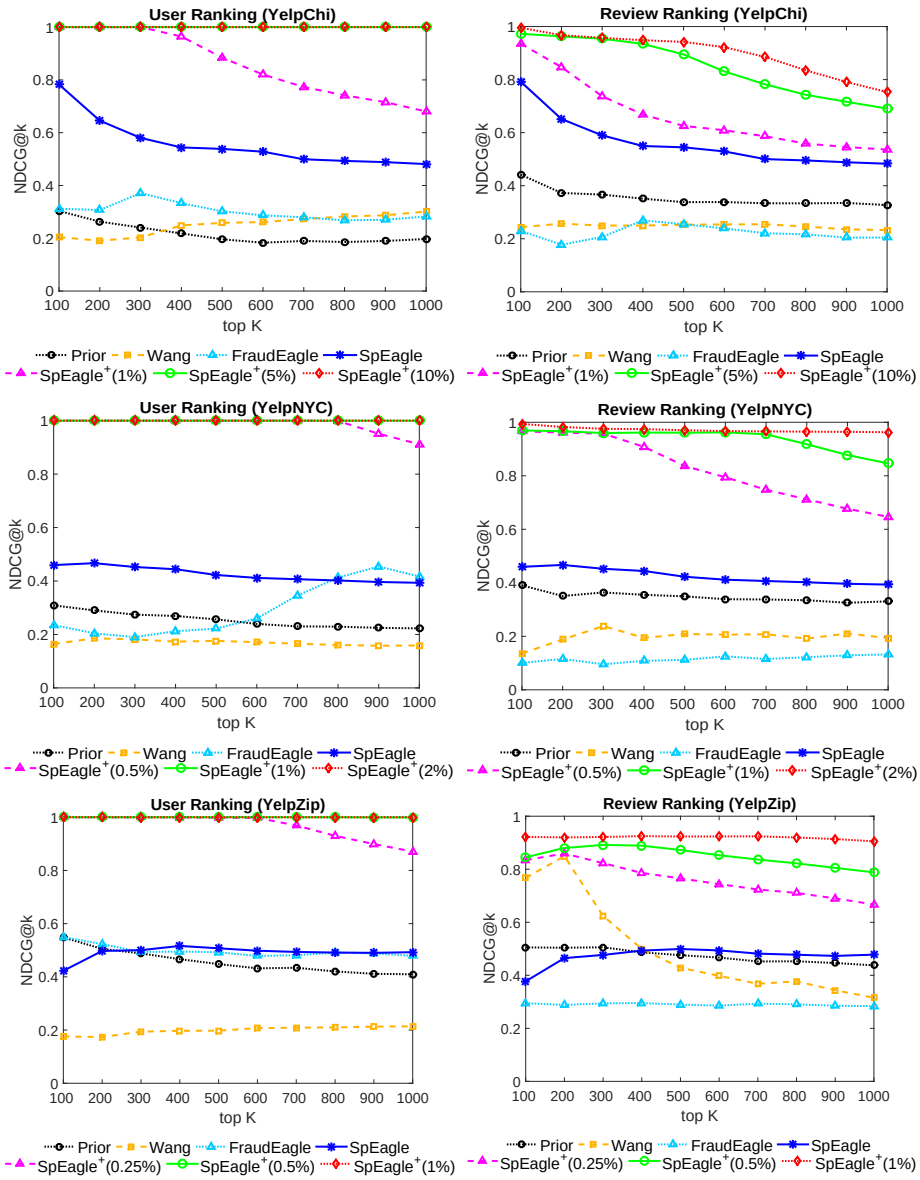
We start by comparing the detection performance of the methods. Table 4.5 provides the AP and AUC over all three datasets for both user and review ranking. Notice that SPEAGLE outperforms all others, WANG ET AL. and PRIOR being the second best method for user and review ranking, respectively. The superiority of SPEAGLE's ranking becomes more evident when the top of the ranking results are considered through *precision*@k in Table 4.6, as well as the $NDCG@k$ in Figure 4.3.

Table 4.6: *Precision@k* of compared methods on (from top to bottom) `YelpChi`, `YelpNYC`, and `YelpZip`.

	User Ranking				Review Ranking			
k	Prior	FraudEagle	Wang et al.	SpEagle	Prior	FraudEagle	Wang et al.	SpEagle
100	0.32	0.30	0.21	**0.73**	0.38	0.25	0.24	**0.74**
200	0.26	0.30	0.19	**0.59**	0.33	0.18	0.26	**0.59**
300	0.23	0.38	0.21	**0.52**	0.33	0.21	0.25	**0.53**
400	0.21	0.33	0.26	**0.49**	0.32	0.29	0.25	**0.50**
500	0.18	0.29	0.27	**0.50**	0.31	0.27	0.25	**0.50**
600	0.17	0.28	0.27	**0.49**	0.32	0.25	0.26	**0.49**
700	0.18	0.27	0.29	**0.46**	0.31	0.22	0.26	**0.46**
800	0.18	0.26	0.30	**0.46**	0.32	0.22	0.25	**0.46**
900	0.18	0.26	0.30	**0.46**	0.32	0.20	0.23	**0.45**
1000	0.19	0.28	0.32	**0.45**	0.31	0.20	0.23	**0.45**
100	0.34	0.21	0.15	**0.44**	0.34	0.10	0.17	**0.44**
200	0.30	0.19	0.19	**0.46**	0.32	0.12	0.22	**0.46**
300	0.28	0.17	0.18	**0.44**	0.34	0.09	0.27	**0.44**
400	0.27	0.21	0.17	**0.44**	0.34	0.11	0.21	**0.44**
500	0.25	0.22	0.17	**0.41**	0.33	0.11	0.22	**0.41**
600	0.23	0.27	0.17	**0.40**	0.32	0.13	0.22	**0.40**
700	0.22	0.37	0.16	**0.39**	0.32	0.12	0.22	**0.39**
800	0.22	**0.45**	0.16	0.39	0.32	0.13	0.20	**0.39**
900	0.22	**0.50**	0.15	0.38	0.31	0.13	0.22	**0.38**
1000	0.22	**0.45**	0.16	0.38	0.32	0.14	0.20	**0.38**
100	0.51	**0.55**	0.18	0.44	0.51	0.29	**0.86**	0.43
200	0.48	0.52	0.18	**0.53**	0.51	0.29	**0.92**	0.52
300	0.46	0.48	0.20	**0.52**	0.51	0.29	**0.61**	0.51
400	0.44	0.49	0.20	**0.54**	0.48	0.30	0.46	**0.53**
500	0.42	0.48	0.20	**0.52**	0.47	0.29	0.38	**0.53**
600	0.41	0.47	0.21	**0.51**	0.46	0.28	0.35	**0.52**
700	0.41	0.47	0.21	**0.50**	0.44	0.29	0.32	**0.50**
800	0.40	0.49	0.22	**0.50**	0.45	0.29	0.34	**0.49**
900	0.39	0.48	0.22	**0.49**	0.44	0.28	0.30	**0.48**
1000	0.39	0.47	0.22	**0.50**	0.43	0.28	0.27	**0.49**

4.3.2 Semi-supervised detection

Next we investigate how much the detection performance can be improved by semi-supervision; i.e., providing SpEagle with a subset of the *review* labels. We analyze performance for varying amount of labeled data. Our datasets contain different num-



(a) User Ranking (b) Review Ranking

Figure 4.3: $NDCG@k$ of compared methods on (from top to bottom) `YelpChi`, `YelpNYC`, and `YelpZip` for both user and review ranking. Also shown are results for SpEagle$^+$ with varying % of labeled data.

ber of reviews (see Table 4.4), where `YelpChi` is the smallest dataset and `YelpZip` is the largest. To keep the number of provided labels at realistic size, we label smaller percentages of the data for larger datasets; in particular $\{1\%, 5\%, 10\%\}$ for `YelpChi`, $\{0.5\%, 1\%, 2\%\}$ for `YelpNYC`, and $\{0.25\%, 0.5\%, 1\%\}$ for `YelpZip`.

Figure 4.3 shows the $NDCG@k$ performance of semi-supervised SpEagle,

Table 4.7: *Precision@k* of SpEagle$^+$ for review ranking on all three datasets with varying % of labeled data.

	YelpChi				YelpNYC			
k	**0%**	**1%**	**5%**	**10%**	**0%**	**0.5%**	**1%**	**2%**
100	0.7400	0.9300	0.9650	0.9950	0.4400	0.9650	0.9630	0.9930
200	0.5900	0.8195	0.9565	0.9600	0.4600	0.9595	0.9625	0.9790
300	0.5333	0.6910	0.9477	0.9500	0.4433	0.9557	0.9553	0.9713
400	0.4975	0.6162	0.9245	0.9408	0.4350	0.8935	0.9587	0.9710
500	0.5020	0.5736	0.8772	0.9344	0.4100	0.8076	0.9586	0.9664
600	0.4900	0.5617	0.8008	0.9110	0.3983	0.7602	0.9603	0.9633
700	0.4600	0.5407	0.7451	0.8671	0.3943	0.7079	0.9521	0.9623
800	0.4587	0.5125	0.7015	0.8078	0.3900	0.6685	0.9067	0.9616
900	0.4544	0.5018	0.6739	0.7570	0.3844	0.6307	0.8586	0.9610
1000	0.4510	0.4944	0.6471	0.7141	0.3820	0.5982	0.8225	0.9597
	YelpZip							
k	**0%**	**0.25%**	**0.5%**	**1%**				
100	0.4300	0.8740	0.8540	0.9090				
200	0.5150	0.8850	0.8935	0.9130				
300	0.5133	0.8303	0.9037	0.9173				
400	0.5250	0.7823	0.8972	0.9225				
500	0.5260	0.7574	0.8750	0.9212				
600	0.5150	0.7320	0.8500	0.9218				
700	0.4971	0.7090	0.8307	0.9226				
800	0.4900	0.6946	0.8138	0.9178				
900	0.4833	0.6711	0.7938	0.9106				
1000	0.4880	0.6453	0.7744	0.9004				

denoted as SpEagle$^+$, on all three datasets for both user and review ranking, where varying amount (%) of the review labels are provided (values are averaged over 10 independent runs). Table 4.7 shows the corresponding *precision@k* values for review ranking (user ranking performance is similar, omitted for brevity). We notice that the performance is improved considerably even with very small amount of supervision, where the semi-supervised results are significantly better than all the competing methods. For example, labeling only 1% of the reviews (around 670 labels for `YelpChi`, 3600 for `YelpNYC`, and 6000 for `YelpZip`), *precision@k* for review ranking is increased by 4-22% on `YelpChi`, 44-56% on `YelpNYC`, and 39-47% on `YelpZip` in absolute terms, for $k \in [100, 1000]$.[4] Table 4.5 also includes the AP and AUC performance of SpEagle$^+$ across all datasets for 1% labeled data.

[4]The corresponding absolute improvements for user ranking are 18-47% on `YelpChi`, 54-61% on `YelpNYC`, and 46-56% on `YelpZip`.

4.3.3 Analyzing priors

Next we investigate the informativeness of feature categories and individual features in estimating effective priors.

User vs. Review vs. Product priors We start by analyzing the user, review, and product priors. To study the effectiveness of a certain group of priors (e.g., user, or user+review), we only initialize the priors for the nodes in that group in the graph (as estimated from metadata) and set the remaining node priors to unbiased, i.e. $\{0.5, 0.5\}$. We then compare the performance of SpEagle with priors of various groups.

Table 4.8 shows the AP and AUC performance of SpEagle across datasets with various prior groups. We find that the review priors produce the most effective results, followed by user priors, and product priors. The difference in performance is especially pronounced on our largest dataset `YelpZip`. To our surprise, we find that the product priors alone yield performance that is lower than that by random ranking. As a result, SpEagle with only user and review priors performs almost as well as using all the priors.

Table 4.8: AP and AUC performance of SpEagle when priors are initialized (estimated from metadata) for various node types; (U)sers, (R)eviews, (P)roducts (rest set to unbiased). P-priors yield the lowest performance, while R-priors are the most effective.

	User Ranking					
	AP			**AUC**		
	Y'Chi	Y'NYC	Y'Zip	Y'Chi	Y'NYC	Y'Zip
Random	0.2024	0.1782	0.2392	0.5000	0.5000	0.5000
SpEagle (U)	0.3197	0.2624	0.2808	0.6767	0.6483	0.6183
SpEagle (P)	0.1550	0.1357	0.1814	0.3905	0.3930	0.3801
SpEagle (R)	0.3226	0.2575	0.3449	0.6771	0.6477	0.6562
SpEagle (UR)	**0.3398**	**0.2680**	0.3615	**0.6905**	**0.6575**	0.6709
SpEagle (URP)	0.3393	**0.2680**	**0.3616**	**0.6905**	**0.6575**	**0.6710**
	Review Ranking					
	AP			**AUC**		
	Y'Chi	Y'NYC	Y'Zip	Y'Chi	Y'NYC	Y'Zip
Random	0.1327	0.1028	0.1321	0.5000	0.5000	0.5000
SpEagle (U)	0.3043	0.2400	0.1427	0.7783	0.7629	0.5940
SpEagle (P)	0.0755	0.0640	0.0806	0.1643	0.2536	0.2277
SpEagle (R)	0.3098	0.2378	0.3180	0.7820	0.7656	0.7884
SpEagle (UR)	**0.3241**	**0.2460**	**0.3320**	**0.7887**	**0.7695**	**0.7942**
SpEagle (URP)	0.3236	**0.2460**	0.3319	**0.7887**	**0.7695**	**0.7942**

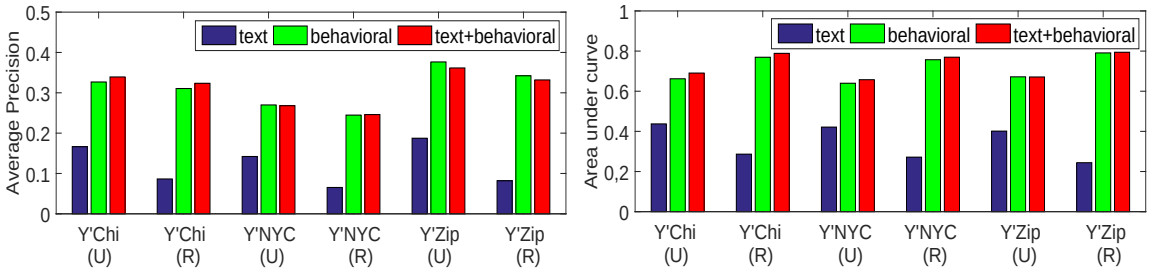


Figure 4.4: (left) AP and (right) AUC performance of SpEagle when various feature types are used to estimate priors; text, behavior, all (see Table 4.2) on all datasets for both (U)ser and (R)eview ranking.

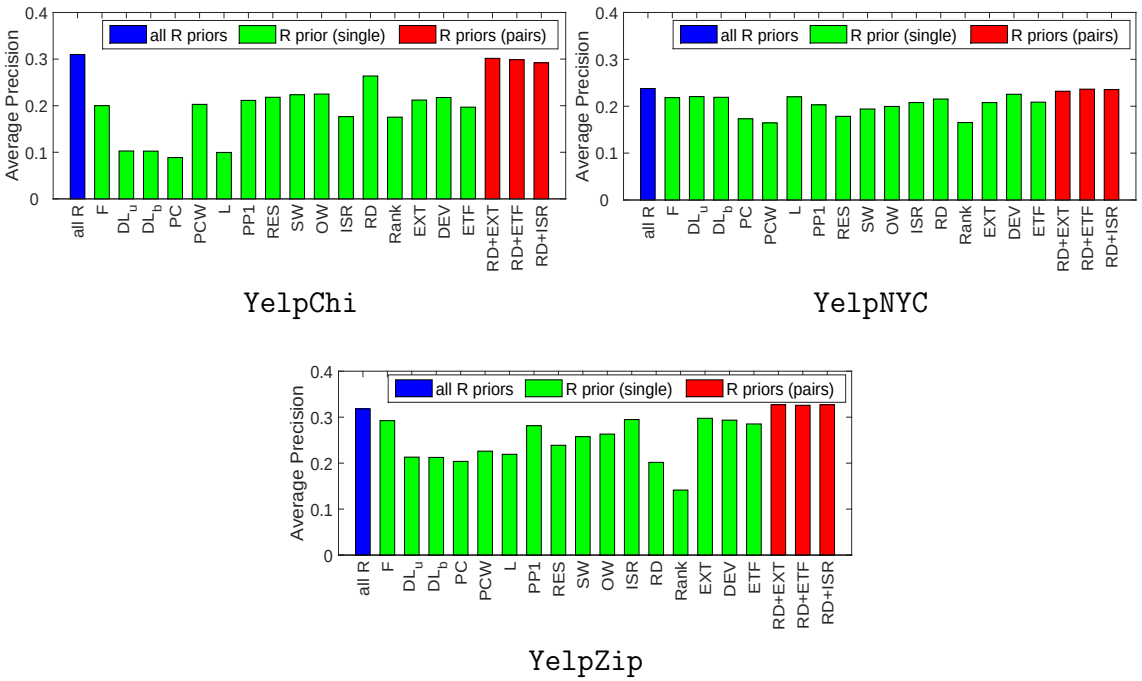


Figure 4.5: AP performance of SpEagle when all (green), individual (blue), and behavioral pairs (red, only 3 shown) of (R)eview features are used to estimate review priors (rest set to unbiased), on (clockwise) `YelpChi`, `YelpNYC`, and `YelpZip`.

Text- vs. Behavior-based priors Recall from Table 4.2 that our features are derived from review text as well as behavioral clues. Here we investigate the performance of SpEagle when priors are estimated from only text-based versus only behavioral features (for all user, review, and product nodes) as compared to using all the possible features. Figure 4.4 shows the AP and AUC performance across all datasets for both user and review ranking.

We observe that using text-based features alone yields inferior performance

compared to behavioral features. Moreover, behavioral features alone produce comparable results to using all the features, where the differences across datasets and (U)ser vs. (R)eview ranking tasks are insignificant. These findings are in agreement with those in [109], which found that their behavioral features performed very well, whereas the linguistic features were not as effective.

Table 4.9: *NDCG@k* performance comparison of SpEagle vs. SpLite (with 1% supervision on all datasets).

	User Ranking					
	YelpChi		YelpNYC		YelpZip	
k	Sp'le	SpLite	Sp'le	SpLite	Sp'le	SpLite
100	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
200	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
300	1.0000	0.9995	1.0000	1.0000	0.9997	1.0000
400	0.9645	0.9589	1.0000	1.0000	0.9998	1.0000
500	0.8841	0.8677	1.0000	1.0000	0.9998	1.0000
600	0.8205	0.8107	1.0000	1.0000	0.9998	1.0000
700	0.7731	0.7650	1.0000	1.0000	0.9999	1.0000
800	0.7416	0.7279	1.0000	1.0000	0.9999	1.0000
900	0.7157	0.6980	1.0000	1.0000	0.9999	1.0000
1000	0.6803	0.6670	1.0000	1.0000	0.9999	1.0000
	Review Ranking					
	YelpChi		YelpNYC		YelpZip	
k	Sp'le	SpLite	Sp'le	SpLite	Sp'le	SpLite
100	0.9354	0.9334	0.9694	0.9651	0.9219	0.9377
200	0.8469	0.8007	0.9665	0.9595	0.9200	0.9379
300	0.7373	0.6986	0.9597	0.9584	0.9216	0.9377
400	0.6682	0.6397	0.9615	0.9571	0.9248	0.9360
500	0.6255	0.6103	0.9610	0.9529	0.9234	0.9276
600	0.6089	0.5740	0.9620	0.9432	0.9236	0.9121
700	0.5864	0.5556	0.9552	0.8925	0.9240	0.9021
800	0.5587	0.5317	0.9179	0.8351	0.9199	0.8977
900	0.5458	0.5279	0.8775	0.7923	0.9138	0.8899
1000	0.5361	0.5218	0.8463	0.7577	0.9052	0.8810

4.3.4 SpLite performance

In light of our analysis results, we aim to design a "light" version of SpEagle that is computationally more efficient. Our analyses suggest that (1) review priors alone are the most effective, and achieve comparable performance to using priors for all user, review, and product nodes, and that (2) behavioral features are superior to text-based features.

The computationally most demanding component of SpEagle is feature extraction (network inference is only linear-time in number of edges in the graph [111]). Armed with the above conclusions, our goal is then to identify a few *behavioral* features for only the *review* nodes to be used in estimating priors fast. The rest of the priors, i.e., those for user and product nodes, are to be set to unbiased.

Figure 4.5 shows the AP performance of SpEagle (R) (as discussed in Table 4.8) for the review ranking task on all three datasets, when all the review features (blue bar), individual review features (green bars), as well as pairs of behavioral features (red bars, only 3 shown) are used. We find that while there exists no single feature that produces high performance across all datasets, using only two behavioral features often yields similar performance to using all.

We design SpLite to utilize only the RD and EXT features for estimating priors for only the review nodes. The rest are set to unbiased. Table 4.9 compares SpEagle and SpLite under 1% labeled data across all datasets for both ranking tasks, and Figure 4.6 illustrates the running times. Notice that SpLite reduces the feature extraction and hence prior estimation overhead significantly, while yielding quite comparable performance to SpEagle.

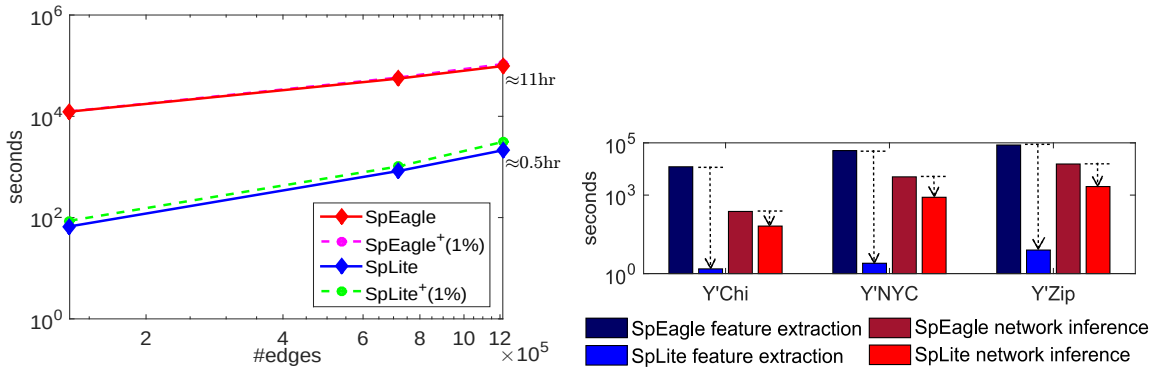


Figure 4.6: (left) Total running time of SpEagle vs. SpLite, (right) Break-down of runtime: feature extraction and network inference, for all datasets.

4.4 Summary of Contributions

We design a new holistic framework called SpEagle that exploits both relational data (user–review–product graph) and metadata (behavioral and text data) collectively to detect suspicious users and reviews, as well as targeted products of spam. Our main contributions described in this chapter are the following:

- SpEagle employs a review network-based classification task which accepts prior knowledge on the class distribution of the nodes estimated from meta-data.
- SpEagle works in an unsupervised fashion, but can easily leverage labels (if available). As such, we introduce a *semi-supervised* version called SpEagle$^+$ which improves performance significantly.
- We further design a *light* version of SpEagle called SpLite which uses a very small set of review features as prior information providing significant speed-up.

We evaluate our method on three real-world labeled (filtered vs. recommended) review datasets collected from Yelp.com. We provide the largest scale quantitative evaluation results on opinion spam detection. Our results show that SpEagle is superior to several baselines and state-of-the-art techniques. We share our code and datasets with ground truth at `http://shebuti.com/collective-opinion-spam-detection/`.

Chapter 5

Collective Opinion Spam Detection using Active Inference

Opinion spam has become a widespread problem in recent years. However, it remains challenging, as unlabeled data is abundant, but labels are difficult or expensive to obtain where human judges are only slightly better than random [95]. Scarcity of labeled data makes the supervised learning and evaluation hard [95, 109, 120]. As such, most existing works on opinion spam detection are unsupervised [98, 99, 103, 106, 121–123]. A useful trade-off is semi-supervised learning, which uses only a small set of labeled data to improve performance over the unsupervised setting. Most recently, Rayana et al. proposed a flexible approach called SpEagle [19] which addresses the spam detection problem as a network inference task on the review network, with the potential of seamless label incorporation. Importantly, they showed that a small fraction of labeled data improves the detection performance significantly.

In SpEagle, the labels of a *random* sample of nodes are consumed to improve the network inference performance. Our intuition suggests that it is possible to do better by a wise selection of only the "valuable" nodes with lower labeling cost. The very field of selecting valuable instances for acquiring labels from an oracle (e.g., human) within a small budget for learning improved models with lower cost is widely known as *active learning* [124, 125]. Like active learning, *active inference* is a well known approach [126, 127] that targets to achieve higher accuracy with fewer labeled examples given a small budget, where the existing inference model can pose queries to choose most valuable instances which are labeled by an oracle. The difference between active learning and inference is that the former re-trains the model whenever new instances are obtained from the oracle, whereas the latter assumes that a model already exists and the new labeled instances by the oracle are used during the inference. The most challenging task of active inference is the

selection of valuable instances for label acquisition.

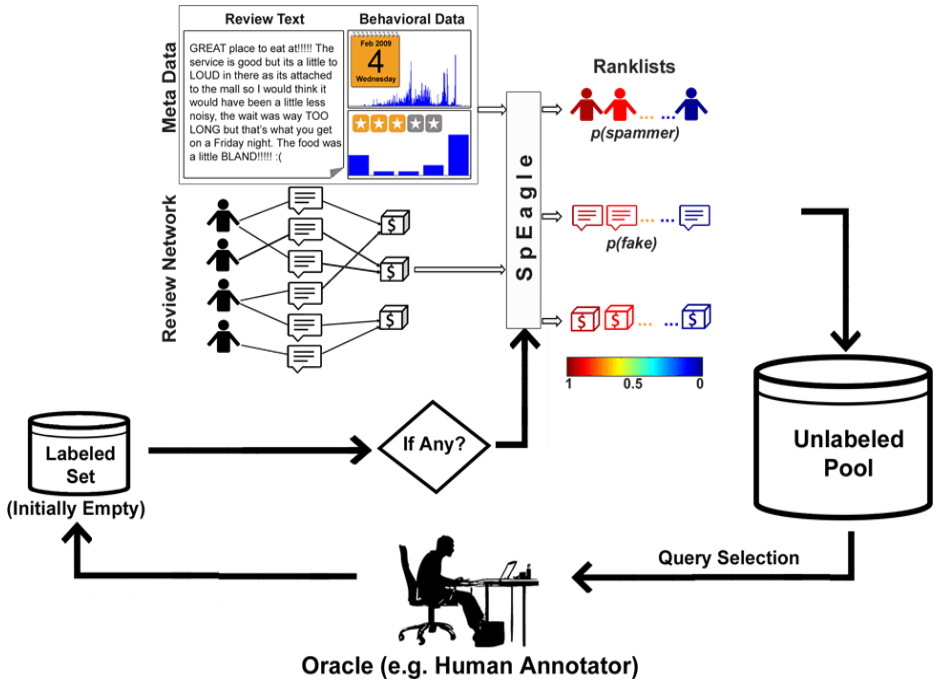


Figure 5.1: Label acquisition by selecting valuable nodes and seamlessly incorporating those labels in SpEagle framework.

A number of successful label acquisition approaches have been proposed in active learning as well as active inference literature [124–127]. Selecting valuable instances for these label acquisition techniques can be divided into two main types based on the nature of the data, (i) using flat data (no network) [128, 129], and (ii) using relational information of the data (network structure) [126, 127, 130]. In this work, we address the problem of active inference for the opinion spam detection problem described in Chapter 4. Our goal is to achieve improved performance over random selection within a small budget. Our main contributions are as follows.

- We adapt several existing label acquisition approaches of active inference into our collective opinion spam detection framework called SpEagle [19], in order to wisely select valuable nodes to label.
- We present three important characteristics of a valuable node in a label acquisition strategy: (i) uncertainty of a node, (ii) density of the region it belongs to, and (iii) proximity of the node to other uncertain nodes. Intuitively, a node is more valuable if its uncertainty is high and it is close-by to other uncertain nodes in a dense region, hence, acquiring the label of this node helps other uncertain nodes, through diffusion of its label information within the neighborhood of this node.

- Based on the above characteristics, we devise a label acquisition strategy called Expected UnCertainty Reach (EUCR) that wisely selects valuable nodes within a small budget (see the work flow in Figure 5.1) to improve performance significantly over the random selection.

We evaluate our method on two real-world datasets collected from Yelp.com, containing filtered (spam) and recommended (non-spam) reviews. To the best of our knowledge, this is the first work using active inference for opinion spam detection with a large-scale evaluation on real-world datasets. Our experiments shows that Expected Uncertainty Reach (EUCR) outperforms random sampling and several state-of-the-art active inference approaches.

5.1 Related Works

Although both active learning and active inference are widely explored in the literature (detailed survey in [124, 125]), there are only a few recent works on active learning (or inference) applied to network data. In network data, often the label of a node is influenced by its neighborhood. Hence, the common intuition is that knowing the label of a particular node can help inferring labels of other nodes in its neighborhood. In [127], Rattigan et al. proposed to select the most central nodes for labeling in order to get more significant impact. However, exploiting network structure sometimes becomes disadvantageous. In collective classification, wrong labels can be propagated throughout the network, misclassifying other unlabeled nodes. Bilgic et al. [126] proposed a collective classification approach called *Reflect and Correct (RAC)* to find islands of misclassification to correct the labels of a few nodes to improve performance. However, RAC requires an initial labeled training graph to find misclassification. They also proposed in [126] a greedy approach called *AIGA* which acquires the labels by minimizing the expected error (e.g., log loss). In AIGA, the expected error is minimized by considering all possible labels for each network instance, which is intractable to be applied on large-scale graphs. A similar approach to AIGA called Expected Risk Minimization (ERM) has been proposed by Macskassy [131] which provides significant speed up over AIGA by leveraging the graph structure (e.g., betweenness centrality) to initialize good candidates for labeling. Moreover, there are some scenarios where both meta information and network structure are available (like our spam detection problem). The work in [130] proposed an algorithm ALFNET where they utilize two classifiers (i) content-only, and (ii) collective classifier to combine their prediction for label acquisition. In particular, the nodes on which the two classifiers have disagreement are considered to be good candidates. While this approach has been tested on document classification, we adapt it to our opinion spam detection setting and compare to our

proposed approach EUCR. To the best of our knowledge, ours is the first work on active inference for collective opinion spam detection, where we utilize all of network, meta-data, and active label acquisition simultaneously.

5.2 Active Network Inference

Active inference addresses the problem of minimizing the labeling cost while maximizing the classification performance. The key idea is to achieve higher accuracy with fewer labeled examples given a budget, where the existing model of inference can pose queries to choose most "valuable" data instances which are labeled by an oracle (e.g., human). These labels are then used at inference time. The goal is to devise an effective strategy to identify such "valuable" nodes and a metric to quantify the "value" of instances (i.e., nodes). There exist a variety of active inference settings [124, 125]. In this work, we utilize the pool-based setting, in which the collective classifier is initially provided with a pool $\mathcal{P}$ of unlabeled nodes. At each iteration it selects the most informative node, adds it to the labeled set $\mathcal{L}$ and removes from $\mathcal{P}$ until the budget $\mathcal{B}$ (given) is exhausted.

Given a set of unlabeled nodes $\mathcal{U}$, we address the problem of finding the most valuable node to be labeled at each step iteratively, so as to improve the performance within a budget $\mathcal{B}$. We update the beliefs of all nodes each time a new label is acquired. In this work, we utilize some state-of-the-art label acquisition strategies, such as, random sampling, uncertainty sampling and query-by-committee approaches. We also adapt ALFNET [130] by modifying it to work with our network inference setting, using the metadata as well as relational information of the review network for label acquisition. Finally, we propose an efficient label acquisition strategy which we call Expected UnCertainty Reach (EUCR). We mainly build on uncertainty sampling, where our key insight is to select nodes that (i) exhibit high uncertainty, (ii) reside in a dense region, and (iii) are close-by to other uncertain nodes in the network (hence "reach"). We first describe how we adapt existing approaches to our setting in Sections 5.2.1 through 5.2.4, and later introduce our proposed approach in Section 5.2.5. We denote the most valuable node with x_A^*, where A is the label acquisition strategy. Here, we incorporate the label acquisition in our network inference framework for review nodes only. We consider Yelp.com to be our oracle, as they provide recommended and filtered reviews.

5.2.1 Random Sampling (RS)

In random sampling strategy, we randomly pick a review node for labeling and

simply set its prior as $\{\epsilon, 1-\epsilon\}$ if it belongs to the spam class (i.e., fake), and $\{1-\epsilon, \epsilon\}$ otherwise. This random selection is done iteratively until the budget $\mathcal{B}$ is exhausted. This is the strategy used in [19]. Our goal is to improve over this baseline with careful selection of "valuable" nodes to query the oracle.

5.2.2 Uncertainty Sampling (US)

Uncertainty sampling [128,132] is perhaps the simplest and most commonly applied approach in active inference. In this framework, we select the node for which the model is most uncertain and label it by the oracle (i.e., Yelp.com). For example, while using SpEagle for binary classification of the network entities (users, reviews, and/or products), uncertainty sampling selects the node whose final belief is near 0.5. Hence, we utilize a general entropy-based uncertainty sampling approach [128], in which we compute the entropy of the final beliefs as the uncertainty measure given in Eq. (5.1):

$$x^*_{US} = \underset{x}{\operatorname{argmax}} - \sum_i b_x(y_i) \log b_x(y_i) \tag{5.1}$$

Here $b_x(y_i)$ is the belief of node x to belong to class y_i, that is the marginal probability of assigning each node of type $\{R\}$ with the label y_i from label domain $\mathcal{L}_R = \{genuine, fake\}$. As such, at each iteration we select the review node with the highest uncertainty score to be labeled by the oracle. We incorporate the provided label by the oracle by initiating the review's priors as $\{\epsilon, 1-\epsilon\}$ if it is labeled as fake, and $\{1-\epsilon, \epsilon\}$ otherwise.

5.2.3 Query-by-Committee

Query-by-Committee (QBC) is an effective method of sampling for active inference where disagreement among different committee members is exploited to select nodes for labeling. QBC approach involves maintaining a committee $C = \{\theta^{(1)}, \ldots, \theta^{(|C|)}\}$ of models which represent competing hypotheses. Each committee member is then allowed to vote on the labeling of candidate nodes (i.e., reviews). The most informative candidates are those about which the committee members disagree the most. The key requirements of the QBC approach are (i) constructing a committee of models that represent different regions of input space and (ii) a measure of disagreement among the committee members.

(i) Committee Building: In this work, we build the committee by selecting 4 features out of 16 review features at random without replacement four times. This gives us a committee of four members each utilizing 4 review features to compute their priors (see Section 4.2.2).

(ii) Disagreement measure: We utilize the average *Kullback-Leibler (KL) Divergence* proposed by MacCallum et al. [129] as our disagreement measure which is an information-theoretic approach to calculate the difference between two probability distributions. This strategy is called the *soft voting (SV)* and represented by Eq. (5.2):

$$x^*_{QBC-SV} = \underset{x}{\text{argmax}} \frac{1}{|C|} \sum_{c=1}^{|C|} D(b_x^{\theta^{(c)}} || b_x^C) \tag{5.2}$$

where, $D(b_x^{\theta^{(c)}} || b_x^C) = \sum_i b_x^{\theta^{(c)}}(y_i) \log \frac{b_x^{\theta^{(c)}}(y_i)}{b_x^C(y_i)}$.
Here $\theta^{(c)}$ represents a particular member model in the committee and C represents the whole committee. $b_x^C(y_i) = \frac{1}{|C|} \sum_{c=1}^{|C|} b_x^{\theta^{(c)}}(y_i)$ is the average belief that y_i is the correct label for node x. This soft voting measure considers the node as highest informative which has the largest average difference between the label distributions of any one committee member and the whole committee.

For a budget $\mathcal{B}$, at each iteration we select the review node with the highest disagreement score to be labeled by the oracle. We leverage the provided label during the inference in the next step, by initiating priors as $\phi_{x^*} = \{\epsilon, 1-\epsilon\}$ if selected review is labeled fake, and $\{1-\epsilon, \epsilon\}$ otherwise.

In addition to the above, we build two strategies (i) most-sure disagreement and (ii) least-sure disagreement, above the soft voting based QBC approach motivated by Sharma et al. [133]. Our approach is different from [133] in a sense that they use uncertainty of different features, whereas, we use disagreement of different committee members. Most-sure disagreement occurs if the committee members have *strong* and *conflicting* evidence about an instance and least-sure disagreement occurs if the committee members have *no conclusive* evidence about an instance. For example, when half of the committee members vote $fake$ and the other half vote *genuine* for the same node, then in most-sure disagreement the committee members are more certain about their decision (e.g., beliefs [0.01 0.99] for $fake$ and [0.99 0.01] for *genuine*), whereas, in least-sure disagreement the committee members are less certain about their decision (e.g., beliefs [0.45 0.55] for $fake$ and [0.55 0.45] for *genuine*). For the review network, we classify a node x based on the ratio $\frac{b_x(+)}{b_x(-)}$, where $b_x(+)$ ($b_x(-)$) is the belief of node x belonging to spam or positive class (non-spam or negative class):

$$y_x = \begin{cases} + & \text{if } b_x(+) > b_x(-), \\ - & \text{otherwise} \end{cases} \tag{5.3}$$

From the above equation, it follows that for a node x the committee member $\theta^{(c)}$ provides evidence for positive class if $\frac{b_x^{\theta^{(c)}}(+)}{b_x^{\theta^{(c)}}(-)} > 1$, and it provides evidence for negative class otherwise. Let P_x and N_x denote two sets, such that P_x contains committee members that provide evidence for positive class and N_x contains committee members that provide evidence for negative class:

$$P_x = \{\theta^{(c)} | \frac{b_x^{\theta^{(c)}}(+)}{b_x^{\theta^{(c)}}(-)} > 1\} \tag{5.4}$$

$$N_x = \{\theta^{(c)} | \frac{b_x^{\theta^{(c)}}(-)}{b_x^{\theta^{(c)}}(+)} > 1\} \tag{5.5}$$

Note that these two sets are defined around a particular node x. The total evidence for node x of belonging to the positive class and the negative class are calculated using the following Eq. (5.6) and (5.7) respectively:

$$E^+(x) = \prod_{\theta^{(c)} \in P_x} \frac{b_x^{\theta^{(c)}}(+)}{b_x^{\theta^{(c)}}(-)} \tag{5.6}$$

$$E^-(x) = \prod_{\theta^{(c)} \in N_x} \frac{b_x^{\theta^{(c)}}(-)}{b_x^{\theta^{(c)}}(+)} \tag{5.7}$$

Our investigation shows that we have to optimize several objectives at the same time to make this evidence framework work on top of the QBC approach:

- Committee members should disagree on node x (i.e., high average KL divergence score).
- For most-sure disagreement, both $E^+(x)$ and $E^-(x)$ need to be large.
- For least-sure disagreement, both $E^+(x)$ and $E^-(x)$ need to be small.

We define the overall evidence of node x as:

$$E(x) = E^+(x) + E^-(x) \tag{5.8}$$

This aggregation makes sense as the overall evidence $E(x)$ is large if both $E^+(x)$ and $E^-(x)$ are large and close to each other. Similarly, $E(x)$ is small when both

$E^+(x)$ and $E^-(x)$ are small. Furthermore, selecting an unlabeled node x for which $E(x)$ is largest (or smallest) will not guarantee that the committee members have disagreement on x. To guarantee the disagreement of committee members, we first rank the nodes in decreasing order of their soft voting score x_{QBC-SV} (measured by equation (5.2)) and take the top k nodes. Let S be the set of top k nodes on which the committee members disagree the most. The most-sure disagreement approach selects the node with the maximum overall evidence:

$$x^*_{QBC-MS} = \underset{x \in S}{\text{argmax}}\ E(x) \tag{5.9}$$

and, the least-sure disagreement approach selects the node with the minimum overall evidence:

$$x^*_{QBC-LS} = \underset{x \in S}{\text{argmin}}\ E(x) \tag{5.10}$$

Most of the existing works on active inference do not use any relational information for query selection during label acquisition. There exist some recent works which utilize the relational information among the instances to improve the selection strategy [126, 130]. However, requirement of an initial labeled training graph or non-scalable greedy approach [126] makes some existing techniques inapplicable in our spam detection setting. We utilize both metadata and relational information for label acquisition using two techniques, one of them is a modified version of ALFNET [130] and the other is Expected UnCertainty Reach (EUCR) that we propose in this work. We describe the relational label acquisition approaches in the following sections.

5.2.4 ALFNET

Proposed by Bilgic et al. [130], ALFNET is an active learning algorithm for collective classification. This algorithm uses two learners called CO (content-only) and CC (collective classifier), and combines their decision in order to select nodes for labeling. In particular, this algorithm considers those nodes to be informative for which the decisions of the two classifiers differ. It is assumed that the labels are acquired iteratively in batch size of k. At first, this algorithm clusters the nodes in the graph into C clusters using the network structure of the data. Then, it selects the k clusters which satisfy two important properties: (i) the decisions of CO and CC differ the most, and (ii) the decisions of the classifiers do not match with the already observed labels in the cluster. Based on these two properties, an overall score is computed for each cluster $c \in C$ and for a batch of size k, the top k clusters

$C_k \subset C$ are selected. From each of these top k clusters a node $x \in c_i$ $(i = 1, \ldots, k)$ is randomly selected for labeling. For further details on this algorithm, we refer the readers to the original work [130].

We modify this algorithm to work with our spam detection framework. Specifically, instead of using Iterative Classification Algorithm (ICA), we utilize SpEagle as the collective classifier CC and logistic regression (as original work) as the content-only classifier CO. The CO is trained based on the features extracted from meta-data. Furthermore, we construct a review–review network on which to perform the clustering, by connecting two reviews with an edge if they share at least one reviewer. Initially, we have no labels to train the CO classifier. Therefore, we first sort the clusters by size and select one random review node to query from each of the top k clusters with largest size. The acquired labels constitute the initial training set for CO. We also incorporate these labels to CC for inference. Following the initial selection, in each iteration we compute a score for each cluster based one the disagreement between CO and CC as well as the estimated labels in the clusters. We then select top k clusters based on these scores to draw a node x for querying from each cluster randomly.

The main constraints of ALFNET are that (i) it needs several iterations to acquire enough labels to be able to learn an effective CO classifier, (ii) CO is susceptible to high class-imbalance, as most of the acquired labels are non-spam, and (iii) it needs to re-train CO at every iteration. As a result, it requires more labels to improve performance over random sampling in our opinion spam detection setting.

5.2.5 Expected UnCertainty Reach (EUCR)

Finally, we propose a label acquisition approach which considers both the uncertainty of a node as well as the uncertainty of other nodes close-by to it in the network structure. Our main objective is to find "islands" of uncertainty, in which we aim to obtain correct classification for all the nodes by acquiring labels for only a few. Following our intuition, we find three important characteristics of a valuable node for label acquisition, those are (i) uncertainty of a node, (ii) density of the region the node belongs to, and (iii) its proximity to other uncertain nodes. As such, a node is more valuable to query it exhibits uncertain beliefs and resides close-by to other uncertain nodes in a dense region, such that acquiring its label could help the nodes in its proximate neighborhood.

Based on our intuition about the characteristics of a valuable node, we propose a scoring measure called Expected UnCertainty Reach (EUCR). In this method, to

address the first two characteristics of a valuable review node, we first calculate the weighted uncertainty score WUC_x for all review nodes $x \in R$ using Eq. (5.11).

$$WUC_x = -w_x \sum_i b_x(y_i) \log b_x(y_i) \tag{5.11}$$

where w_x is calculated from the user-degree of the corresponding review node as

$$w_x = \frac{UD_x - \min_{UD}}{\max_{UD} - \min_{UD}} .$$

In particular UD_x denotes the degree (total number of reviews) of the user who posted review x and $\min_{UD}$ and max_{UD} denote the minimum and maximum degree of a user node in the network. The weighted uncertainty (WUC) score gives higher values to those (review) nodes which are more uncertain and also reside in a denser region (i.e., have many other review nodes nearby).

We then rank the review nodes based on their WUC score and take the top k nodes. Let S be the set of top k nodes that we pick by the uncertainty of the nodes and density of the region they belong to. We want a review node x to be not only uncertain itself, but also close-by to many other uncertain review nodes. Importantly, due to the existence of homophily it is considered that the neighboring nodes have similar labels, thus acquiring label of a node from a dense region helps all nodes in that region. To quantify proximity, we leverage the review–review network (denoted by G_R) where two reviews are connected if they share the same reviewer, on which we compute the Random Walk with Restart probability vector p_x for x. $p_x(j)$ depicts the probability of reaching node j through a (infinitely long) random walk, with occasional restarts to x. As such, this probability captures proximity of j to x. Then for each node $x \in S$ we calculate the probability of reaching a node j for all $j \in R$ under random walk with restart as Eq. (5.12).

$$p_x = cWp_x + (1 - c)e_x \tag{5.12}$$

where, $1 - c$ ($c = 0.85$) is the teleportation probability, e_x is a unit vector containing 1 for node x and 0's for all other nodes, and W is the column normalized adjacency matrix of G_R. Eq. (5.12) defines a linear system problem, where we can write p_x as

$$p_x = (1 - c)(I - cW)^{-1}e_x \tag{5.13}$$

The most informative node is then the one with the maximum total uncertainty reach as weighted by proximity, i.e.,

$$x^*_{EUCR} = \underset{x \in S}{\operatorname{argmax}} \sum_{j \in R} p_x(j) \times WUC_j \tag{5.14}$$

In summary, we assume that a node is more valuable not only for its level of uncertainty and residence in a dense region but also with respect to the existence of other uncertain nodes in its proximity (or "reach").

5.3 Evaluation

We evaluate our approach on two real world datasets from Yelp.com, `YelpChi` and `YelpNYC` described in Chapter 4. These datasets consist of recommended as well as filtered reviews from Yelp. We describe the evaluation metrics, followed by performance results.

Evaluation Metrics We use three evaluation metrics to measure the performance of our approach and all other compared approaches. We generate the precision vs. recall (PR) curve for different thresholds and calculate the area under the curve to get Average precision (AP). For spam detection in an imbalanced dataset (fake class represents minority), often the top positions in the ranking are more important. Therefore, we also calculate (i) *precision@k* and (ii) *NDCG@k*, for $k = 100, 200, \ldots, 1000$ to provide the performance on the top positions of the ranking.

Performance Results We compare the performance of different label acquisition approaches described in Section 5.2. Fig. 5.2 provides the AP curves with varying budget (e.g., budget $= 0, 1, 2, \ldots, 500$) of compared approaches for both user and review ranking. In review ranking, EUCR outperforms all the competing approaches for both the datasets, except for ALFNET in `YelpNYC`. However, ALFNET starts performing well only after around 350 labels are acquired, whereas, EUCR does better with fewer labels. Experiments show that the sudden performance increase of ALFNET is due to acquiring more $fake$ labels only after 300 labels, which improves the learning of CO and also helps the inference of CC. Specifically, out of the first 300 acquired labels only 26 are from $fake$ class, whereas, for the next 200 acquired labels 123 are $fake$. Thus acquiring some balance between the number of $fake$ and *genuine* labels improves ALFNET's performance. However, balance is not always guaranteed—even after 500 labels for `YelpChi`, ALFNET performs worse than RS. Fig. 5.3 shows the NDCG curves of the compared methods with varying budget (e.g., budget $= 0, 1, 2, \ldots, 500$) and fixed k (e.g., $k = 100, 1000$) to better depict the performance of review ranking. Again, the *NDCG* curves for compared methods show similar trend as the AP curves for review ranking.

Our analysis shows that the US and QBC approaches, which do not consider the network structure in label acquisition end up selecting nodes which may be most

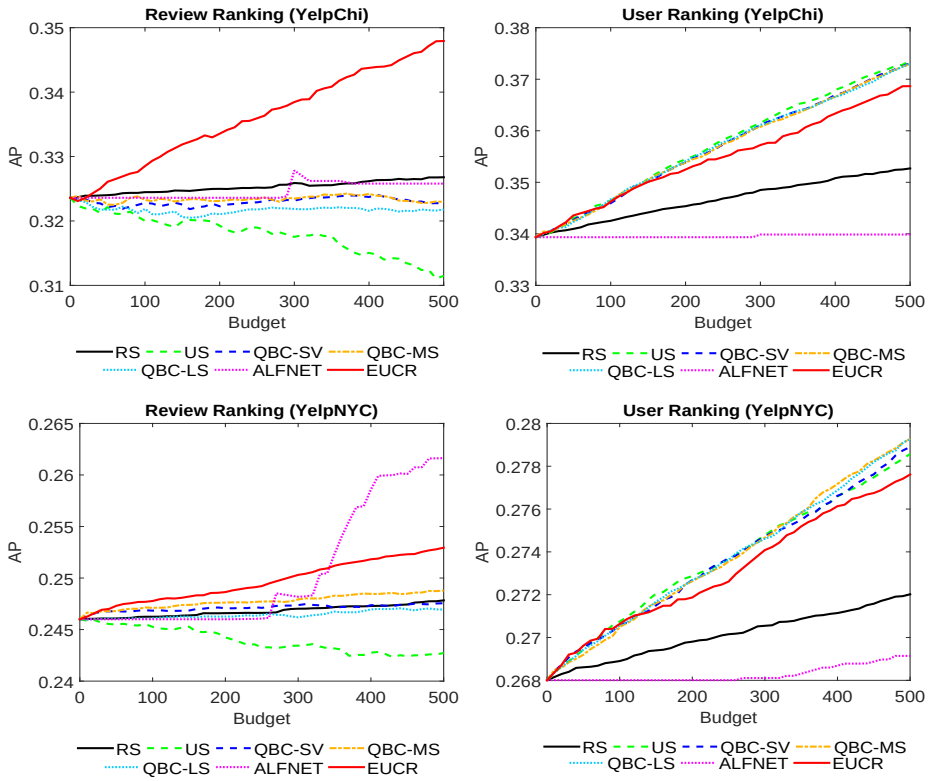


Figure 5.2: *AP of compared methods on* `YelpChi` *(top),* `YelpNYC` *(bottom) for both user and review ranking.*

uncertain (or most disagreed upon), however, not representative (i.e., close-by) of some or many other nodes. Although acquiring the labels for such review nodes are useful for the classification of corresponding users, they are not assisting other review nodes. As a result, both US and QBC (SV, MS, LS) have significant performance improvement for user ranking, but same is not true for review ranking as depicted in Fig. 5.2 and Fig. 5.3. For user ranking these baseline approaches provide very close results. Our EUCR approach also shows a comparable trend to those baselines.

Our analyses show that when a label acquisition method selects reviews written by different users then it is likely that more users get correct labels (i.e., spammer or benign), hence, performance improves for user ranking. Again, label acquisition of more $fake$ reviews which are also representative of neighboring $fake$ reviews improves the performance of review ranking. In Table 5.1, we provide the statistics of the labeled reviews and their corresponding users for different compared label acquisition approaches on both `YelpChi` and `YelpNYC` datasets with budget 500. This summary shows that our proposed label acquisition approach EUCR provides labels to the reviews of different users, resulting approximately as many correct user classification as the budget size. On the other hand, ALFNET acquires labels for

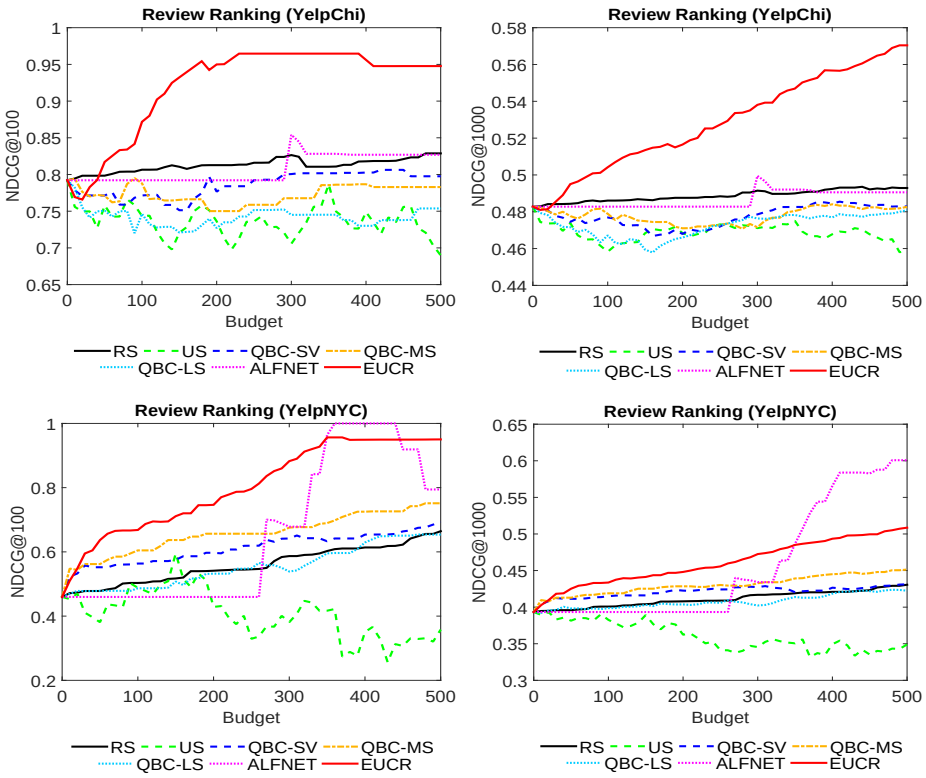


Figure 5.3: *NDCG*@100 *(left) and* *NDCG*@1000 *(right) of compared methods on* `YelpChi` *(top),* `YelpNYC` *(bottom) for review ranking with varying budget* $(0, 1, \ldots, 500)$.

multiple reviews of the same user, allowing label passing to fewer number of unique users, hence, reducing user ranking performance. EUCR also has some balance between number of $fake$ and *genuine* reviews compared to RS and ALFNET. As our datasets are imbalanced (fake being minority), RS gets the most imbalanced number of $fake$ vs. *genuine* label passing. Although the summary shows better statistics for US and QBC, these approaches do not achieve the expected performance due to selfishly selecting uncertain (or disagreed) nodes without considering the neighborhood information of the corresponding nodes in the network structure.

We further provide the *NDCG@k* curves in Fig. 5.4 for varying top k (e.g., $k = 100, 200, \ldots, 1000$) nodes and budget = 500, to better describe the ranking performance of compared methods, for both user and review ranking. Our EUCR approach outperforms all other compared approaches significantly on `YelpChi` for review ranking. However, ALFNET performs better than EUCR on `YelpNYC` with budget 500. The same arguments for Fig. 5.2 hold here as well. Having smaller budget (e.g., budget = 300), EUCR outperforms ALFNET (due to imbalanced labeled data) as well as other approaches, as depicted in Fig. 5.5. Recall that

Table 5.1: *Summary of* 500 *labeled reviews of compared method for* `YelpChi` *and* `YelpNYC` *datasets.*

	`YelpChi`		`YelpNYC`	
Methods	#unique users	#fake/#gen. labeled	#unique users	#fake/#gen. labeled
RS	494	62 / 438	500	53 / 447
US	500	231 / 269	500	205 / 295
QBC-SV	500	186 / 314	476	164 / 336
QBC-MS	500	183 / 317	494	154 / 346
QBC-LS	500	184 / 316	495	163 / 337
ALFNET	483	19 / 481	335	149 / 351
EUCR	498	139 / 361	500	131 / 369

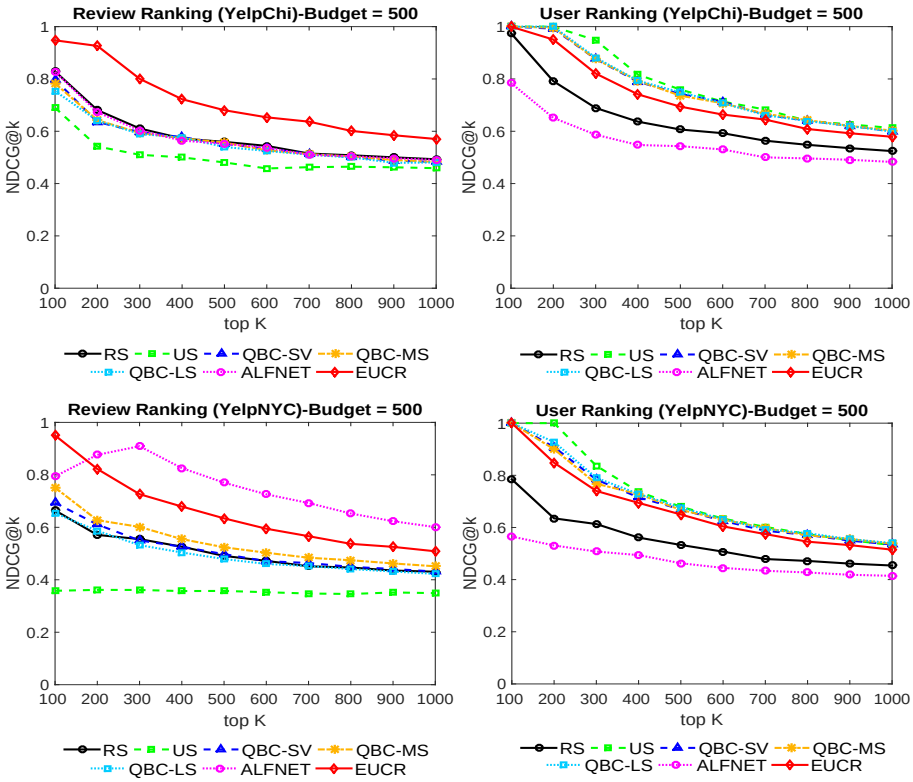


Figure 5.4: *NDCG@k on* `YelpChi` *(top),* `YelpNYC` *(bottom) for both user and review ranking with budget 500.*

besides a larger budget (i.e., training data), ALFNET requires a balanced labeled set to perform well and re-trains its local feature-based classifier *CO* at every step.

In Table 5.2 we also show the *precision@k* values for review ranking under

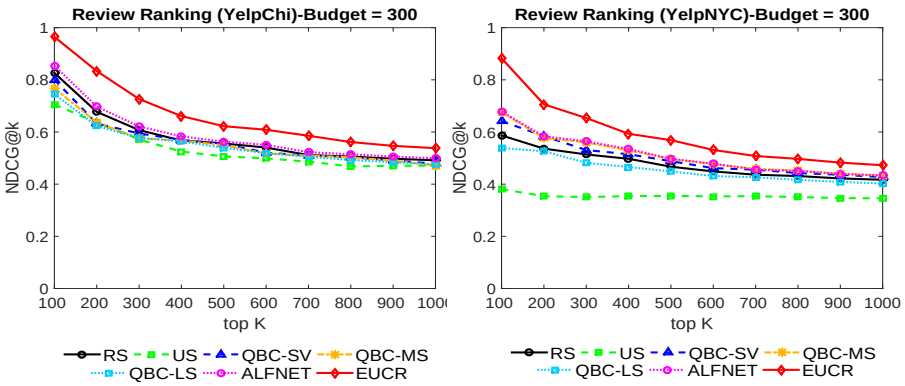


Figure 5.5: *NDCG@k on* `YelpChi` *(left),* `YelpNYC` *(right) for review ranking with budget 300.*

Table 5.2: *Precision@k for review ranking on* `YelpChi` *and* `YelpNYC` *with budget 300.*

	`YelpChi`						
k	**RS**	**US**	**Q'-SV**	**Q'-MS**	**Q'-LS**	**A'NET**	**EUCR**
100	0.78	0.64	0.77	0.75	0.70	0.81	**0.98**
200	0.62	0.59	0.58	0.60	0.57	0.64	**0.81**
300	0.55	0.52	0.55	0.53	0.53	0.56	**0.68**
400	0.51	0.48	0.53	0.53	0.53	0.53	**0.61**
500	0.51	0.46	0.52	0.52	0.50	0.51	**0.57**
600	0.50	0.46	0.48	0.48	0.48	0.51	**0.56**
700	0.47	0.45	0.47	0.48	0.47	0.48	**0.54**
800	0.47	0.43	0.47	0.47	0.46	0.47	**0.52**
900	0.46	0.44	0.46	0.46	0.45	0.47	**0.50**
1000	0.46	0.44	0.45	0.44	0.45	0.46	**0.50**
	`YelpNYC`						
k	**RS**	**US**	**Q'-SV**	**Q'-MS**	**Q'-LS**	**A'NET**	**EUCR**
100	0.51	0.41	0.57	0.60	0.49	0.60	**0.85**
200	0.49	0.36	0.54	0.52	0.50	0.52	**0.65**
300	0.47	0.35	0.48	0.51	0.45	0.52	**0.60**
400	0.46	0.36	0.48	0.49	0.44	0.49	**0.54**
500	0.43	0.36	0.45	0.45	0.42	0.45	**0.52**
600	0.42	0.36	0.42	0.44	0.41	0.44	**0.48**
700	0.41	0.36	0.42	0.42	0.40	0.42	**0.46**
800	0.40	0.35	0.41	0.42	0.40	0.42	**0.45**
900	0.39	0.35	0.41	0.41	0.39	0.40	**0.44**
1000	0.39	0.35	0.40	0.40	0.38	0.40	**0.43**

fixed budget = 300, to provide further evidence of the ranking performance of the compared methods. Once again, our EUCR approach outperforms RS as well as all other label acquisition approaches on both `YelpChi` and `YelpNYC` datasets.

In conclusion, our analyses show that most state-of-the-art approaches perform better than random sampling for only one type of ranking, users or reviews. In contrast, our EUCR approach achieves significant improvement over random selection for both user and review ranking. Specifically, with a small budget of 300 it improves *NDCG@100* by 20–34%, and *precision@100* by 14–30% for review ranking, and for user ranking it improves *NDCG@100* by 9–23%, and *precision@100* by 12–28%, over random sampling used in [19].

The `YelpNYC` dataset is 5 times larger than the `YelpChi` and the network inference algorithm is linear in the size of the network. As such, we apply a batch mode selection strategy for the review nodes for `YelpNYC` with a batch size of k (e.g., $k = 5$) for US, QBC (SV, MS, LS) and EUCR (ALFNET is a batch mode selection by construction) for speed-up. In batch mode selection strategy, we select the top k review nodes having k distinct users to label by the oracle to ensure informativeness as well as diversity of the selected nodes. As a results, no more than one review of the same user is selected in a batch.

5.4 Summary of Contributions

In this work, we extend the semi-supervised opinion spam detection framework SpEagle [19] with active inference, by carefully selecting valuable nodes (for acquiring labels from the oracle) within a small budget based on the network structure. The main contributions of the work presented in this chapter are:

- We show how to adapt existing general label acquisition techniques of active inference for the semi-supervised relational inference setting.
- We present three useful characteristics of a valuable node for querying: (i) uncertainty, (ii) neighborhood density, and (iii) proximity to other uncertain nodes.
- We propose a new label acquisition approach called Expected UnCertainty Reach (EUCR) for relational active inference, which selects uncertain nodes from dense regions within reach to other uncertain nodes.

We evaluate our method on two large datasets from Yelp.com, where EUCR outperforms random selection as well as other state-of-the-art approaches.

Chapter 6

Conclusion

In this Chapter, we first summarize our major contributions and then conclude with some interesting future research directions.

6.1 Summary of Contributions

In this thesis, we focused on two ways of designing anomaly mining methods, (i) ensemble learning, and (ii) multimodal learning. We summarize our contribution towards both types of approaches in the following.

Ensemble Learning:
In Chapter 2 and Chapter 3, we developed two novel ensemble learning methods for anomaly mining, each addressing the problem in different setting. In particular, we address the problem of (i) event detection in dynamic graphs, and (ii) outlier detection in multidimensional point data. We solve these problems with our novel ensemble learning approaches. We briefly present the summary of our contributions in the following.

- **Selective ensemble learning:** We proposed two novel methods, SelectH and SelectV, to detect events in dynamic graphs and outliers in multidimensional point datasets. The previous state-of-the-art ensemble methods have several limitations, such as, combining all the detectors or combining only diverse detectors the ensemble get hurt by the presence of inaccurate detectors. We developed our method to overcome these limitations and design two unsupervised selective approaches, which wisely discards the inaccurate detectors and only select the accurate ones to combine.
- **Sequential ensemble learning:** We developed a sequential ensemble

method, CARE, to detect outliers in multidimensional point datasets. None of the previous state-of-the-art outlier ensembles addressed the problem of reducing both bias and variance to improve the performance. In CARE, we proposed a two phase, (i) parallel, and (ii) sequential aggregation approach with filtering top outliers in each iteration to reduce both bias and variance. We present experiments with synthetic datasets to show this reduction to support our claim. We design multiple versions of CARE with different base detectors, e.g., distance based, density based and isolation based detectors. We showed that isolation based iCARE approach improves performance the most with respect to both accuracy and scalability.

Multimodal Learning:
In Chapter 4, we developed a multimodal learning approach for anomaly mining and addressed the problem of collective opinion spam detection in online review network and further improve the approach with active inference in Chapter 5. We briefly present the summary in the following.

- **Collective opinion spam detection, bridging review networks and metadata:** We designed a new holistic opinion spam detection approach called SpEagle. Existing approaches to opinion spam have successfully but separately utilized linguistic clues of deception, behavioral footprints, or relational ties between agents in a review system. In SpEagle, we utilize multiple modalities of the review data, e.g., the meta information (text, timestamp, rating) and relational information (network), and harness them collectively under a joint framework to spot suspicious users and reviews, as well as products targeted by spam. Moreover, we incorporated a semi-supervised version of SpEagle called SpEagle$^+$, which seamlessly absorb labels to improve performance.
- **Collective opinion spam detection with active inference:** This is the first work which addressed the active inference problem in collective opinion spam detection. We adapted the existing active inference models to collective classification of opinion spam and designed a new utility measure, EUCR to choose important nodes from the review network for labeling in each iteration to improve performance of detection.

6.2 Future Research Directions

Anomaly mining is an exciting research area which has extensive use in a wide variety of application domains. In future, I am interested in working on domain

specific as well as general anomaly mining problems. In the following, I describe some specific directions of future research works.

Online algorithms: Now-a-days, most real-world application domains generate data that is dynamic in nature. Mining anomalies in such dynamic datasets with offline techniques require to execute the detection algorithm on the whole extended dataset as new data arrives. As the size of the data increases rapidly with time, running offline methods periodically is not feasible. Most of the existing online anomaly detectors use some kind of supervision and there are very few unsupervised online detectors. Hence, designing online versions of our ensemble and mutimodal learning approaches will be an interesting future research direction.

Scalability: There is continuous advancement in data collection and storage as real-world data is keep growing. It needs peta-bytes of storage to process such big data. Therefore, the major bottleneck of any data mining algorithm design is the scalability. Previously, the main focus of an algorithm design was the accuracy of the algorithm. However, now-a-days researchers focus on both accuracy and running time as the algorithms need to process data that does not fit into the main memory. As such, an exciting future research would be to design effective approximation algorithms which require linear time and space. Moreover, designing anomaly ensemble algorithms which exploit parallelism can also be an interesting future work.

Anomaly ensemble for complex heterogeneous graphs: Majority of the existing works in graph anomaly mining are designed to work for simple undirected homogeneous graphs. However, real-world graphs are much complex having various types of nodes, edges with weights, node and edge attributes etc. Most of the social media graphs, such as, Facebook, Twitter etc. has heterogeneous structure and representing them as plain undirected graph results in losing a lot of information. There is lack of work in finding anomalies from such complex graphs. Therefore, designing effective and robust ensemble approaches for complex heterogeneous graphs can be a very exciting future research direction.

Appendix A

Multidimensional Point Datasets

Evaluating our proposed anomaly mining approaches on real-world datasets is extremely important to understand and measure their performance in various unpredictable scenarios. Following is a description of the outlier detection datasets (all available at http://odds.cs.stonybrook.edu/#table1) employed in Chapter 2 and Chapter 3. Most of these datasets are originally available at UCI Machine Learning repository [70] as classification data.

- **Lympho:** The Lymphography (Lympho) dataset is a multi-class dataset having 18 dimensions and four classes, two of them are quite small (2 and 4 data records). Therefore, those two small classes are merged and considered as outliers compared to other two large classes (81 and 61 data records) which are considered as inliers.
- **WBC:** The Wisconsin Breast Cancer (diagnostics) dataset (WBC) contains two classes *malignant* and *benign*. This is a 30 dimensional dataset. In [68], they started with a processed version[1] of the WBC dataset and the instances in malignant class are down sampled to 21 instances to get the outliers and all the instances of benign class are considered as inliers.
- **Glass:** The Glass dataset was obtained for the study of classification of types of glass which was motivated by criminological investigation. At the scene of the crime, the glass left can be used as evidence, if correctly identified. This dataset contains 9 attributes regarding several glass types (multi-class). Here, class 6 is a clear minority class, as such points of class 6 are marked as outliers, while all other points are inliers.

[1] http://www.ipd.kit.edu/~muellere/HiCS/

- **Vowels:** The Japanese Vowels (Vowels) dataset is a multivariate time series data, where nine male speakers uttered two Japanese vowels /ae/ successively. Here, one utterance by a speaker forms a time series whose length is in the range 7-29 and each point of a time series is of 12 features (12 coefficients). For outlier detection, each frame in the training data is treated as an individual data point, whereas the UCI repository treats a block of frames (utterance) as an individual point. In this case, class (speaker) 1 is downsampled to 50 outliers. The inliers contained classes 6, 7 and 8. Other classes are discarded.

- **Cardio:** The Cardiotocography (Cardio) dataset consists of measurements of fetal heart rate (FHR) and uterine contraction (UC) features on cardiotocograms classified by expert obstetricians. This is a classification dataset, where the classes are normal, suspect, and pathologic. For outlier detection, The normal class formed the inliers, while the pathologic class is downsampled to 176 points as outliers. The suspect class is discarded.

- **Thyroid:** The thyroid disease (Thyroid) dataset were used for training ANNs. It has 3772 training instances and 3428 testing instances. This dataset contains 15 categorical and 6 real attributes. The problem is to determine whether a patient referred to the clinic is hypothyroid. Therefore three classes are built: normal (not hypothyroid), hyperfunction and subnormal functioning. For outlier detection, 3772 training instances are used, with only 6 real attributes. The hyperfunction class is treated as outlier class and other two classes as inliers, because hyperfunction is a clear minority class.

- **Musk:** The Musk dataset contains multiple classes some of them are *mask* type and some *non-musk* type. The non-musk classes j146, j147, and 252 were combined to form the inliers, while the musk classes 213 and 211 were added as outliers without down-sampling. Other classes were discarded.

- **Optdigits:** The optical recognition of handwritten digits (Optdigits) dataset is a multi-class classification dataset of digits 0-9 having 64 dimensions. The instances of digits 1-9 are considered inliers and instances of digit 0 are down-sampled to 150 outliers.

- **Satimage-2:** The Statlog (Satimage-2) dataset is a multi-class classification dataset having 36 dimensions. For outlier detection, the training and test data are combined. Class 2 is down-sampled to 71 outliers, while all the other classes are combined to form an inlier class.

- **Letter:** The letter recognition (Letter) dataset is a multi-class classification dataset. The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English

alphabet, where letters of the alphabet are represented in 16 dimensions. This dataset was introduced as an outlier detection data in [71]. To get data suitable for outlier detection, they subsample data from 3 letters to form the normal class and randomly concatenate pairs of them so that their dimensionality doubles. To form the outlier class, they randomly select few instances of letters that are not in the normal class and concatenate them with instances from the normal class. The concatenation process is performed in order to make the detection much more challenging as each outlier also shows some normal attribute values.

- **Pima:** The Pima Indians diabetes (Pima) dataset is a binary classification dataset with 8 attributes. Several constraints were placed on the selection of instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage. The dataset is utilized as it is from the UCI repository.

- **Satellite:** The original Statlog Landsat Satellite (Satellite) dataset is a multi-class classification dataset. For outlier detection, the training and test data are combined. The smallest three classes, i.e. 2, 4, 5 are combined to form the outlier class, while all the other classes are combined to form inlier class.

- **BreastW:** The Breast Cancer Wisconsin (BreastW) dataset is a binary classification dataset with 9 attributes. This dataset records the measurements for breast cancer tests. There are two classes, benign and malignant. The malignant class of this dataset is considered as outliers, while points in the benign class are considered inliers.

- **Arrhythmia:** The Arrhythmia dataset is a multi-class classification dataset with dimensionality 279. There are five categorical attributes which are discarded for outlier detection, totalling 274 attributes. The smallest classes, i.e., 3, 4, 5, 7, 8, 9, 14, 15 are combined to form the outlier class and the rest of the classes are combined to form the inlier class.

- **Ionosphere:** The Ionosphere dataset is a binary classification dataset with dimensionality 34. There is one attribute having values all zeros, which is discarded for outlier detection. So the total number of dimensions are 33. The bad class is considered as outlier class and the good class as inlier.

- **Mnist:** The MNIST dataset (http://yann.lecun.com/exdb/mnist/) of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. This dataset is converted for outlier detection as digit-zero class is considered as inliers, while

700 images are sampled from digit-six class as the outliers. In addition, 100 features are randomly selected from 784 total features.

- **HAR:** This dataset contains 561 features of various human activities captured using different sensor readings (accelerometer and gyroscope) of smartphones [134]. It contains data of 6 human activities, e.g., walking, walking upstairs, walking down- stairs, sitting, standing and laying. The dataset is transformed in to an outlier detection dataset by labeling the activities which involve walking as inliers and other as outliers. This approach is followed based on the intuition that the activities which include walking is similar and possess different features from the other activities. The data points from other activities are down-sampled to 200 each as outliers.

- **isolet:** This dataset was introduced as an outlier detection dataset in [135]. It is originally a classification dataset consists of the features extracted from pronunciations of letters in English alphabet. It was transformed into an outlier detection dataset by labeling the letters C, D and E (letters that give e sound) as inliers and 10 instances of letter 'Y' as outliers.

- **mfeat:** This dataset was introduced as an outlier detection dataset in [135]. The original dataset consists of features of handwritten numerals (0 - 9) extracted from a collection of Dutch utility maps. It has 2000 data points (200 for each numeral). Its 649 features are a combination of multiple feature types extracted from the digit images. It was transformed to an outlier detection dataset by labeling the data points representing digits 6 and 9 as the inliers and 10 instances of digit 0 as outliers.

- **shuttle:** This dataset was introduced as an outlier detection dataset in [30]. It is a 9 dimensional dataset having 49,097 data points with 7.0% outliers. This data is collected from the US Space shuttle Challenger. It has 7 classes representing 7 states of the shuttles propulsion system. It was transformed to an outlier detection dataset by labeling the largest class (Rad flow) as the inliers (about 93% of the data belong to this class), and labeling rest of the classes (Fpv Close, Fpv Open, High, Bypass, Bpv Close, Bpv Open) as outliers.

- **mulcross:** This dataset is generated from a synthetic data generator named Mulcross [136]. It is a 4 dimensional dataset having 262,144 data points with 1.0% outliers. Mulcross generates a multivariate normal distribution with a selective number of outlier clusters. The mulcross dataset used in our work has 10% contamination ratio (number of outliers over the total number of points), distance factor 2 (distance between the center of normal cluster and outlier clusters), and 2 outlier clusters.

Appendix B

Additional Evaluation Results of SELECT

Table B.1: Accuracy of ensembles for Challenge Network: (feature: unweighted in & outdegree). $*$ depicts selected detector/consensus results.

	Algorithms	Full	DivE	ULARA(w_i)	SelectV	SelectH
Base Algorithms	EBED (uin)	0.1587	*	▬		*
	PTSAD (uin)	0.5208	*	▬		*
	SPIRIT (uin)	0.7556	*	▬	*	*
	ASED (uin)	1.0000	*	▬	*	*
	MAED (uin)	1.0000		▬	*	*
	EBED (uout)	0.7333	*	▬		*
	PTSAD (uout)	0.0994	*	▬		*
	SPIRIT (uout)	0.4021	*	▬	*	*
	ASED (uout)	1.0000	*	▬	*	*
	MAED (uout)	1.0000	*	▬	*	*
Consensus	Inverse Rank	* 1.0000	* 1.0000	–	* 1.0000	* 1.0000
	Kemeny-Young	* 0.9167	0.9167	–	* 1.0000	* 0.9167
	RRA	* 0.9167	* 0.9167	–	1.0000	* 0.9167
	Uni (avg)	* 1.0000	* 1.0000	–	* 1.0000	* 1.0000
	Uni (max)	* 0.2778	* 0.2778	–	1.0000	0.2778
	MM (avg)	* 1.0000	* 1.0000	–	* 1.0000	* 1.0000
	MM (max)	* 0.7000	0.7000	–	1.0000	* 0.7000
Final Ensemble		1.0000	1.0000	0.9167	1.0000	1.0000

Table B.1 shows the accuracies for SelectV, SelectH, Full, DivE, and ULARA on

CyberChallenge dataset for unweighted indegree and outdegree features (10 components), along with the accuracies of the base detectors and consensus methods. Similarly, Table B.3 shows the accuracies for all five detectors for weighted in and outdegree and unweighted in and outdegree (20 components) on CyberChallenge dataset. Table B.2 and Table B.4 represents the accuracies of all five detectors with 10 components and 20 components respectively for characterization of time tick 377 of CyberChallenge dataset.

Table B.2: Accuracy of ensembles for Challenge Network: Characterization of time tick 377 (Feature: unweighted in & outdegree). $*$ depicts selected detector/consensus results.

	Algorithms	Full	DivE	ULARA(w_i)	SelectV	SelectH
Base Algorithms	EBED (uin)	1.0000	*			*
	PTSAD (uin)	0.2000	*			*
	SPIRIT (uin)	0.5000	*			*
	ASED (uin)	1.0000			*	*
	MAED (uin)	1.0000			*	*
	EBED (uout)	0.2500				*
	PTSAD (uout)	0.2000				*
	SPIRIT (uout)	0.0270	*			
	ASED (uout)	1.0000	*			*
	MAED (uout)	1.0000	*		*	*
Consensus	Inverse Rank	* 1.0000	* 1.0000	–	* 1.0000	* 1.0000
	Kemeny-Young	* 1.0000	* 1.0000	–	1.0000	* 1.0000
	RRA	* 1.0000	* 1.0000	–	* 1.0000	* 1.0000
	Uni (avg)	* 1.0000	* 1.0000	–	* 1.0000	* 1.0000
	Uni (max)	* 0.5000	0.5000	–	1.0000	* 0.5000
	MM (avg)	* 1.0000	* 1.0000	–	0.2500	* 1.0000
	MM (max)	* 0.0147	* 0.0200	–	0.2500	0.0147
Final Ensemble		1.0000	1.0000	1.0000	1.0000	1.0000

Table B.3: Accuracy of ensembles for Challenge Network: (Feature: weighted in & outdegree and unweighted in & outdegree). $*$ depicts selected detector/consensus results.

	Algorithms	Full	DivE	ULARA(w_i)	SelectV	SelectH
Base Algorithms	EBED (win)	0.0054	*	▬		*
	PTSAD (win)	0.1183	*	▬	*	*
	SPIRIT (win)	0.0039		▬	*	*
	ASED (win)	0.0178	*	▬		*
	MAED (win)	0.0568		▬		*
	EBED (wout)	0.0064	*	▬		*
	PTSAD (wout)	0.6717	*	▬	*	*
	SPIRIT (wout)	0.0036		▬		*
	ASED (wout)	0.0104	*	▬	*	*
	MAED (wout)	0.0494	*	▬		*
	EBED (uin)	0.1587	*	▬		*
	PTSAD (uin)	0.5208	*	▬	*	*
	SPIRIT (uin)	0.7556	*	▬		*
	ASED (uin)	1.0000	*	▬		*
	MAED (uin)	1.0000		▬		*
	EBED (uout)	0.7333	*	▬		*
	PTSAD (uout)	0.0994	*	▬		*
	SPIRIT (uout)	0.4021	*	▬		*
	ASED (uout)	1.0000	*	▬		*
	MAED (uout)	1.0000	*	▬		*
Consensus	Inverse Rank	$*$ 0.8333	$*$ 0.8333	–	$*$ 0.4712	$*$ 0.8333
	Kemeny-Young	$*$ 0.4720	$*$ 0.5167	–	$*$ 0.1730	$*$ 0.4720
	RRA	$*$ 0.8095	$*$ 0.7778	–	$*$ 0.3827	$*$ 0.8095
	Uni (avg)	$*$ 0.5000	$*$ 0.6250	–	$*$ 0.0956	$*$ 0.0339
	Uni (max)	$*$ 0.2778	$*$ 0.2778	–	$*$ 0.3503	$*$ 0.0219
	MM (avg)	$*$ 0.0244	$*$ 0.3432	–	$*$ 0.0093	$*$ 0.0244
	MM (max)	$*$ 0.0151	$*$ 0.0151	–	$*$ 0.0499	0.0151
Final Ensemble		0.6325	0.6667	0.4549	0.3575	0.7500

Table B.4: Accuracy of ensembles for Challenge Network: Characterization of time tick 377 (Feature: weighted in & outdegree and unweighted in & outdegree). $*$ depicts selected detector/consensus results.

	Algorithms	Full	DivE	ULARA(w_i)	SelectV	SelectH
Base Algorithms	EBED (win)	0.0909	*	▬	*	*
	PTSAD (win)	1.0000	*	▬		*
	SPIRIT (win)	0.0238		▬		
	ASED (win)	0.3333		▬	*	*
	MAED (win)	1.0000	*	▬		*
	EBED (wout)	0.0385		▬		
	PTSAD (wout)	0.1429	*	▬		*
	SPIRIT (wout)	0.0133	*	▬		
	ASED (wout)	0.2500	*	▬	*	*
	MAED (wout)	1.0000	*	▬		*
	EBED (uin)	1.0000	*	▬		*
	PTSAD (uin)	0.2000	*	▬	*	*
	SPIRIT (uin)	0.5000	*	▬		
	ASED (uin)	1.0000		▬		*
	MAED (uin)	1.0000		▬	*	*
	EBED (uout)	0.2500		▬	*	*
	PTSAD (uout)	0.2000		▬		*
	SPIRIT (uout)	0.0270	*	▬		
	ASED (uout)	1.0000	*	▬		*
	MAED (uout)	1.0000		▬		*
Consensus	Inverse Rank	* 1.0000	* 1.0000	–	1.0000	* 1.0000
	Kemeny-Young	* 1.0000	* 1.0000	–	* 1.0000	* 1.0000
	RRA	* 1.0000	* 1.0000	–	1.0000	* 1.0000
	Uni (avg)	* 1.0000	* 1.0000	–	* 1.0000	* 1.0000
	Uni (max)	* 0.1667	* 0.2000	–	0.5000	0.3333
	MM (avg)	* 1.0000	1.0000	–	1.0000	* 1.0000
	MM (max)	* 0.0128	* 0.0128	–	0.0189	0.0135
Final Ensemble		1.0000	1.0000	1.0000	1.0000	1.0000

Bibliography

[1] Vipin Kumar. Parallel and distributed computing for cybersecurity. *IEEE Distributed Systems Online*, 6(10), 2005.

[2] Ryohei Fujimaki, Takehisa Yairi, and Kazuo Machida. An approach to spacecraft anomaly detection problem using kernel feature space. In *PAKDD*, pages 401–410, 2005.

[3] Nitin Jindal and Bing Liu. Opinion spam and analysis. In *WSDM*, pages 219–230, 2008.

[4] Arjun Mukherjee, Abhinav Kumar, Bing Liu, Junhui Wang, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. Spotting opinion spammers using behavioral footprints. In *KDD*, pages 632–640. ACM, 2013.

[5] Clay Spence, Lucas Parra, and Paul Sajda. Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model. In *Mathematical Methods in Biomedical Image Analysis, 2001. MMBIA 2001. IEEE Workshop on*, pages 3–10. IEEE, 2001.

[6] Sanjeev Jha and J Christopher Westland. A descriptive study of credit card fraud pattern. *Global Business Review*, 14(3):373–384, 2013.

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

[8] Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2000.

[9] Christine Preisach and Lars Schmidt-Thieme. Ensembles of relational classifiers. *Knowl. and Info. Sys.*, 14:249–272, 2007.

[10] Lior Rokach. Ensemble-based classifiers. *Artif. Intell. Rev.*, 33(1-2):1–39, 2010.

[11] Giorgio Valentini and Francesco Masulli. Ensembles of learning machines. In *WIRN*, 2002.

[12] Xiaoli Z. Fern and Wei Lin. Cluster ensemble selection. In *SDM*, pages 787–797. SIAM, 2008.

[13] Joydeep Ghosh and Ayan Acharya. Cluster ensembles: Theory and applications. In *Data Clustering: Alg. and Appl.*, pages 551–570. 2013.

[14] Charu C. Aggarwal. Outlier ensembles: position paper. *SIGKDD Explor. Newsl.*, 14(2):49–58, 2012.

[15] Arthur Zimek, Ricardo J.G.B. Campello, and Jörg Sander. Ensembles for unsupervised outlier detection: Challenges and research questions. *SIGKDD Explor. Newsl.*, 15(1):11–22, 2013.

[16] Shebuti Rayana and Leman Akoglu. Less is more: Building selective anomaly ensembles with application to event detection in temporal graphs. *SDM*, 17, 2015.

[17] Shebuti Rayana and Leman Akoglu. Less is more: Building selective anomaly ensembles. *Transactions on Knowledge Discovery from Data (TKDD)*, 2016.

[18] Shebuti Rayana, Wen Zhong, and Leman Akoglu. Sequential ensemble learning for outlier detection: A bias-variance perspective. *arXiv preprint arXiv:1609.05528*, 2016.

[19] Shebuti Rayana and Leman Akoglu. Collective opinion spam detection: Bridging review networks and metadata. In *KDD*, 2015.

[20] Shebuti Rayana and Leman Akoglu. Collective opinion spam detection using active inference. In *SDM*, 2016.

[21] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(10), 1990.

[22] Brijesh Verma and Ashfaqur Rahman. Cluster-oriented ensemble classifier: Impact of multicluster characterization on ensemble classifier learning. *IEEE Trans. Knowl. Data Eng.*, 24(4):605–618, 2012.

[23] Faisal Zaman and Hideo Hirose. Classification performance of bagging and boosting type ensemble methods with small training sets. *New Gen. Comput.*, 29(3):277–292, 2011.

[24] Stefan Todorov Hadjitodorov, Ludmila I. Kuncheva, and Ludmila P. Todorova. Moderate diversity for better cluster ensembles. *Information Fusion*, 7(3):264–275, 2006.

[25] Xiaoli Zhang Fern and Carla E. Brodley. Random projection for high dimensional data clustering: A cluster ensemble approach. In *ICML*, pages 186–193. AAAI Press, 2003.

[26] Alexander P. Topchy, Anil K. Jain, and William F. Punch. Clustering ensembles: Models of consensus and weak partitions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(12):1866–1881, 2005.

[27] Jun Gao, Weiming Hu, Zhongfei (Mark) Zhang, and Ou Wu. Unsupervised ensemble learning for mining top-n outliers. In *PAKDD*, 2012.

[28] Jing Gao and Pang-Ning Tan. Converting output scores from outlier detection algorithms to probability estimates. In *ICDM*, 2006.

[29] Hans-Peter Kriegel, Peer Krger, Erich Schubert, and Arthur Zimek. Interpreting and unifying outlier scores. In *SDM*, pages 13–24, 2011.

[30] Aleksandar Lazarevic and Vipin Kumar. Feature bagging for outlier detection. In *KDD*, pages 157–166. ACM, 2005.

[31] Nguyen Hoang Vu, Hock Hee Ang, and Vivekanand Gopalkrishnan. Mining outliers with ensemble of heterogeneous detectors on random subspaces. In *DASFAA*, volume 5981, pages 368–383, 2010.

[32] Erich Schubert, Remigius Wojdanowski, Arthur Zimek, and Hans-Peter Kriegel. On evaluation of outlier rankings and outlier scores. In *SDM*, pages 1047–1058, 2012.

[33] Arthur Zimek, Matthew Gaudet, Ricardo J. G. B. Campello, and Jörg Sander. Subsampling for efficient and effective unsupervised outlier detection ensembles. In *KDD*, pages 428–436. ACM, 2013.

[34] Shebuti Rayana and Leman Akoglu. An ensemble approach for event detection in dynamic graphs. In *KDD ODD*^{2} *Workshop*, 2014.

[35] Alexandre Klementiev, Dan Roth, and Kevin Small. An unsupervised learning algorithm for rank aggregation. In *ECML*, 2007.

[36] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph-based anomaly detection and description: A survey. *DAMI*, 28(4), 2014.

[37] Manish Gupta, Jing Gao, Charu C. Aggarwal, and Jiawei Han. Outlier detection for temporal data: A survey. *IEEE Trans. Knowl. Data Eng.*, 26(9):2250–2267, 2014.

[38] Robert E. Schapire. The strength of weak learnability. In *Machine Learning*, pages 197–227, 1990.

[39] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:214–259, 1992.

[40] Jennifer A. Hoeting, David Madigan, Adrian E. Raftery, and Chris T. Volinsky. Bayesian model averaging: A tutorial. *Statistical Science*, 14(4):382–417, 1999.

[41] A. Juditsky, P. Rigollet, and A. B. Tsybakov. Learning by mirror averaging. *Annals of Stat.*, 36(5):2183–2206, 2008.

[42] Kristine Monteith, James L. Carroll, Kevin D. Seppi, and Tony R. Martinez. Turning bayesian model averaging into bayesian model combination. In *IJCNN*, pages 2657–2663. IEEE, 2011.

[43] Alexandre B. Tsybakov. Optimal aggregation of classifiers in statistical learning. *Annals of Statistics*, 32:135–166, 2004.

[44] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[45] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: Identifying density-based local outliers. In *SIGMOD*, 2000.

[46] Spiros Papadimitriou, Hiroyuki Kitagawa, Phillip B. Gibbons, and Christos Faloutsos. LOCI: fast outlier detection using the local correlation integral. In *ICDE*, pages 315–326, 2003.

[47] Jing Gao, Wei Fan, Deepak Turaga, Olivier Verscheure, Xiaoqiao Meng, Lu Su, and Jiawei Han. Consensus extraction from heterogeneous detectors to improve performance over network traffic anomaly detection. In *IEEE International Conference on Computer Communications Mini-Conference*, 2011.

[48] Xin Luna Dong, Laure Berti-Equille, and Divesh Srivastava. Data fusion: Resolving conflicts from multiple sources. In *WAIM*, 2013.

[49] Douglas M Hawkins. *Identification of outliers*, volume 11. Springer, 1980.

[50] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Outlier detection: A survey. *ACM Computing Surveys*, 2007.

[51] Edwin M Knorr and Raymond T Ng. A unified notion of outliers: Properties and computation. In *KDD*, pages 219–222, 1997.

[52] Ke Zhang, Marcus Hutter, and Huidong Jin. A new local distance-based outlier detection approach for scattered real-world data. In *Advances in Knowledge Discovery and Data Mining*, pages 813–822. 2009.

[53] Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. LoOP: local outlier probabilities. In *CIKM*, pages 1649–1652, 2009.

[54] Leman Akoglu and Christos Faloutsos. Event detection in time series of mobile communication graphs. In *27th Army Science*, 2010.

[55] Spiros Papadimitriou, Jimeng Sun, and Christos Faloutsos. Streaming pattern discovery in multiple time-series. In *VLDB*, 2005.

[56] Anukool Lakhina, Mark Crovella, and Christophe Diot. Diagnosing network-wide traffic anomalies. In *SIGCOMM*, pages 219–230, 2004.

[57] Oskar Perron. Zur theorie der matrices. In *Mathematische Annalen*, pages 248–263, 1907.

[58] Diane Lambert. Zero-inflated poisson regression with an application to defects in manufacturing. In *Technometrics*, pages 1–14, 1992.

[59] Michael D. Porter and Gentry White. Self-exciting hurdle models for terrorist actovity. In *The Annals of Applied Statistics*, pages 106–124, 2012.

[60] A. Cameron and P.K. Trivedi. *Regression Analysis of Count Data*. Cambridge Univ. Press, 1st edition, 1998.

[61] Quang H. Vuong. Likelihood ratio tests for model selection and non-nested hypotheses. In *Econometrica*, 1989.

[62] A. H. Copeland. A 'reasonable' social welfare function. In *Seminar on Mathematics in Social Sciences*. University of Michigan, 1951.

[63] Cynthia Dwork, Ravi Kumar, Moni Naor, and D Sivakumar. Rank aggregation methods for the Web. In *WWW*, 2001.

[64] John Kemeny. Mathematics without numbers. In *Daedalus*, pages 577–591, 1959.

[65] Raivo Kolde, Sven Laur, Priit Adler, and Jaak Vilo. Robust rank aggregation for gene list integration and meta-analysis. *Bioinformatics*, 28(4):573–580, 2012.

[66] Gavin Brown, Jeremy L. Wyatt, Rachel Harris, and Xin Yao. Diversity creation methods: A survey and categorisation. *Information Fusion*, 6(1):5–20, 2005.

[67] L. Kuncheva and C. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51:181–207, 2003.

[68] Charu C Aggarwal and Saket Sathe. Theoretical foundations and algorithms for outlier ensembles. *ACM SIGKDD Explor. News.*, 17(1):24–47, 2015.

[69] Nathan Eagle, Alex (Sandy) Pentland, and David Lazer. Inferring friendship network structure by using mobile phone data. *PNAS*, 2009.

[70] M. Lichman. UCI machine learning repository, 2013.

[71] Barbora Micenková, Brian McWilliams, and Ira Assent. Learning outlier ensembles: The best of both worlds–supervised and unsupervised. In *ACM SIGKDD ODD*^{2} *Workshop*, 2014.

[72] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. LOF: identifying density-based local outliers. In *SIGMOD*, pages 93–104, 2000.

[73] Spiros Papadimitriou, Hiroyuki Kitagawa, Philip B Gibbons, and Christos Faloutsos. LOCI: Fast outlier detection using the local correlation integral. In *ICDE*, 2003.

[74] Matthew Eric Otey, Amol Ghoting, and Srinivasan Parthasarathy. Fast distributed outlier detection in mixed-attribute data sets. *DMKD*, 12(2-3):203–228, 2006.

[75] Arthur Zimek, Matthew Gaudet, Ricardo JGB Campello, and Jörg Sander. Subsampling for efficient and effective unsupervised outlier detection ensembles. In *KDD*, 2013.

[76] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of online learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[77] Jing Gao and Pang-Ning Tan. Converting output scores from outlier detection algorithms into probability estimates. In *ICDM*, pages 212–221, 2006.

[78] A Platanios, Avrim Blum, and Tom M Mitchell. Estimating accuracy from unlabeled data. In *In Proceedings of UAI*, 2014.

[79] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *ICDM*, pages 413–422. IEEE, 2008.

[80] Leo Breiman. Using adaptive bagging to debias regressions. Technical report, Statistics Dept. UCB, 1999.

[81] Yanmin Sun, Mohamed S Kamel, Andrew KC Wong, and Yang Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378, 2007.

[82] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[83] Charu C Aggarwal. Outlier ensembles: position paper. *ACM SIGKDD Explorations Newsletter*, 14(2):49–58, 2013.

[84] G. Grimmett and D. Stirzaker. *Probability and Random Proc.* 2001.

[85] Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. Interpreting and unifying outlier scores. In *SDM*, 2011.

[86] Jon Louis Bentley. Multidimensional binary searchtrees used for associative searching. In *Communications of the ACM 18*, 1975.

[87] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388. ACM, 2002.

[88] T.R. Bandaragoda. Isolation based anomaly detection: A re-examination, 2015.

[89] T. R. Bandaragoda, Kai Ming Ting, D. Albrecht, F. T. Liu, and J.R. Wells. Efficient anomaly detection by isolation using nearest neighbor ensemble. In *ICDM Workshop*, 2014.

[90] Fabian Keller, Emmanuel Müller, and Klemens Böhm. HiCS: high contrast subspaces for density-based outlier ranking. In *ICDE*, pages 1037–1048, 2012.

[91] KM Ting, SC Tan, and FT Liu. Mass: A new ranking measure for anomaly detection. *Gippsland School of Information Technology, Monash University*, 2009.

[92] Jillian D'onfro. *A Whopping 20% Of Yelp Reviews Are Fake*, 2013. http://www.businessinsider.com/20-percent-of-yelp-reviews-fake-2013-9.

[93] Michael Luca and Georgios Zervas. Fake it till you make it: Reputation, competition, and yelp review fraud. Working Papers 14-006, Harvard Business School, 2013.

[94] David Streitfeld. *Best Book Reviews Money Can Buy*, 2012. www.nytimes.com/2012/08/26/business/book-reviewers-for-hire-meet-a-demand-for-online-raves.html.

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

[96] Song Feng, Ritwik Banerjee, and Yejin Choi. Syntactic stylometry for deception detection. In *ACL*, 2012.

[97] Myle Ott, Claire Cardie, and Jeffrey T. Hancock. Estimating the prevalence of deception in online review communities. In *WWW*, pages 201–210, 2012.

[98] Song Feng, Longfei Xing, Anupam Gogar, and Yejin Choi. Distributional footprints of deceptive product reviews. In *ICWSM*, 2012.

[99] Nitin Jindal, Liu Bing, and Ee-Peng Lim. Finding unusual review patterns using unexpected rules. In *CIKM*, 2010.

[100] Fangtao Li, Minlie Huang, Yi Yang, and Xiaoyan Zhu. Learning to identify review spam. In *IJCAI*, 2011.

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

[102] Sihong Xie, Guan Wang, Shuyang Lin, and Philip S. Yu. Review spam detection via temporal pattern discovery. In *KDD*, pages 823–831, 2012.

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

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

[105] Huayi Li, Zhiyuan Chen, Bing Liu, Xiaokai Wei, and Jidong Shao. Spotting fake reviews via collective positive-unlabeled learning. In *ICDM*, 2014.

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

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

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

[109] Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie S. Glance. What yelp fake review filter might be doing? In *ICWSM*, 2013.

[110] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[111] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Understanding BP and its generalizations. In *Explor. AI in New Millen.* 2003.

[112] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[113] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. Catchsync: catching synchronized behavior in large directed graphs. In *KDD*, pages 941–950, 2014.

[114] Ross Kindermann and J. L. Snell. *Markov Random Fields and Their Applications*. 1980.

[115] Peter V. Marsden. Homogeneity in confiding relations. *Social Networks*, 10(1):57–76, March 1988.

[116] Charles Elkan. The foundations of cost-sensitive learning. In Bernhard Nebel, editor, *IJCAI*, pages 973–978. Morgan Kaufmann, 2001.

[117] Nguyen Thai-Nghe, Zeno Gantner, and Lars Schmidt-Thieme. Cost-sensitive learning methods for imbalanced data. In *IJCNN*, pages 1–8. IEEE, 2010.

[118] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)*, 16:321–357, 2002.

[119] Karen Weise. *A Lie Detector Test for Online Reviewers*, 2011. http://www.bloomberg.com/bw/magazine/a-lie-detector-test-for-online-reviewers-09292011.html.

[120] Nitin Jindal and Bing Liu. Opinion spam and analysis. In *WSDM*, pages 219–230, 2008.

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

[122] Sihong Xie, Guan Wang, Shuyang Lin, and Philip S. Yu. Review spam detection via temporal pattern discovery. In *KDD*, 2012.

[123] Junting Ye and Leman Akoglu. Discovering opinion spammer groups by network footprints. In *ECML/PKDD*, pages 267–282, 2015.

[124] Charu C. Aggarwal, Xiangnan Kong, Quanquan Gu, Jiawei Han, and P. S. Yu. Active learning: A survey, 2014.

[125] Burr Settles. Active learning. In *Synthesis Lectures on Artificial Intelligence and Machine Learning*, pages 1–114, 2012.

[126] M. Bilgic and L. Getoor. Effective label acquisition for collective classification. In *ACM KDD*, pages 43–51, 2008.

[127] M.J. Rattigan, M. Maier, and D. Jensen. Exploiting network structure for active inference in collective classification. In *ICDM*, pages 429–434, 2007.

[128] D. Lewis and W. Gale. A sequential algorithm for training text classifiers. In *ACM SIGIR*, pages 3–12, 1994.

[129] A. MacCallum and K. Nigam. Employing EM in pool-based active learning for text classification. In *ICML*, 1998.

[130] M. Bilgic, L. Mihalkova, and L. Getoor. Active learning for networked data. In *ICML*, pages 79–86, 2010.

[131] S. A. Macskassy. Using graph-based metrics with empirical risk minimization to speed up active learning on networked data. In *ICML*, pages 385–392, 2011.

[132] D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In *ICML*, pages 148–156, 1994.

[133] Manali Sharma and Mustafa Bilgic. Most-surely vs. least-surely uncertain. In *ICDM*, 2013.

[134] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L Reyes-Ortiz. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *International workshop on ambient assisted living*, pages 216–223. Springer, 2012.

[135] Ninh Pham and Rasmus Pagh. A near-linear time approximation algorithm for angle-based outlier detection in high-dimensional data. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 877–885. ACM, 2012.

[136] David M Rocke and David L Woodruff. Identification of outliers in multivariate data. *Journal of the American Statistical Association*, 91(435):1047–1061, 1996.