TEXT MINING WITH WORD EMBEDDING FOR OUTLIER AND SENTIMENT ANALYSIS

BY

HONGLEI ZHUANG

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

Doctoral Committee:

Professor Jiawei Han, Chair
Professor ChengXiang Zhai
Assistant Professor Jian Peng
Dr. Joel Young, LinkedIn

ABSTRACT

The technology today makes it unprecedentedly easy to collect and store massive text data in various domains such as online social networks, medical records and news reports. In contrast to the gigantic volume of text data, human capabilities to read and process text data is limited. Hence, there is an emerging demand for automatic text mining tools to analyze massive text data.

Word embedding is an emerging text analysis technique that leverages the fine-grained statistics of context information to map each word to a vector in the embedding space which reflects the semantic proximity between words. Embedding techniques not only enrich the statistical signals to utilize in downstream text mining applications, but also provide the possibility to characterize and represent higher-level objects in the embedding space, such as sentences, documents or topics.

This study integrates word embedding techniques into a series of text mining approaches and models. The general idea is to take a text object such as a document or a sentence as a bag of embedding vectors and characterize their distributions in the embedding space. Specifically, this study focuses on two tasks: *outlier analysis* and *weakly-supervised sentiment analysis*.

Outlier analysis aims to identify documents that topically deviate from the majority of a given corpus. We develop an unsupervised generative model to identify frequent and representative semantic regions in the word embedding space to represent the given corpus. Then we propose a novel outlierness measure to identify outlier documents. We also study the cost-sensitive scenario of outlier analysis.

Sentiment analysis typically identifies the subjective opinion (*e.g.*, positive vs. negative) in a piece of text. Despite being extensively studied as a supervised learning task, we tackle the problem in a weakly-supervised fashion, where users only provide a small set of seed words as guidance. We study to identify aspects and corresponding sentiments at both document and sentence level.

To my family.

ACKNOWLEDGMENTS

I would like to extend my deepest gratitude to my advisor Professor Jiawei Han for his relentless support and guidance on my research. This dissertation would not be possible without his inspiring ideas, constructive criticism and invaluable insights.

I also wish to thank other members of my Ph.D. committee, Prof. ChengXiang Zhai, Prof. Jian Peng and Dr. Joel Young for their ingenious suggestions on my dissertation.

I am also grateful to my collaborators and many other students and researchers who contribute to our research in many different ways. I would like to thank Dr. Paul Ogilvie, Dr. Joel Young from LinkedIn, Dr. Chi Wang from Microsoft Research, Dr. Laura Chiticariu, Dr. Yunyao Li, Dr. Chenguang Wang from IBM Research and Dr. Hasan Cam, Dr. Timothy Hanratty, Dr. Lance Kaplan from Army Research Lab for their guidance when I was working with them. I also want to thank Prof. Xifeng Yan, Prof. Aditya Parameswaran, Prof. Dan Roth and Prof. Jie Tang for their contribution in our collaboration. I am extremely lucky to work with many talented peer students and researchers. Thanks to DAIS members and alumni Chi Wang, Fangbo Tao, Chao Zhang, Yu Shi, Jonathan Kuck, Fang Guo, George Brova, Yanglei Song, Liyuan Liu, Ahmed El-Kishky, Yucheng Chen, Yihan Gao, Huan Gui, Quanquan Gu, Prof. Jing Gao, Shi Zhi, Jialu Liu, Jingjing Wang, Dr. Meng Jiang, Meng Qu, Stephen Macke, Doris Xin, Jingbo Shang, Jiaming Shen, Dr. Quan Yuan, Dr. Qi Li, Carl Yang, Xiaotao Gu, Yu Meng, Xiang Ren, Marina Danilevsky, Brandon Norick, Xuan Wang, Yuning Mao, Jinfeng Xiao, Qi Zhu, Jiaxin Huang, Xinyang Zhang, Yu Zhang, Sha Li, Wanzheng Zhu, Po-Wei Chan, Tobias Kin Hou Lei, Hongkun Yu, Min Li, Qi Wang as well as Haoruo Peng from Cognitive Computation Group at University of Illinois and Yifan Wang from Tsinghua University for their assistance in my research.

Finally, I would like to thank my parents for their understanding and encouragement during my pursuit of the Ph.D. degree. I am also deeply indebted to my beloved wife, Siying, for her tremendous love, support and sacrifice. She was the one who always stayed strong and got my feet back to the ground during the hard times. I truly thank her for staying with me. There are no words to convey how much I love her.

My work was sponsored in part by U.S. Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), DARPA under Agreement No. W911NF-17-C-0099, National Science Foundation IIS 16-18481, IIS 17-04532, and IIS-17-41317, DTRA HDTRA11810026, and grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (`www.bd2k.nih.gov`).

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION 1
1.1 Background 1
1.2 Research Problems and Challenges 2

CHAPTER 2 LITERATURE REVIEW 6

CHAPTER 3 UNSUPERVISED TEXT MINING FOR OUTLIER ANALYSIS . . . 9
3.1 Overview 9
3.2 Preliminaries 9
3.3 Mining Outlier Documents 11
3.4 Experiment Setup 17
3.5 Results 19
3.6 Summary 23

CHAPTER 4 COST-SENSITIVE OUTLIER DETECTION 24
4.1 Overview 24
4.2 Problem Definition 24
4.3 Algorithms 26
4.4 Experimental Results 31
4.5 Theoretical Analysis 35
4.6 Summary 48

CHAPTER 5 WEAKLY-SUPERVISED TEXT MINING FOR ASPECT-BASED SENTIMENT ANALYSIS 49
5.1 Overview 49
5.2 Preliminaries 51
5.3 Lexicon Expansion 53
5.4 Aspect-Based Sentiment Classification 57
5.5 Experiments 60
5.6 Summary 65

CHAPTER 6 JOINT WEAKLY-SUPERVISED TEXT MINING FOR ASPECT AND SENTIMENT ANALYSIS 66
6.1 Overview 66
6.2 Preliminaries 67
6.3 Aspect Sentiment Autoencoder 68
6.4 Joint Aspect Sentiment Autoencoder 73
6.5 Experimental Results 76
6.6 Summary 82

CHAPTER 7 CONCLUSION AND FUTURE WORK 84
7.1 Conclusion 84
7.2 Future Work 85

REFERENCES 87

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND

The volume of text data in numerous domains is explosively growing thanks to the recent progress in technology. The emergence of multiple novel infrastructures and platforms such as cloud service and online social networks substantially facilitates the creation and storage of text data for users. Regarding the gigantic size of text data available, it is intractable to rely on human effort to digest the entire corpus and extract knowledge of interest. Instead, it becomes increasingly important to develop automatic tools to mine essential insights from massive text data. There are a series of studies on text mining techniques such as document clustering [4], summarization [56] and topic modeling [13, 31].

Traditional text mining techniques take each word as an isolated unit and calculate their frequencies and correlations of occurrences to identify valuable patterns. For example, classical topic models [13, 31] represent each input document by the bag-of-word representation and derive topics represented as a multinomial distribution over all the possible words. However, the semantic proximity between words is largely neglected. For example, the word "good" and the word "great" carry very similar semantic meaning, but are treated as two isolated units while calculating their frequencies.

Recent progress in word embedding [61, 67, 66] provides a promising tool to address this issue, where each word can be mapped to a vector in a continuous embedding space. The mapping is optimized by ensuring a specific similarity measure (*e.g.*, cosine similarity) can properly reflect the semantic proximity between words. For example, the embedding vectors of "good" and "great" should be close to each other in the embedding space. The word embedding technique actually encodes the statistics of context information for each word into its embedding vector to serve higher-level text mining techniques.

We integrates word embedding to develop novel text mining techniques. Instead of characterizing a text object such as a sentence, a document or a corpus by discrete word distributions, we propose to leverage continuous distributions in the embedding space as the representation. This philosophy is capable of merging the frequencies of words with similar meanings and smoothing the sparsity of rare words.

The modeling principle benefits various downstream applications. It improves the unsupervised text modeling and therefore benefits tasks like outlier detection. Moreover, it enables some traditionally supervised tasks to be performed with few supervision, such as aspect-based sentiment analysis.

1.2 RESEARCH PROBLEMS AND CHALLENGES

We study to leverage word embedding in different text mining scenarios. First, we utilize word embedding to strengthen unsupervised text mining tasks. We apply word embedding in a novel unsupervised text mining task: identifying outlier documents. Then, we explore to attack tasks which traditionally require a lot of labeled data with very few user guidance by taking advantage of the word embedding space. Specifically, we study to perform sentiment analysis with only few keywords provided by users as guidance.

In the following subsections, we describe the two tasks we study as well as their major challenges.

1.2.1 Topically deviating document outliers

Outlier documents, which substantially deviate from the semantic focuses of the given corpus, can provide valuable insights or imply potential errors. For example, an outlier health record from records of the same disease could indicate a new variation of the disease if it has an abnormal symptom description, or a medical error if it has an abnormal treatment description. A previous study [26] uses structured data in health records to show the importance of this application, and points out that further improvement should be achieved by leveraging text data.

Existing work has studied a related albeit different task, novel document detection [40, 39, 95, 93], where one aims to identify from a document stream if a newly arriving document is novel or redundant. In other words, this task assumes all the previous documents are known to be "normal", and only checks if a new document is novel. In our task, no document is known to be normal, and there could be multiple outliers in the corpus. Outlier detection [16, 30] is a popular topic in data mining but few focus on text data. A study [25] identifies anomalous text segments in a document, but mainly based on writing styles. We focus on studying topically deviating documents.

Figure 1.1 briefly illustrates a real world example. Suppose the analyst is reviewing a set of health records. The left figure shows the input, which is a given set of documents majorly focusing on cardiovascular diseases while some documents are more relevant to urology. Such records are worth more attention as they might indicate abnormal complications or medical errors. The right figure presents the output where documents are ranked based on there outlierness. By examining the ranked list of outlier documents, one can identify valuable insights from the health records without exhaustively reading all the health records.

The problem of detecting outlier documents has its unique challenges. First, different words

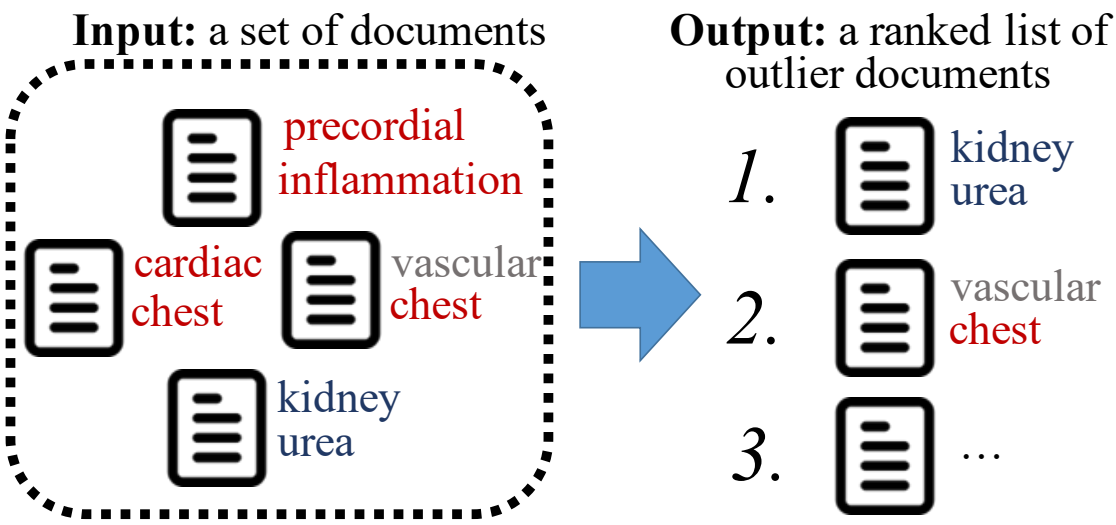


Figure 1.1: An example of mining outlier documents in a given corpus. The left figure shows data input: a set of health records on cardiovascular diseases. The right figure shows a set of outlier records relevant to urology.

or phrases may be used to indicate the same semantic meaning, which introduces lexical sparsity. Second, finding proper words or phrases to characterize the corpus is non-trivial. Some frequent words or phrases can be too general or too vague. Third, a document can carry extremely rich and noisy signals, most of which are not helpful to determine whether it is an outlier.

1.2.2 Sentiment analysis with minimal guidance

Sentiment analysis, which aims to identify the subjective opinion (*e.g.* positive vs. negative) of a given piece of text, is an essential task of text understanding with a broad range of applications, such as recommendations [12, 18], stock prediction [76, 65] *etc.* Aspect-based sentiment analysis [74] takes a further step to identify the target aspect of sentiment in a given sentence. For example, sentences from restaurant reviews "*The food is good*" and "*The servers are friendly*" both convey a positive sentiment, while the first sentence should be identified as a comment on the *Food* aspect and the second on the *Service* aspect.

The majority of studies on aspect-based sentiment analysis adopt a supervised framework [47, 57, 99, 34, 28, 51, 80, 86, 87] where a significant number of labeled sentences are required to train the model. Nevertheless, such labels are extremely expensive and difficult to obtain, especially for a new domain or a new language. Another thread of studies [32, 63, 38, 55, 70, 71, 52, 7] focus on weak supervision or distant supervision to perform

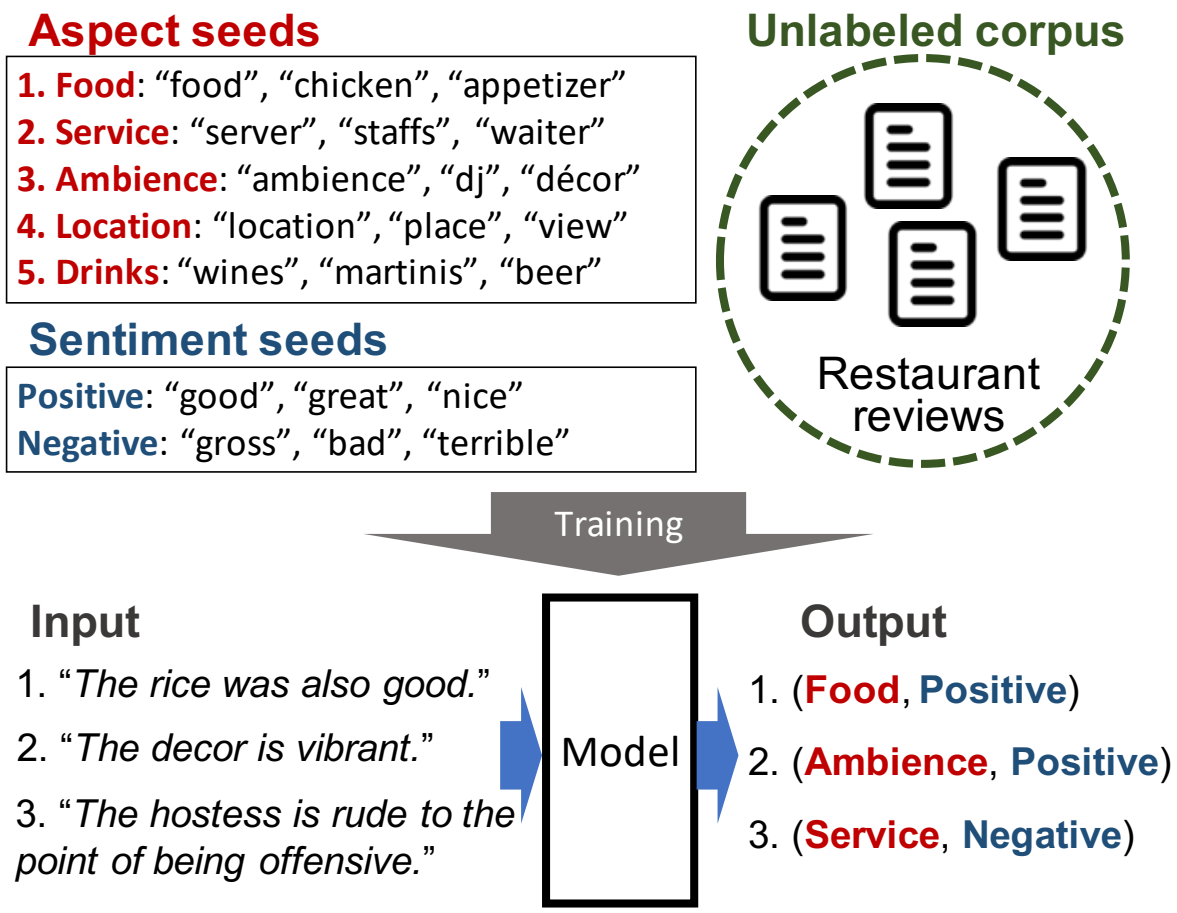


Figure 1.2: An example of aspect-sentiment analysis with minimal guidance. Users only provide small sets of aspect and sentiment words. The algorithm outputs the identified aspect and sentiment class for each sentence.

aspect-based sentiment analysis. However, some of them either rely on external language resource such as thesaurus information [32, 63, 38, 55, 70, 7] or syntactic structures generated by well-trained NLP tools [71, 52]. In reality, such information is not always available or accurate in new domains or low-resource languages.

We aim to develop an aspect-based sentiment analysis tool merely from the massive unlabeled text data without heavy external language resource or syntactic structures other than a few seed words from users. More specifically, users only need to provide a small set of seed words for each aspect class and a set of seed words for each sentiment class. The objective is to build a model to identify the aspect class and sentiment class for any sentence expressing an opinion on a target aspect class. Comparing to most previous studies on weakly supervised aspect-based sentiment analysis, our setting requires significantly less effort or resource from users.

Figure 1.2 presents an example, where a user needs to conduct aspect-based sentiment analysis on a set of restaurant reviews. The aspect classes are known for the restaurant domain, which are *Food*, *Service*, *Ambience*, *Location* and *Drinks*. The user only needs to provide a small set of seed words for each aspect class, a small set of seed words for each sentiment class, and the unlabeled corpus. For example, the user can specify {"food", "chicken", "appetizer"} for *Food* aspect, {"server", "staffs", "waiter"} for *Service* aspect *etc.*, while {"good", "great", "nice} and {"gross", "bad", "terrible} for *Positive* and *Negative* sentiment class respectively. Based on the user-provided seed words and the corpus, our aim is to train a model to identify the aspect and sentiment class for any given sentence. For example, given a sentence "The rice was also good", the model should output its aspect class as *Food* and sentiment class as *Positive*.

There are several research challenges in this problem. The first is how to model the aspect and sentiment perspectives of sentences in the unlabeled corpus. The second is how to utilize the user-provided seed words to guide the aforementioned modeling process, such that the learned model can be well aligned with user intention.

CHAPTER 2: LITERATURE REVIEW

Text embedding and applications. Word embedding aims to learn a distributed representation vector for each word/phrase as a compact representation. A series of work is done to derive word embedding by leveraging neural networks [21, 62, 83]. The seminal work by Mikolov *et al.* [61] propose to learn vector representations of words by fitting a skip-gram model to a corpus. Specifically, they aim to derive a vector representation for each word such that the surrounding words of its occurrence in a sentence or a document can be best predicted. They also propose several tricks in training such as negative sampling. They show that the learned vector presents interesting properties like additive composition, where simple arithmetic operation produces meaningful results.

There are a series of studies addressing limitations of the proposed word embedding technique. Pennington *et al.* [67] further incorporate global information in addition to the local context. Neelakantan *et al.* [64] learns multiple vector representations for words with multple senses. Bojanowski *et al.* [14] use subword information to enrich the word embedding, while Wieting *et al.* [89] similarly use character-level information to improve the learned embedding. Nickel *et al.* [66] derive word embedding in a hyperbolic space to reflect the hierarchical structure of words. In addition, some studies also focus on deriving contextualized word embedding, which does not assign a fixed embedding vector for each word in a lexicon, but outputs the vector representation for each word occurrence based on its context. Peters *et al.* [68] propose ELMo, a word representation based on the internal states of a pretrained bi-directional language model.

Le *et al.* [46] also propose to derive embedding vector representations for paragraphs and documents by concatenating embedding of words in the context window and a document vector to be learned. Kiros *et al.* [42] propose an encoder-decoder model to derive vector representation for sentences by reconstructing surrounding sentences. Recent studies such as ELMo [68] and BERT [23] on unsupervised language model can also provide vector representation for sentences.

Word embedding greatly boosts the performance of multiple downstream text-related tasks from syntactic parsing [77, 17] and entity typing [91] to machine reading comprehension [54] and machine translation [37, 10]. Massive labeled data is still required for these tasks, but word embedding improves the performance comparing to traditional methods with similar supervision. On the other hand, word embedding also benefits some unsupervised or weakly-supervised tasks including topic modeling [22, 11], set expansion [72, 75], taxonomy construction [92] *etc.*

Outlier document. Outlier detection is an essential task studied in data mining [16, 30]. There have been extensive studies on outlier detection in data mining community [15, 48, 43, 44, 35, 78]. Few studies explicitly focus on detection of document outliers. Wang *et al.* [88] study outlier document detection leveraging domain ontologies but mainly focus on short text; Aouf *et al.* [9] utilize the TF-IDF representation and propose a random projection method to approximate the outliers. Guthrie [25] studies identifying anomalous text segments, but basically based on writing styles, while the problem we discuss focuses more on topical outliers. Aggarwal *et al.* [2] also provides a brief review of relevant studies on outlier analysis in text.

The most relevant stream of research focuses on novel document detection [40, 39, 95, 93], where one aims to identify from a document stream if a newly arriving document is novel or redundant. However, novel document detection is a fundamentally differed problem. Novel document detection only judges if the arriving document is novel as compared to older documents, and time factor is usually used to define the norm, whereas in outlier document detection, any document in the given corpus could be an outlier.

Sampling for outlier detection. Sampling techniques have been used for detecting outlier data points from observed data points for different purposes. In [1], outlier detection is reduced to a classification problem and an active learning algorithm is proposed to selectively sample data points for training the outlier detector. In [79, 90], a subset of data points is uniformly sampled to accelerate the outlier detector. [45] propose a biased sampling strategy. [98, 49] use a subsampling technique to introduce diversity in order to apply ensemble methods for better outlier detection performance.

There are also studies on outlier detection when uncertainty of data points is considered [3, 33]. However, these algorithms do not attempt to actively request more information about data points to reduce the uncertainty.

Weakly-supervised aspect-based sentiment analysis. A series of studies on aspect-based sentiment analysis utilize aspect and/or sentiment lexicons. Some methods directly leverage an existing lexicon from an external source, such as [63] which uses a sentiment lexicon. Other methods develop algorithms to automatically build the aspect and/or sentiment lexicons.

Frequency-based methods construct the lexicons by counting the frequencies of each word in a given corpus and developing reasonable measures to distinguish aspect/sentiment words from others. Hu *et al.* [32] use a frequency-based method to identify frequent nouns to build the aspect lexicon. Then, they extract adjectives adjacent to the identified aspect words to build the sentiment lexicon. This method relies on part-of-speech (POS) tags of

words in sentences. Popescu *et al.* [70] develop another frequency-based method and achieve improvement from [32], but they rely on more external resource such as the web statistics data. Scaffidi *et al.* [73] also propose a frequency-based method developed from a statistical test to construct the aspect lexicon but still requires the POS tags.

Syntax-based methods further leverage the syntactic structure of each word occurrence in the lexicon construction process. Qiu *et al.* [71] choose to build the lexicons from some seed aspect or sentiment words by syntactic rules. They first obtain the dependency structure of each sentence in the corpus. Then they start from the given seed words to add new words that follow certain expert-given syntactic rules. However, the quality of their method heavily rely on the accuracy of the dependency parser, which can be low on a new domain without training data. Moreover, the method requires users to specify syntactic rules, while users are not necessarily familiar with linguistic knowledge. Although there are some follow-up studies to improve this algorithm, they still suffer from these drawbacks [52, 53]. Zhang *et al.* [94] also utilize similar ideas, with a different set of rules, as well as a HITS-based algorithm to rank the aspects. Zhao *et al.* [96] study to generalize some syntactic structures for better coverage on aspect extraction.

Generative topic models are also frequently adopted to model the aspect and sentiment data. A series of work by Wang *et al.* [84, 85] propose generative models to predict rating on each aspect. Nevertheless, their work rely on additional overall rating data for each review. Titov and McDonald [82] also propose a multi-aspect sentiment model to jointly characterize aspect occurrences and sentiment ratings of users. Similarly, they rely on the rating as supervision. Mei *et al.* [60] study a topic model for the general sentiment as well as the dynamics of topics. They focus more on corpus level summarization, while our objective is aspect-based sentiment analysis.

Jo and Oh [36] propose a sentence-level generative topic model. They use a mixture of joint aspect-sentiment topics to model each sentence with a seed set of sentiment words. Their model is capable of performing sentence-level aspect-based sentiment analysis from similar input as ours. However, they only leverage the co-occurrence signals between words for semantic proximity, without enjoying the benefit of recent progress on word embedding.

Recently, there are also some neural-network-based methods focusing on unsupervised or weakly supervised aspect-based sentiment analysis. He *et al.* [27] propose an unsupervised neural model to extract aspects by an attention mechanism. However, their method requires a manual step to map the learned aspects to the desired aspect classes. They focus more on aspect extraction, and do not study sentiment analysis. Angelidis *et al.* [7, 6] also explore to use seed aspect words as user guidance and perform aspect extraction and sentiment prediction, but they again require overall sentiment rating as supervision.

CHAPTER 3: UNSUPERVISED TEXT MINING FOR OUTLIER ANALYSIS

3.1 OVERVIEW

In this chapter, we describe an unsupervised modeling technique to characterize text data in the embedding space with word embedding by studying the application of identifying topically deviating outlier documents from a given corpus.

We tackle the problem of mining outlier documents in the following steps. We leverage word embedding [61] to capture the semantic proximities between words and/or phrases, in order to solve the sparsity issue. Then we propose a generative model to identify semantic regions in the embedded space frequently mentioned by documents in the corpus. The model represents each semantic region with a von Mises-Fisher distribution. We also learn a concentration parameter for each region with our model, and develop a selection method to identify semantically specific regions which can better represent the corpus, and filter regions with largely uninformative words.

As the final step, we design a robust outlierness measure emphasizing only the words or phrases in a document relatively close to the semantic focuses identified, and eliminating the noises and redundant information.

3.2 PRELIMINARIES

In this section, we formalize the problem and then briefly describe the preprocessing step.

3.2.1 Notations

The notations used in this study are introduced here. A *document* is represented as a sequence $d_i = (w_{i1}, w_{i2}, \cdots, w_{in_i})$, where each $w_{ij} \in \mathcal{V}$ represents a word or phrase from a given vocabulary $\mathcal{V}$ and n_i denotes the length of the d_i. We refer to a set of documents as a *corpus*, represented as $D = \{d_i\}_{i=1}^{|D|}$.

Notice that w_{ij} may refer to a unigram word or a multi-gram phrase. Although it is non-trivial to appropriately segment a document into a mixed sequence of words and phrases, it is not the focus of our paper. A recently developed phrase mining technique [50] is used to extract quality phrases and to segment the documents.

Word embedding provides vectorized representations of words and phrases to capture their semantic proximity. We assume there is an effective word embedding technique (*e.g.* [61]),

$f : \mathcal{V} \mapsto \mathbb{R}^{\nu}$, where f is the transforming function that takes a word or a phrase as input and projects it into a ν-dimensional vector as its distributed representation. The semantic proximity between two words or phrases w and w' can be preserved by the cosine similarity between their embedded vectors:

$$\mathrm{sim}\big(f(w), f(w')\big) = \frac{f(w) \cdot f(w')}{\|f(w)\| \times \|f(w')\|} \tag{3.1}$$

This work studies how to effectively rank documents in a corpus based on how much they deviate from the semantic focuses of the corpus. More formally,

Problem 3.1. *Given a set of documents D, our objective is to design an outlierness measure $\Omega : D \mapsto \mathbb{R}$, such that documents with larger outlierness $\Omega(d)$ is more semantically deviating from the majority of the corpus D.*

3.2.2 Preprocessing

We perform several steps of preprocessing to derive the input representation of each document in a given corpus.

Phrase mining. SegPhrase, a recently developed phrase-mining method [50], is utilized to automatically identify quality phrases in a corpus. After being trained in one corpus, SegPhrase is also capable of segmenting unseen documents into chunks of phrases with mixed lengths. We train SegPhrase on an external corpus D_e to obtain the list of quality phrases. Then for each corpus D given for outlier detection, we employ the trained SegPhrase to chunk each document into a sequence of words and quality phrases.

Word embedding. We adopt word embedding as a preprocessing step to capture the semantic proximity between words/phrases. Instead of using the raw text, similar to [50], we use the sequence derived from SegPhrase as input to the word embedding algorithm. In particular, *word2vec* [61] is utilized in our experiments, but can be seamlessly replaced by any other embedding results.

We run the embedding algorithm based on the external corpus D_e, the same corpus used in phrase mining. As D_e is sufficiently large, there are only few words or phrases in D which never appear in D_e, and they are simply discarded in the experiments.

Stop words removal. We remove stop words, as well as the words or phrases ranked high within a certain quantile in terms of document frequency[1] (DF) in the external corpus D_e.

[1]Document frequency of a word (or phrase) is defined as number of documents where this word or phrase appears.

Such words or phrases usually carry background noise, and obstruct outlier detection.

3.3 MINING OUTLIER DOCUMENTS

Our framework consists of the following steps. First, we leverage a generative model to identify semantic "regions" in the word embedding space frequently mentioned by documents in the given corpus. Second, we develop a selection method to further remove semantics regions that are too general to properly characterize the given corpus and only keep regions both frequent and semantically specific, denoted as "semantic focuses". Finally, we calculate the outlierness measure for each document based on the mined semantic focuses. We design a robust outlierness measure which is less sensitive to noisy words or phrases in documents.

3.3.1 Embedded von Mises-Fisher Allocation

We start with a generative model to identify the frequent semantic regions in the word embedding space.

Since we use cosine similarity to capture the semantic proximities between two words or phrases, the magnitude of the embedding vector of each word can be omitted in this part. We use $\mathbf{x}_{ij} = f(w_{ij})/\|f(w_{ij})\|$ to represent the unit vector with the same direction as the embedded vector of w_{ij}, and use $\mathbf{X}$ to represent the collection of all $\mathbf{x}_{ij}$ where $1 \leq i \leq |D|$ and $1 \leq j \leq n_i$.

In order to characterize a semantic region in the embedded space, we introduce von Mises-Fisher (vMF) distribution. The von Mises-Fisher (vMF) distribution is prevalently adopted in directional statistics, which studies the distribution of normalized vectors on a spherical space. The probability density function of the vMF distribution is explicitly instantiated by the cosine similarity. It is an ideal distribution for our task because we use cosine similarity to measure the semantic proximity. Moreover, as we will see later, it empowers us to characterize how specific each semantic region is, which is helpful in further identification of semantic focuses for outlier detection.

We first introduce the formalization of the von Mises-Fisher distribution.

Von Mises-Fisher (vMF) distribution. A ν-dimensional unit random vector $\mathbf{x}$ (*i.e.* $\mathbf{x} \in \mathbb{R}^\nu$ and $\|\mathbf{x}\| = 1$) follows a von Mises-Fisher distribution $\text{vMF}(\cdot|\boldsymbol{\mu}, \kappa)$ if the probability density function follows:

$$p(\mathbf{x}) = C_\nu(\kappa) \exp\big(\kappa \boldsymbol{\mu}^\top \mathbf{x}\big) \tag{3.2}$$

where $C_\nu(\kappa) = \kappa^{\nu/2-1}\big/(2\pi)^{\nu/2} I_{\nu/2-1}(\kappa)$; and $I_{\nu/2-1}(\cdot)$ is the modified Bessel function of the first kind; $(\nu/2-1)$ is the order.

The two parameters in the vMF distribution are the mean direction $\boldsymbol{\mu}$ and the concentration parameter κ respectively, where $\boldsymbol{\mu} \in \mathbb{R}^\nu$, $\|\boldsymbol{\mu}\| = 1$ and $\kappa > 0$. The distribution concentrated around the mean direction $\boldsymbol{\mu}$, and is more concentrated if the concentration parameter κ is larger.

Embedded von Mises-Fisher allocation. We propose a generative model by regarding each document as a bag of normalized embedded vectors, analogous to the bag-of-word representation of documents utilized in typical topic model (*e.g.*, LDA [13]). The major difference is that the data to be generated is now a bag-of-vector representation for each document, which should be generated from a mixed vMF distribution instead of a mixed multinomial distribution.

A formalized description of the model is summarized as follows:

$$
\begin{aligned}
\boldsymbol{\mu}_t &\sim \text{vMF}(\cdot|\boldsymbol{\mu}_0, C_0), && t = 1, 2, \cdots, T \\
\kappa_t &\sim \text{logNormal}(\cdot|m_0, \sigma_0^2), && t = 1, 2, \cdots, T \\
\boldsymbol{\pi}_i &\sim \text{Dirichlet}(\cdot|\boldsymbol{\alpha}), && i = 1, 2, \cdots, |D| \\
z_{ij} &\sim \text{Categorical}(\cdot|\pi_i), && j = 1, 2, \cdots, |d_i| \\
\mathbf{x}_{ij} &\sim \text{vMF}(\cdot|\boldsymbol{\mu}_{z_{ij}}, \kappa_{z_{ij}}), && j = 1, 2, \cdots, |d_i|
\end{aligned}
$$

where $T > 0$ is an integer indicating the number of semantic regions, namely the number of vMF distributions in our mixture model.

We regularize the vMF parameters by the following prior distributions. We assume the mean direction $\boldsymbol{\mu}_t$ of each vMF distribution is generated from a prior vMF distribution $\text{vMF}(\cdot|\boldsymbol{\mu}_0, C_0)$, while the concentration parameter κ_t is generated from a log-normal prior $\text{logNormal}(\cdot|m_0, \sigma_0^2)$. A similar design is also adopted in [24].

Parameter inference. We infer the parameters by Gibbs sampling. Because both the von Mises-Fisher distribution and the Dirichlet distribution have conjugate priors, we can integrate out parameters $\boldsymbol{\mu}_t$ and $\boldsymbol{\pi}_i$ and develop a collapsed Gibbs sampler of z_{ij}:

$$
\begin{aligned}
&P(z_{ij} = t|\mathbf{Z}^{-ij}, \mathbf{X}, \boldsymbol{\kappa}; \boldsymbol{\alpha}, m_0, \sigma_0^2, \boldsymbol{\mu}_0, C_0) \\
\propto &\frac{\big(n_{it}^{-ij} + 1 + \boldsymbol{\alpha}^{(t)}\big) C_\nu(\kappa_t) C_\nu\big(\big\|C_0\boldsymbol{\mu}_0 + \kappa_t \mathbf{x}_{\cdot t}^{-ij}\big\|\big)}{C_\nu\Big(\Big\|C_0\boldsymbol{\mu}_0 + \kappa_t\big(\mathbf{x}_{\cdot t}^{-ij} + \mathbf{x}_{ij}\big)\Big\|\Big)}
\end{aligned}
\tag{3.3}
$$

where $n_{it}^{-ij} = \sum_{j'}^{|d_i|} \delta(z_{ij'} = t) - \delta(z_{ij} = t)$ is the number of words in the i-th document being assigned to the t-th von Mises-Fisher distribution without taking w_{ij} into account; $\mathbf{x}_{\cdot t}^{-ij} = \sum_{i'}^{|D|} \sum_{j'}^{|d_i|} \mathbf{x}_{i'j'} \delta(z_{i'j'} = t) - \delta(z_{ij} = t)$ is the sum of word vectors assigned to semantic region t without counting w_{ij}. Here $\delta(\cdot)$ is the indicator function.

We can also derive a collapsed Gibbs sampler for concentration parameters κ_t's:

$$\begin{aligned} & P(\kappa_t | \mathbf{Z}, \mathbf{X}, \boldsymbol{\kappa}^{-t}; \boldsymbol{\alpha}, m_0, \sigma_0^2, \boldsymbol{\mu}_0, C_0) \\ \propto & \frac{C_\nu^{n_{\cdot t}}(\kappa_t)}{C_\nu\big(\|C_0 \boldsymbol{\mu}_0 + \kappa_t \mathbf{x}_{\cdot t}\|\big)} \text{logNormal}(\kappa_t | m_0, \sigma_0^2) \end{aligned} \tag{3.4}$$

where $n_{\cdot t}$ is the number of words in semantic region t.

While sampling z_{ij} is relatively trivial, sampling κ_t is not straightforward. Similar difficulty is also mentioned in [24]. We employ a Metropolis-Hasting algorithm with another log-normal distribution centered at the current κ_t value as the proposal distribution.

After obtaining a sample from the posterior distribution of z_{ij}'s and κ_t's, we can easily obtain the MAP estimate of mean directions $\boldsymbol{\mu}_t$'s and the mixing distribution of each documents $\boldsymbol{\pi}_i$:

$$\hat{\boldsymbol{\mu}}_t = \frac{C_0 \boldsymbol{\mu}_0 + \kappa_t \mathbf{x}_{\cdot t}}{\|C_0 \boldsymbol{\mu}_0 + \kappa_t \mathbf{x}_{\cdot t}\|}, \quad \hat{\boldsymbol{\pi}}_i = \frac{n_{it} + \boldsymbol{\alpha}^{(t)}}{n_{i\cdot} + \sum_t \boldsymbol{\alpha}^{(t)}}$$

Discussions. We notice that there are some topic models [22, 11] proposed for similar data, where words are represented as embedding vectors. Our model is proposed independently for the purpose of identifying semantic focuses, which serves the task of outlier detection. Existing models may lack signals for the following outlier detection steps and hence cannot be directly plugged in. However, it is possible to adapt certain models to the outlier detection task.

3.3.2 Identifying Semantic Focuses

The semantic regions learned from the Embedded vMF Allocation model provide a set of candidates frequently mentioned by documents in the corpus. However, not all of them are semantic focuses of the corpus — some are too general to distinguish outlier and normal document.

We notice that uninformative semantic regions (*e.g.* a semantic region containing {"percent", "average", "compare", ...}) tend to have more scattered distribution over embedded vectors, possibly because of the diverse context of their usage. In contrast, corpus-specific semantic

regions are more concentrated, (*e.g.* a semantic region containing {"drugs", "antidepressant", "prescription", ...}). Modeling semantic regions by vMF distributions provides us with a parsimonious signal to characterize how concentrated a semantic region is, *i.e.* the concentration parameter κ_t. This allows us to simply filter unqualified semantic regions with too small concentration parameters and obtain high-quality semantic focuses. Let a binary variable ϕ_t $(t = 1, 2, \cdots, T)$ indicate whether the t-th vMF distribution is a semantic focus. Suppose a user specifies a threshold parameter $0 \leq \beta \leq 1$. We can determine ϕ_t by estimating the log-normal distribution that generates all κ_t's, $logNormal(\hat{m}, \hat{\sigma}^2)$, where

$$\hat{m} = \frac{1}{T}\sum_t \log(\kappa_t),\ \hat{\sigma}^2 = \frac{1}{T}\sum_t \Big(\log(\kappa_t) - \hat{m}\Big)^2$$

Set $\hat{F}_\kappa(\cdot)$ to be its cumulative distribution function. We assign $\phi_t = 1$ for semantic regions with $\kappa_t \geq F_\kappa^{-1}(\beta)$, and filter all the other semantic regions as $\phi_t = 0$.

Although parameter β needs to be set manually, our experiments suggest that the performance is not quite sensitive to its value.

3.3.3 Document Outlierness

In this subsection, we start with a straightforward definition of outlierness based on the mined semantic focuses. Then we present several refinements to improve its robustness.

Baseline outlierness measure. A straightforward intuition is to assume outlier documents averagely have fewer words or phrases drawn from semantic focuses. To estimate this, we first need to calculate the probability of each word being drawn from the semantic focuses.

$$P\big(\phi_{z_{ij}} = 1 | \mathbf{x}_{ij}, \boldsymbol{\pi}_i\big) = \frac{\sum_t \phi_t \boldsymbol{\pi}_i^{(t)} \mathrm{vMF}(\mathbf{x}_{ij}|\boldsymbol{\mu}_t, \kappa_t)}{\sum_t \boldsymbol{\pi}_i^{(t)} \mathrm{vMF}(\mathbf{x}_{ij}|\boldsymbol{\mu}_t, \kappa_t)}$$

It is then possible to estimate the expected percentage of words *not* drawn from semantic focuses in each document as the outlierness:

$$\Omega_{\mathrm{sf}}(d_i) = 1 - \frac{1}{|d_i|}\sum_{j=1}^{|d_i|} P(\phi_{z_{ij}} = 1 | \mathbf{x}_{ij}, \boldsymbol{\pi}_i) \tag{3.5}$$

However, due to the noisiness in text data, this assumption oversimplifies the characterization of outlier documents. In practice, we observe the following two issues: lexically general words/phrases, and noisy content in documents.

Penalizing lexically general words and phrases. Not all words or phrases close to semantic focuses are strong indicators of normal documents. General words (*e.g.* "science") can happen to be semantically close to a semantic focus, but are not as specific as most other words close to it (*e.g.* "medical research"). Therefore, we utilize a background corpus D_{bg} to calculate the specificity of the word. Assuming the actual mention of the word can be chosen from either the general background or a corpus-specific vocabulary, we write down the probability that a word is corpus-specific to be:

$$P\big(\lambda_{ij}|w_{ij}\big) = \frac{nd(w_{ij})/|D|}{nd(w_{ij})/|D| + nd_{bg}(w_{ij})/|D_{bg}|}$$

where $nd(w) = |\{d_i|w \in d_i, d_i \in D\}|$ is the number of documents in D containing word w; $nd_{bg}(w) = |\{d_i|w \in d_i, d_i \in D_{bg}\}|$ is the number of documents containing word w in the background corpus D_{bg}; λ_{ij} is a binary random variable indicating whether w_{ij} is specific enough.

For each word, we define that a word is *orthodox* if the word is not only semantically close to a semantic focus of the corpus, but also sufficiently specific. We then define the probability that a word or phrase w_{ij} in document d_i is orthodox as:

$$P(\varphi_{ij}|\mathbf{x}_{ij}, \boldsymbol{\pi}_i w_{ij}) = P\big(\phi_{z_{ij}}|\mathbf{x}_{ij}, \boldsymbol{\pi}_i\big)P\big(\lambda_{ij}|w_{ij}\big)$$

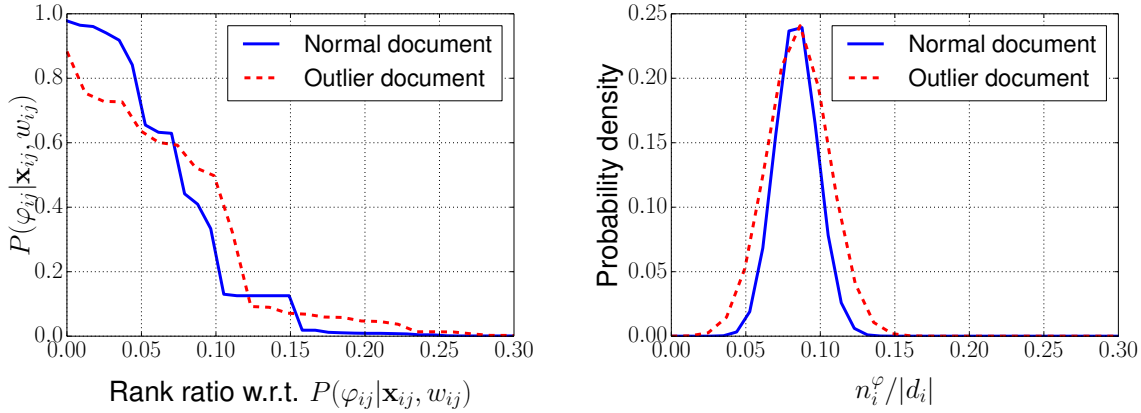
where $\varphi_{ij} = 1$ indicates that w_{ij} (or equivalently $\mathbf{x}_{ij}$) is orthodox.

Now, we can define a second outlierness measure as the expected percentage of words that are *not* orthodox.

$$\Omega_{\mathrm{e}}(d_i) = 1 - \frac{1}{|d_i|}\sum_{j=1}^{|d_i|} P(\varphi_{ij}|\mathbf{x}_{ij}, \boldsymbol{\pi}_i, w_{ij}) \tag{3.6}$$

Noisy content in documents. We present the second issue of normal documents with an example. We compare a normal document in a corpus of New York Times news articles with tag "Health", to another document originally from another corpus, but with its outlierness calculated with regard to the semantic focuses of the "Health" corpus.

In Figure 3.1(a), we show the distribution of inferred orthodox probability $P(\varphi_{ij} = 1|\mathbf{x}_{ij}, w_{ij})$ by ranking the words or phrases according to their probability value. We can observe that the outlier document barely has any words or phrases surely orthodox, while the normal document has 5% of words or phrases with a probability no less than 0.8 to be orthodox. However, if we simply take the average, these two documents become indistinguishable as



(a) Ranked orthodox probability $P(\varphi_{ij}|\mathbf{x}_{ij}, \boldsymbol{\pi}_i, w_{ij})$ (b) Probability distribution of random variable $n_i^{\varphi}/|d_i|$

Figure 3.1: Comparison of a normal document and an outlier document in a news corpus ("Health" topic).

the average is substantially dominated by the "tail" where most words or phrases in either documents are clearly not orthodox. Let n_i^{φ} be a random variable indicating the true number of orthodox words or phrases in document d_i. Since n_i^{φ} follows a Poisson-Binomial distribution, we can plot the probability distribution of n_i^{φ} normalized by the length of the document, as shown in Figure 3.1(b). It can be observed that the difference between the normalized expectation $\mathbb{E}[n_i^{\varphi}]/d_i$ of two documents is insignificant. Therefore, the measure described in Eq. (3.6) will be unable to tell the difference between these two documents.

This example illustrates why the strategy of taking the average over the whole document can make mistakes, and also provides an important insight. As long as a document has a (potentially small) portion of words or phrases that are highly certain to be orthodox, it should not be considered as an outlier. Based on the above observation, we propose a third outlierness measure.

Orthodox quantile outlierness. We define a *quantile-based outlierness* definition to rank document outliers. Notice that the distribution of random variable n_i^{φ} follows a Poisson-Binomial distribution, which is the total number of success trials when one tosses a coin for each word or phrase in the document to determine whether it is orthodox with probability $P(\varphi_{ij}|\mathbf{x}_{ij}, w_{ij})$.

Moreover, we define the first $\frac{1}{1-\theta}$-quantile of the Poisson-Binomial distribution of n_i^{φ} as:

$$q_\theta(n_i^{\varphi}) = \sup_q \{q : P(n_i^{\varphi} \geq q) \geq \theta\} \tag{3.7}$$

where $0 < \theta < 1$ is a given parameter close to 1. Intuitively, it measures the maximum lower bound of n_i^{φ} we can guarantee with confidence θ.

Based on Eq. (3.7), we can give a formalized definition of our proposed outlierness:

$$\Omega_{\theta\text{-q}}(d_i) = 1 - \frac{q_\theta(n_i^{\varphi}) + 1}{|d_i| + 1} \tag{3.8}$$

where the $\frac{1}{1-\theta}$-quantile is normalized by the document length with a smoothing constant. The cumulative probability distribution of a Poisson-Binomial distribution can be efficiently calculated by dynamic programming [19].

The advantage of the last proposed outlierness measure is that it emphasizes more on the highly orthodox words or phrases and eliminates the noise from a number of relatively uncertain ones.

3.4 EXPERIMENT SETUP

3.4.1 Data Sets

New York Times News (NYT). We collected 41,959 news article published in 2013 from The New York Times API[2]. Each article is assigned with a unique label indicating in which section the article is published, such as Arts, Travel, Sports, and Health. There are totally 9 section labels in our collected data set. We treat papers in each section as a corpus D. Thereby we have a set of corpora $\mathcal{D} = \{D_s\}$, without overlapping documents. We also have an external news data set D_e crawled from Google news, with 51,114 news article published in 2015 without any label information.

ArnetMiner Paper Abstracts (ARNET). We employ abstracts of papers published in the field of computer science up to 2013, collected by ArnetMiner [81], and assign each paper into a field, according to Wikipedia[3]. We use papers from a set of domains to serve as an external corpus D_e, while papers in other domains form different corpora $\mathcal{D} = \{D_s\}$. Each domain (*e.g.*, data mining, computational biology, and computer graphics) forms a corpus D_s respectively. Again, notice that the corpora do not have overlapping documents with each other.

A summary is presented in Table 3.1.

[2] `http://developer.nytimes.com/docs`

[3] `https://en.wikipedia.org/wiki/List_of_computer_science_conferences`

Table 3.1: Data set statistics.

Data set	Corpus D		External corpus D_e	
	Avg. $\|D\|$	Avg. $\|d\|$	$\|D_e\|$	Avg. $\|d\|$
NYT	4,662.11	592.66	52,114	471.63
ARNET	2,930.60	137.21	11,463	152.17

Benchmark generation. Since we do not have true labels for outliers in a corpus, we use injection method to generate outlier detection benchmark. For each data set, we randomly select a corpus $D_s \in \mathcal{D}$ and mark all of its document as "normal documents". We then randomly select another corpus $D'_s \in \mathcal{D}, D'_s \neq D_s$, to inject ω documents from D'_s into D_s and mark them as outliers. We confine ω to be a small integer less than 1% of the size of $|D_s|$. More concretely, ω is an integer uniformly sampled from $(0, 0.01|D_s|]$.

For each data set, we randomly generate 10 outlier detection benchmarks, and evaluate the overall performance by the average performance on all the benchmarks.

3.4.2 Methods Evaluated

We compare the performances of the following methods.

Cosine similarity-based. We characterize each document as a vector, and use the negative average cosine similarity between each document and the corpus as outlierness. We use two different ways to vectorize documents: TF-IDF weighted, and paragraph2vec [46]. The two methods are denoted as TFIDF-COS and P2V-COS respectively.

KL divergence-based. We represent each document as a probability distribution, and the entire corpus as another probability distribution. Then we use the KL-divergence between each document and the entire corpus as the outlierness. We also use two different ways to calculate the probability distribution. The first is to estimate the unigram distribution for each document and the entire corpus respectively, denoted as UNI-KL. The other is to first perform LDA on the entire corpus with 10 topics, and then infer topical allocation distribution of each document and the entire corpus. This method is represented as TM-KL.

Our method Our quantile based method is denoted as VMF-Q. We also provide two baselines derived from our own method as an ablation analysis. One method abandons the quantile based outlierness but use the expected orthodox percentage as Equation (3.6), denoted as VMF-E. The other method further removes the penalty on lexical general words and phrases, using Equation (3.5), denoted as VMF-SF.

3.4.3 Evaluation Measures

In most outlier detection applications, people are more concerned with recall. We measure the performance by *recall at a certain percentage*. More specifically, we compute the recall of outlier detection if the user checks a certain percentage r of the top-ranked documents in the output results. Since in our benchmark generation, the percentage of outliers does not exceed 1%. Therefore, the perfect results for any $r \geq 1\%$ should be 1.0.

We choose r to be 1%, 2%, and 5% respectively and evaluate different methods with recall at top-r (percentage). We also report the performance in terms of mean average precision (MAP).

3.4.4 Parameter Configurations

All benchmark data sets are preprocessed as described in Section 3.2. In the NYT data set we remove words or phrases within top 20% with respect to document frequency, while in the ARNET data set we remove the top 10%. The document frequency is calculated based on a background corpus D_{bg}, which is the same as the external corpus of NYT. Word embedding are trained on the external data set D_e using code of Mikolov *et al.* [61] with default parameter configurations, where the embedded vector length is set to 200. For paragraph2vec, we learn the length-100 vectors for each document along with the external data set to guarantee sufficient training data.

For the prior vMF distribution, we set $C_0 = 0.1$, a sufficiently small number so the prior distribution is close to a uniform distribution. $\boldsymbol{\mu}_0$ is set as a normalized all-1 vector. We also set $m_0 = log(100)$, and $\sigma^2 = 0.01$. The total number for Gibbs sampling is set to be 50 times of the total count of z_{ij}'s (*i.e.* $\eta = 50$). The number of vMF distributions T is set to 20 in the NYT data set and 10 in the ARNET data set respectively, due to the smaller sizes of corpora in the ARNET data set.

To determine semantic focuses, we set threshold parameter $\beta = 0.55$ for both data sets. The confidence parameter θ in outlierness calculation is set to 0.95 in both data sets. Our experiments later will show the performance is relatively robust to different configurations of both parameters.

3.5 RESULTS

We present the experimental results in this section.

Performance comparison. Table 3.2 shows performance of different outlier document

Table 3.2: Performance comparison of different outlier document detection methods. All results are shown as percents.

Data set	Method	MAP	Rcl@1%	Rcl@2%	Rcl@5%
NYT	TFIDF-COS	05.03	04.73	06.72	14.72
	P2V-COS	22.07	23.45	44.64	66.18
	UNI-KL	10.28	11.92	16.32	31.34
	TM-KL	14.51	16.50	16.50	24.67
	VMF-SF	33.70	31.03	44.45	62.60
	VMF-E	36.57	35.91	49.41	67.56
	VMF-Q	**41.88**	**56.99**	**63.29**	**79.23**
ARNET	TFIDF-COS	08.99	15.40	18.75	30.23
	P2V-COS	07.39	10.51	14.78	24.14
	UNI-KL	07.46	14.13	22.26	39.40
	TM-KL	10.09	12.04	15.37	20.24
	VMF-SF	10.69	12.05	22.58	44.51
	VMF-E	10.51	12.67	25.92	45.37
	VMF-Q	**19.74**	**22.40**	**34.40**	**53.87**

detection methods. It can be observed that our method outperforms all the baselines in both data sets. In both data sets, VMF-Q can achieve a 45% to 135% increase from baselines in terms of recall by examining the top 1% outliers. Generally, performances of most methods are lower in the ARNET data set comparing to NYT, potentially because the relatively short document lengths and more technical terminologies in ARNET.

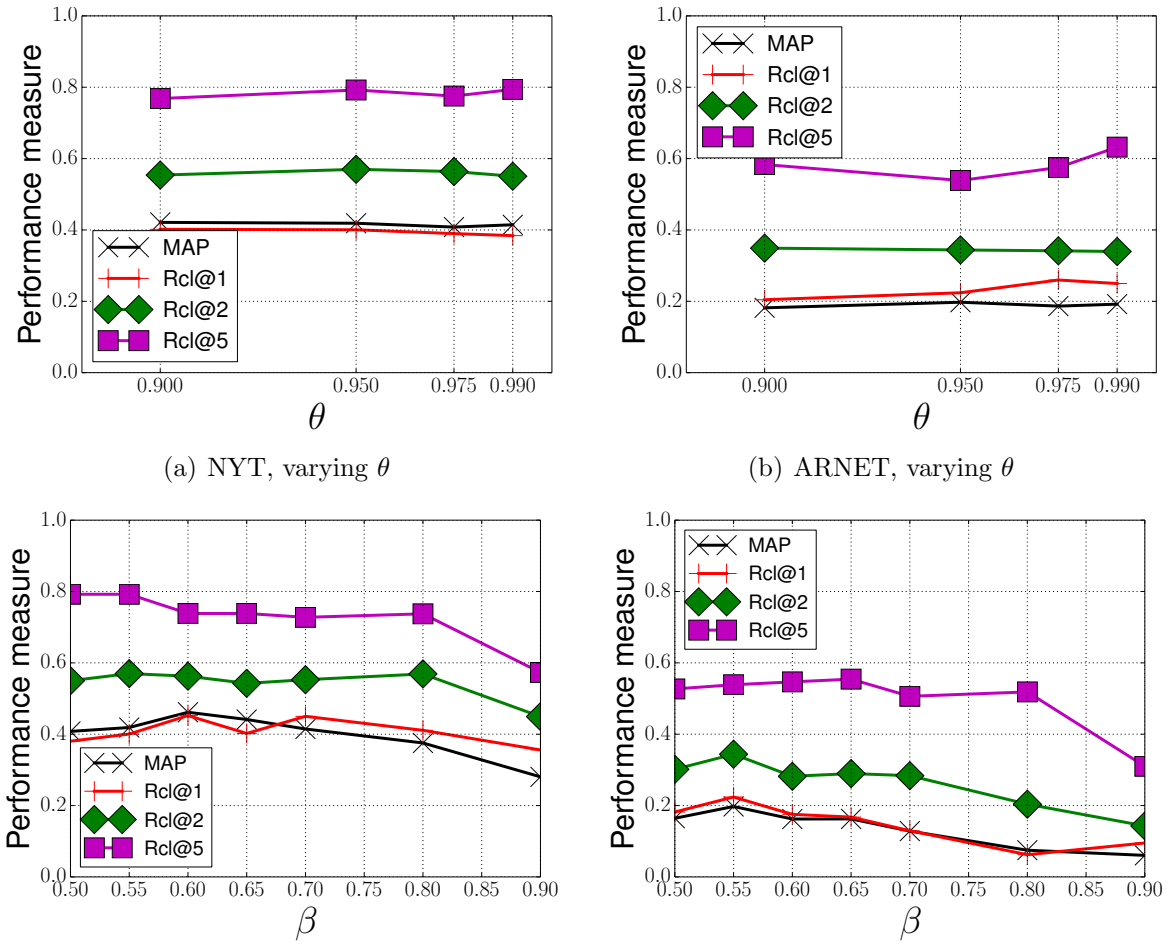
In the NYT data set, by examining the top 1% of outlier documents ranked by VMF-Q, one can find 56% of injected outliers, In contrast, the best performing baseline P2V-COS only achieves 23%, indicating the difficulty of the problem.

In the ARNET data set, VMF-Q achieves 22-34% recall when the top 1-2% of ranked outlier documents are examined, while other methods can barely have a recall greater than 15% if only top-1% of outlier documents are reviewed.

Generally, performances of most methods are lower in the ARNET data set comparing to NYT, potentially because the relatively short document lengths and more technical terminologies in ARNET.

Ablation analysis. Both refinements of the outlierness measure benefits the performance. Specifically, by changing the average based outlierness to quantile based outlierness, the recall@1% can be improved by 50-75%, and the recall@5% can also be improved by more than 17%.

Sensitivity studies of parameters. We study if our proposed method is sensitive to the



(a) NYT, varying θ

(b) ARNET, varying θ

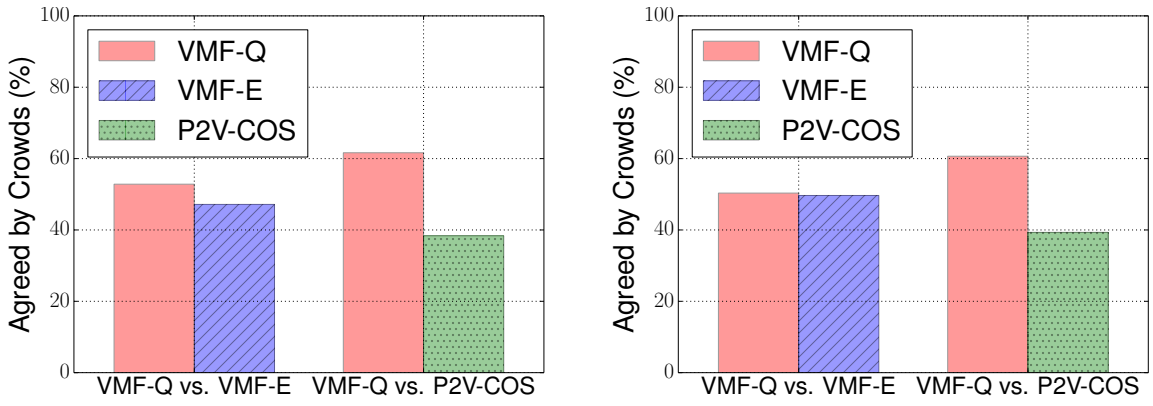
(c) NYT, varying β

(d) ARNET, varying β

Figure 3.2: Performance of outlier document detection with different parameter configurations.

confidence parameter θ and filtering threshold parameter β. We compare the performance of VMF-Q by varying each parameter on both data sets. Figure 3.2(a) and 3.2(b) show that the performance is not very sensitive to different values of θ, as long as θ is sufficiently large (close to 1). Figure 3.2(c) and 3.2(d) show that the performance is relatively stable when β is between 0.5 and 0.7, but drops a little when β is set to larger value.

Human judgments. We compare VMF-Q to VMF-E and P2V-COS respectively by crowdsourcing, *without* artificially inserting "outliers". We conduct this experiments on two corpora in NYT data sets with topic "Health" and "Art" respectively. To compare two methods, we randomly select pairs of documents d_i and d_j such that both are ranked as top-10% outliers by at least one method, but their orders in the two rankings disagree. We conduct the experiments on CrowdFlower. Online crowd workers are given d_i and d_j as well



(a) Health

(b) Art

Figure 3.3: Crowd evaluation to compare different outlier detection methods on two corpora in NYT data set.

as other documents in the corpus, and are asked to judge which one of d_i and d_j deviates more from the corpus. For each corpus, we select 200 pairs of documents.

Before taking the questions, each crowd worker needs to go through at least 10 "test questions" which we know the correct answer. These questions are constructed by taking one document from the corpus as d_i and another document not from the corpus as d_j. Therefore, the one not from the corpus should be the answer. A crowd worker needs to achieve no less than 80% of accuracy to be eligible to work on actual questions, and the accuracy needs to be maintained over 80% during the work, which is measured by "test questions" hidden in actual questions. Each question is answered by 3 workers. The final answer is determined by majority voting.

Figure 3.3 presents the results. On both corpora, there are significantly more workers tend to agree with VMF-Q comparing to P2V-COS, with significance level $\alpha = 0.05$. This further verifies that our method VMF-Q can achieve better performance than the P2V-COS baseline. On the other hand, on both data sets we can still observe more workers favoring VMF-Q than VMF-E, but the difference is not as large as the difference between VMF-Q and P2V-COS.

Case study. We also conduct a case study to show how our proposed method outperforms other baselines. Table 3.3 shows two pairs of documents in "Health" corpus of NYT data set. The left two columns show some comparing methods and their higher ranked outlier documents. The row of "Crowds" shows the outlier document chosen by human workers from the crowdsourcing platform, with a consensus of opinions from multiple workers.

In the first document pair, document A is about gun control policy and is substantially irrelevant to "Health" topic, while document B is about lung infection cases. Document

Table 3.3: Case study of documents in "Health" corpus of NYT data set. We present several pairs of documents and how different methods rank the pair. The "Outlier" column indicates the document ranked higher in the outlier document ranking generated by the corresponding methods, and the row "Crowds" shows the ranking given by human evaluators.

Method	Outlier	Document A	Document B
P2V-COS	Doc B	*States with the most gun control laws have the fewest gun-related deaths, according to a study that suggests sheer quantity of measures might ...*	*A prominent Scottish bagpiping school has warned pipers around to world to clean their instruments regularly after one of its longtime members ...*
VMF-E	Doc A		
VMF-Q	Doc A		
Crowds	Doc A		
P2V-COS	Doc B	*There's more evidence that U.S. births may be leveling off after years of decline. The number of babies born last year only slipped a little, ...*	*Young men in a state prison for juveniles and professors of library science from the University of South Carolina have joined forces to fight AIDS with a graphic novel*
VMF-E	Doc B		
VMF-Q	Doc A		
Crowds	Doc A		

A is a significant outlier, and VMF-Q and VMF-E also agree with our intuition. However, paragraph2vec (P2V) ranks document B higher, probably because it tries to summarize the entire document.

In the second document pair, document B is clearly *not* an outlier as the story is about a new book of AIDS. In comparison, document A discussing U.S. population is an outlier. However, a great part of document B is about the content of the book, which confuses baselines P2V and VMF-E, as both methods tend to summarize the entire document and highly relevant words like "AIDS" are overwhelmed by the majority of the document. The only method that agrees with human annotators is VMF-Q.

3.6 SUMMARY

In this chapter, we propose a novel task of detecting document outliers from a given corpus. We propose a generative model to identify semantic focuses of a corpus, each represented as a vMF distribution in the embedded space. We also design a document outlierness measure. We experimentally verify the effectiveness of our methods. We hope this work provides insights for further studies on outlier document texts in specific domains, and in more challenging settings such as detecting outliers from crowdsourced data.

CHAPTER 4: COST-SENSITIVE OUTLIER DETECTION

4.1 OVERVIEW

In this chapter, we study the efficiency of identifying outliers. In some real-world scenarios, the cost of identifying outliers may not be negligible. For example, if a data analyst needs to perform interactive outlier analysis of text data with keywords, she will expect real-time response from the outlier detection system. However, the calculation for detecting outliers may not meet the latency requirements while handling gigantic text data sets. On the other hand, if the system is requesting data from a third party, the monetary cost may be another concern.

We study to minimize the cost of identifying outliers. We study a simplified but more general version, where the problem is formalized as a variation of the multi-armed bandit problem. We make the following major contributions:

- We propose a Round-Robin sampling algorithm, with a theoretical guarantee of its correctness as well as a theoretical upper bound of its total number of pulls.
- We further propose an improved algorithm Weighted Round-Robin, with the same correctness guarantee, and a better upper bound of its total number of pulls.
- We verify our algorithms on both synthetic and real data sets. Our Round-Robin algorithm has near 100% accuracy, while reducing the cost of a competitive baseline up to 99%. Our Weighted Round-Robin algorithm further reduces the cost by around 60%, with even smaller error.

4.2 PROBLEM DEFINITION

In this section, we describe the problem of identifying outlier arms in a multi-armed bandit. We start with recalling the settings of the multi-armed bandit model.

Multi-armed bandit. A multi-armed bandit (MAB) consists of n-arms, where each arm is associated with a reward distribution. The (unknown) expectation of each reward distribution is denoted as y_i. At each iteration, the algorithm is allowed to select an arm i to play (pull), and obtain a sample reward $x_i^{(j)} \in \mathcal{R}$ from the corresponding distribution, where j corresponds to the j-th samples obtained from the i-th arm. We further use $\mathbf{x}_i$ to represent all the samples obtained from the i-th arm, and $\mathbf{y}$ to represent the configuration of all the y_i's.

Problem formalization. We study to identify outlier arms with extremely high reward expectations compared to other arms in the bandit. To define "outlier arms", we adopt a general statistical rule named k-sigma: The arms with reward expectations higher than the mean plus k standard deviation of all arms are considered as outliers. Formally, we define the mean of all the n arms' reward expectations as well as their standard deviation as:

$$\mu_y = \frac{1}{n}\sum_{i=1}^{n} y_i, \quad \sigma_y = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \mu_y)^2}$$

We define a threshold function based on the above estimators as:

$$\theta = \mu_y + k\sigma_y$$

An arm i is defined as an outlier arm iff $y_i > \theta$ and is defined as a normal (non-outlier) arm iff $y_i < \theta$. We denote the set of outlier arms as $\Omega = \{i \in [n] | y_i > \theta\}$.

In a multi-armed bandit setting, the value of y_i for each arm is unknown. Instead, the system needs to pull one arm at each iteration to obtain a sample, and estimate the value y_i for each arm and the threshold θ from all the obtained samples $\mathbf{x}_i, \forall i$. We introduce the following estimators:

$$\hat{y}_i = \frac{1}{m_i}\sum_{j} x_i^{(j)}, \quad \hat{\mu}_y = \frac{1}{n}\sum_{i=1}^{n}\hat{y}_i, \quad \hat{\sigma}_y = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - \hat{\mu}_y)^2}, \quad \hat{\theta} = \hat{\mu}_y + k\hat{\sigma}_y$$

where m_i is the number of times the arm i is pulled.

We focus on the *fixed confidence* setting. The formal definition of our problem can be described as:

Problem 4.1. *Given a n-arm bandit where each arm i has an unknown expectation y_i and a small tolerance constant $0 < \delta < 1$, we aim to design an efficient pulling strategy such that the strategy can return the set of outlier arms Ω with the probability of being exactly correct no smaller than $1 - \delta$.*

The fewer total number of pulls, *i.e.* $T = \sum_i m_i$, the better, because each pull has a economic or time cost. Note that this is a *pure exploration* setting, i.e., the reward incurred during exploration is irrelevant to the cost.

4.3 ALGORITHMS

In this section, we propose several algorithms, and present the theoretical guarantee of each algorithm.

4.3.1 Round-Robin Algorithm

The most simple algorithm is to pull arms in a round-robin way. That is, the algorithm starts from arm 1 and pulls arm $2, 3, \cdots$ respectively, and goes back to arm 1 after it iterates over all the n arms. The process continues until a certain termination condition is met.

Intuitively, the algorithm should terminate when it is confident about whether each arm is an outlier. We achieve this by using the confidence interval of each arm's reward expectation as well as the confidence interval of the outlier threshold. If the significance levels of these intervals are carefully set, and each reward expectation's confidence interval has no overlap with the threshold's confidence interval, we can safely terminate the algorithm while guaranteeing correctness with desired high probability. In the following, we first discuss the formal definition of confidence intervals, as well as how to set the significance levels. Then we present the formal termination condition.

Confidence intervals. We provide a general definition of *confidence intervals* for $\hat{y}_i$ and $\hat{\theta}$. The confidence interval for $\hat{y}_i$ at significance level δ' is defined as $[\hat{y}_i - \beta_i(m_i, \delta'), \hat{y}_i + \beta_i(m_i, \delta')]$, such that:

$$\mathbb{P}(|\hat{y}_i - y_i| > \beta_i(m_i, \delta')) < 2\delta'$$

Similarly, the confidence interval for $\hat{\theta}$ at significance level δ' is defined as $[\hat{\theta} - \beta_\theta(\mathbf{m}, \delta'), \hat{\theta} + \beta_\theta(\mathbf{m}, \delta')]$, such that:

$$\mathbb{P}(|\hat{\theta} - \theta| > \beta_\theta(\mathbf{m}, \delta')) < 2\delta'$$

The concrete form of confidence interval may vary with the reward distribution associated with each arm. We defer the discussion of concrete form of confidence interval to Section 4.3.3.

In our algorithm, we update the significance level δ' for the above confidence intervals at each iteration. After T pulls, the δ' should be set as:

$$\delta' = \frac{3\delta}{\pi^2(n+1)T^2} \tag{4.1}$$

In the following discussion, we omit the parameters in β_i and β_θ when they are clear from

Input: n arms, outlier parameter k
Output: A set $\hat{\Omega}$ of outlier arms

Pull each arm i once $\forall i \in [n]$; `// Initialization`
$T \leftarrow n$;
Update $\hat{y}_i, m_i, \beta_i, \forall i \in [n]$ and $\hat{\theta}, \beta_\theta$;
$i \leftarrow 1$;
while $A \neq \emptyset$ **do**
    $i \leftarrow i\%n + 1$; `// Round-robin`
    Pull arm i;
    $T \leftarrow T + 1$;
    Update $\hat{y}_i, m_i, \beta_i$ and $\hat{\theta}, \beta_\theta$;
return $\hat{\Omega}$ according to Eq. (4.3);

Algorithm 4.1: Round-Robin Algorithm (RR)

the context.

Active arms. At any time, if $\hat{y}_i$'s confidence interval overlaps with $\hat{\theta}$'s confidence interval, then the algorithm cannot confidently tell if the arm i is an outlier or a normal arm. We call such arms *active*, and vice versa. Formally, an arm i is active, denoted as $\text{ACTIVE}_i = \text{TRUE}$, iff

$$\begin{cases} \hat{y}_i - \beta_i < \hat{\theta} + \beta_\theta, & \text{if } \hat{y}_i > \hat{\theta}; \\ \hat{y}_i + \beta_i > \hat{\theta} - \beta_\theta, & \text{otherwise.} \end{cases} \tag{4.2}$$

We denote the set of active arms as $A = \{i \in [n] | \text{ACTIVE}_i = \text{TRUE}\}$. With this definition, the termination condition is simply $A = \emptyset$. When this condition is met, we return the result set:

$$\hat{\Omega} = \{i | \hat{y}_i > \hat{\theta}\} \tag{4.3}$$

The algorithm is outlined in Algorithm 4.1.

Theoretical results. We first show that if the algorithm terminates with no active arms, the returned outlier set will be correct with high probability.

Theorem 4.1 (Correctness)**.** *With probability $1 - \delta$, if the algorithm terminates after a certain number of pulls T when there is no active arms i.e. $A = \emptyset$, then the returned set of outliers will be correct, i.e. $\hat{\Omega} = \Omega$.*

We can also provide an upper bound for the efficiency of the algorithm in a specific case when all the reward distributions are bounded within $[a, b]$ where $b - a = R$. In this case, the

confidence intervals can be instantiated as discussed in Section 4.3.3. And we can accordingly obtain the following results:

Theorem 4.2. *With probability* $1-\delta$*, the total number of pulls* T *needed for the algorithm to terminate is bounded by*

$$T \leq 8R^2\mathbf{H}_{RR}\left[\log\left(\frac{4R^2\pi^2(n+1)\mathbf{H}_{RR}}{3\delta}\right)+1\right]+4n \tag{4.4}$$

where

$$\mathbf{H}_{RR} = \mathbf{H}_1\big(1+\sqrt{l(k)}\big)^2, \quad \mathbf{H}_1 = \frac{n}{\min_{i\in[n]}(y_i-\theta)^2},$$
$$l(k) = \left[\sqrt{\frac{(1+k\sqrt{n-1})^2}{n}}+\sqrt{\frac{k^2}{2\log\left(\frac{\pi^2 n^3}{3\delta}\right)}}\right]^2$$

4.3.2 Weighted Round-Robin Algorithm

The round-robin algorithm evenly distributes resources to all the arms. Intuitively, active arms deserve more pulls than inactive arms, since the algorithm is almost sure about whether an inactive arm is outlier already.

Based on this idea, we propose an improved algorithm. We allow the algorithm to sample the active arms ρ times as many as inactive arms, where $\rho \geq 1$ is a real constant. Since ρ is not necessarily an integer, we use a method similar to stride scheduling to guarantee the ratio between number of pulls of active and inactive arms are approximately ρ in a long run. The algorithm still pulls by iterating over all the arms. However, after each arm is pulled, the algorithm can decide either to stay at this arm for a few "extra pulls," or proceed to the next arm. If the arm pulled at the T-th iteration is the same as the arm pulled at the $(T-1)$-th iteration, we call the T-th pull an "extra pull." Otherwise, we call it a "regular pull." We keep a counter c_i for each arm i. When $T > n$, after the algorithm performs a regular pull on arm i, we add ρ to the counter c_i. If this arm is still active, we keep pulling this arm until $m_i \geq c_i$ or it becomes inactive. Otherwise we proceed to the next arm to perform the next regular pull.

This algorithm is named Weighted Round-Robin, and outlined in Algorithm 4.2.

Theoretical results. Since the Weighted Round-Robin algorithm has the same termination condition, according to Theorem 4.1, it has the same correctness guarantee.

We can also bound the total number of pulls needed for this algorithm when the reward

Input: n arms, outlier parameter k, ρ
Output: A set of outlier arms $\hat{\Omega}$

Pull each arm i once $\forall i \in [n]$; // Initialization
$T \leftarrow n$;
Update $\hat{y}_i, m_i, \beta_i, \forall i \in [n]$ and $\hat{\theta}, \beta_\theta$;
$c_i \leftarrow 0, \forall i \in [n]$;
$i \leftarrow 1$;
while $A \neq \emptyset$ **do**
    $i \leftarrow i\%n + 1$; // Next regular pull
    $c_i \leftarrow c_i + \rho$;
    repeat
        Pull arm i;
        $T \leftarrow T + 1$;
        Update $\hat{y}_i, m_i, \beta_i$ and $\hat{\theta}, \beta_\theta$;
    until $i \notin A \bigvee m_i \geq c_i$;
return $\hat{\Omega}$ according to Eq. (4.3);

Algorithm 4.2: Weighted Round-Robin Algorithm (WRR)

distributions are bounded.

Theorem 4.3. *With probability $1 - \delta$, the total number of pulls T needed for the Weighted Round-Robin algorithm to terminate is bounded by*

$$T \leq 8R^2\mathbf{H}_{WRR}\left[\log\left(\frac{2R^2\pi^2(n+1)\mathbf{H}_{WRR}}{3\delta}\right) + 1\right] + 2(\rho+2)n \tag{4.5}$$

where

$$\mathbf{H}_{WRR} = \left(\frac{\mathbf{H}_1}{\rho} + \frac{(\rho-1)\mathbf{H}_2}{\rho}\right)\left(1 + \sqrt{l(k)\rho}\right)^2, \quad \mathbf{H}_2 = \sum_i \frac{1}{(y_i - \theta)^2}$$

Determining ρ. One important parameter in this algorithm is ρ. For bounded reward distributions, we have a closed form upper bound of T as $O(\mathbf{H}_{\text{WRR}} \log \frac{\mathbf{H}_{\text{WRR}}}{\delta})$. The lower bound of T is independent of ρ. We conjecture the lower bound to be $\Omega(\mathbf{H}_2 \log \frac{\mathbf{H}_2}{\delta})$. We aim to find the ρ that minimizes the gap between the upper bound and the lower bound. We formalize the objective as finding a ρ to minimize $\mathbf{H}_{\text{WRR}}/\mathbf{H}_2$. Since we do not know the reward distribution configuration $\mathbf{y}$, we use the minimax principle to find ρ^* that optimizes the most difficult configuration $\mathbf{y}$, namely

$$\rho^* = \operatorname*{argmin}_{\rho \geq 1} \sup_{\mathbf{y}} \frac{\mathbf{H}_{\text{WRR}}}{\mathbf{H}_2}$$

Since $\frac{\mathbf{H}_1}{n} \leq \mathbf{H}_2 \leq \mathbf{H}_1$, and $\frac{\mathbf{H}_{\text{WRR}}}{\mathbf{H}_2}$ is monotonically increasing with regard to $\frac{\mathbf{H}_1}{\mathbf{H}_2}$, we can obtain the optimal value ρ^* as

$$\rho^* = \frac{(n-1)^{\frac{2}{3}}}{l^{\frac{1}{3}}(k)} \tag{4.6}$$

Theoretical comparison with RR. We compare theses two algorithms by comparing their upper bounds. Essentially, we study $\mathbf{H}_{\text{WRR}}/\mathbf{H}_{\text{RR}}$ since the two bounds only differ in this term after a small constant is ignored. We have

$$\frac{\mathbf{H}_{\text{WRR}}}{\mathbf{H}_{\text{RR}}} = \left(\frac{1}{\rho} + \frac{\rho-1}{\rho}\frac{\mathbf{H}_2}{\mathbf{H}_1}\right)\left(\frac{1+\sqrt{l(k)\rho}}{1+\sqrt{l(k)}}\right)^2 \tag{4.7}$$

The ratio between $\mathbf{H}_2$ and $\mathbf{H}_1$ indicates how much cost WRR will save from RR. Notice that $\frac{1}{n} \leq \frac{\mathbf{H}_2}{\mathbf{H}_1} \leq 1$. In the degenerated case $\mathbf{H}_2/\mathbf{H}_1 = 1$, WRR does not save any cost from RR. This case occurs only when all arms have identical reward expectations, which is rare and not interesting. However, if $\mathbf{H}_2/\mathbf{H}_1 = 1/n$, by setting ρ to the optimal value in Eq. (4.6), it is possible to save a substantial portion of pulls. In this scenario, the RR algorithm will iteratively pull all the arms until the arm closest to the threshold i^* confidently determined as outlier or normal. However, the WRR algorithm is able to invest more pulls on arm i^* as it remains active, while pulling other arms for fewer times, only to obtain a more precise estimate of the outlier threshold.

4.3.3 Confidence Interval Instantiation

With different prior knowledge of reward distributions, confidence intervals can be instantiated differently. We introduce the confidence interval for a relatively general scenario, where reward distributions are bounded.

Bounded distribution. Suppose the reward distribution of each arm is bounded in $[a, b]$, and $R = b - a$.

According to Hoeffding's inequality and McDiarmid's inequality, we can derive the confidence interval for y_i as

$$\beta_i(m_i, \delta') = R\sqrt{\frac{1}{2m_i}\log\left(\frac{1}{\delta'}\right)}, \quad \beta_\theta(\mathbf{m}, \delta') = R\sqrt{\frac{l(k)}{2h(\mathbf{m})}\log\left(\frac{1}{\delta'}\right)}$$

where m_i is the number of pulls of arm i so far, and $h(\mathbf{m})$ is the harmonic mean of all the

m_i's.

Bernoulli distribution. In many real applications, each arm returns a binary sample 0 or 1, drawn from a Bernoulli distribution. We use the following confidence intervals heuristically.

We leverage a confidence interval presented in [5], defined as

$$\beta_i(m_i, \delta') = z_{\delta'}\sqrt{\frac{\tilde{p}(1-\tilde{p})}{m_i}}\ , \qquad \beta_\theta(\mathbf{m}, \delta') = \sqrt{\sum_i \left(\frac{k\hat{y}_i}{n\sqrt{\hat{\sigma}_y}} + \frac{1}{n}\right)^2 \beta_i^2}$$

where

$$\tilde{p} = \frac{m_i^+ + \frac{z_{\delta'}^2}{2}}{m_i + z_{\delta'}^2}\ , \qquad z_{\delta'} = \mathrm{erf}^{-1}(1-\delta')$$

m_i^+ is the number of samples that equal to 1 among m_i samples, and $z_{\delta'}$ is value of the *inverse error function*.

4.4 EXPERIMENTAL RESULTS

In this section, we present experiments to evaluate both the effectivenss and efficiency of proposed algorithms.

4.4.1 Datasets

Synthetic. We construct several synthetic datasets with varying number of arms $n = 20, 50, 100, 200$, and varying $k = 2, 2.5, 3$. There are 12 configurations in total. For each configuration, we generate 10 random test cases. For each arm, we draw its reward from a Bernoulli distribution $Bern(y_i)$.

Twitter. We consider the following application of detecting outlier locations with respect to keywords from Twitter data. A user has a set of candidate regions $L = \{l_1, \cdots, l_n\}$, and is interested in finding outlier regions where tweets are extremely likely to contain a keyword w. In this application, each region corresponds to an arm. A region has an unknown probability of generating a tweet containing the keyword, which can be regarded as a Bernoulli distribution. We collect a Twitter dataset with $1,500,000$ tweets from NYC, associated with its latitude and longitude. We divide the entire space into regions of $2'' \times 2''$ in latitude and longitude respectively. We select 47 regions with more than $5,000$ tweets as arms and select

20 keywords as test cases.

4.4.2 Setup

Methods for comparison. Since the problem is new, there is no directly comparable solution in existing work. We design two baselines for comparative study.

- *Naive Round-Robin (NRR).* We play arms in a round-robin fashion, and terminate as soon as we find the estimated outlier set $\hat{\Omega}$ has not changed in the last consecutive $1/\delta$ pulls. $\hat{\Omega}$ is defined as in Eq. (4.3). This baseline reflects how well the problem can be solved by RR with a heuristic termination condition.
- *Iterative Best Arm Identification (IB).* We apply a state-of-the-art best arm identification algorithm [20] iteratively. We first apply it to all n arms until it terminates, and then remove the best arm and apply it to the rest arms. We repeat this process until the current best arm is not in $\hat{\Omega}$, where the threshold function is heuristically estimated based on the current data. We then return the current $\hat{\Omega}$. This is a strong baseline that leverages the existing solution in best-arm identification.

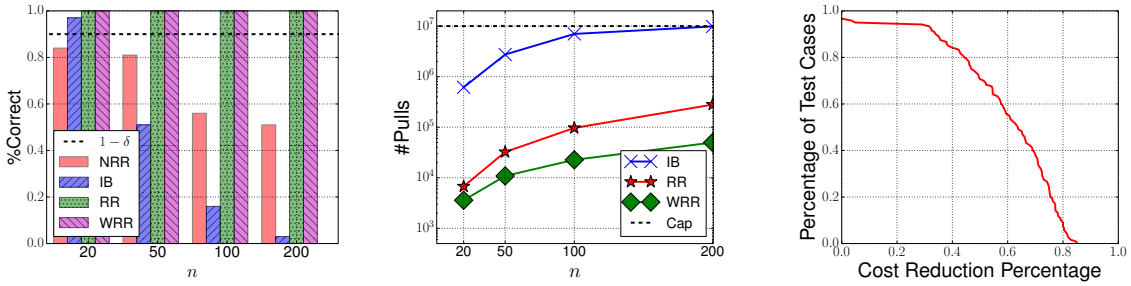
Then we compare them with our proposed two algorithms, Round-Robin (RR) and Weighted Round-Robin (WRR).

Parameter configurations. For both of our algorithms, we derived the confidence intervals based on Bernoulli distribution. Since some algorithm takes extremely long time to terminate in certain cases, we place a cap on the total number of pulls. Once an algorithm runs for 10^7 pulls, the algorithm is forced to terminate and output the current estimated outlier set $\hat{\Omega}$. We set $\delta = 0.1$.

For each test case, we run the experiments for 10 times, and take the average of both the correctness metrics and number of pulls.

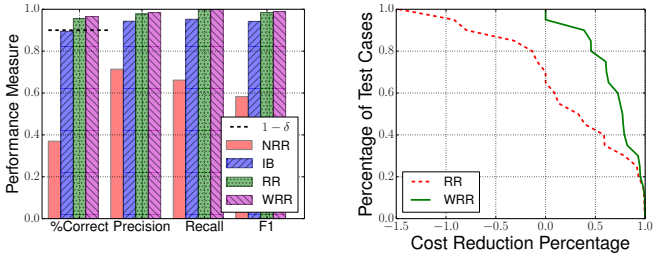
4.4.3 Results

Performance on Synthetic. Figure 4.1(a) shows the correctness of each algorithm when n varies. It can be observed that both of our proposed algorithms achieve perfect correctness on all the test sets. In comparison, the NRR baseline has never achieved the desired level of correctness. Based on the performance on correctness, the naive baseline NRR does not qualify an acceptable algorithm, so we only measure the efficiency of the rest algorithms.



(a) % Exactly Correct (b) Avg. #Pulls vs. n (c) WRR's Cost Reduction wrt RR

Figure 4.1: Effectiveness and efficiency studies on Synthetic data set. Cap indicates the maximum number of pulls we allow an algorithm to run.



(a) Correctness comparison (b) Cost Reduction wrt IB

Figure 4.2: Effectiveness and efficiency studies on Twitter dataset.

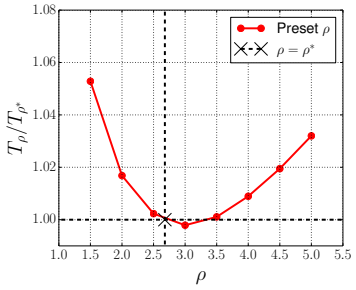
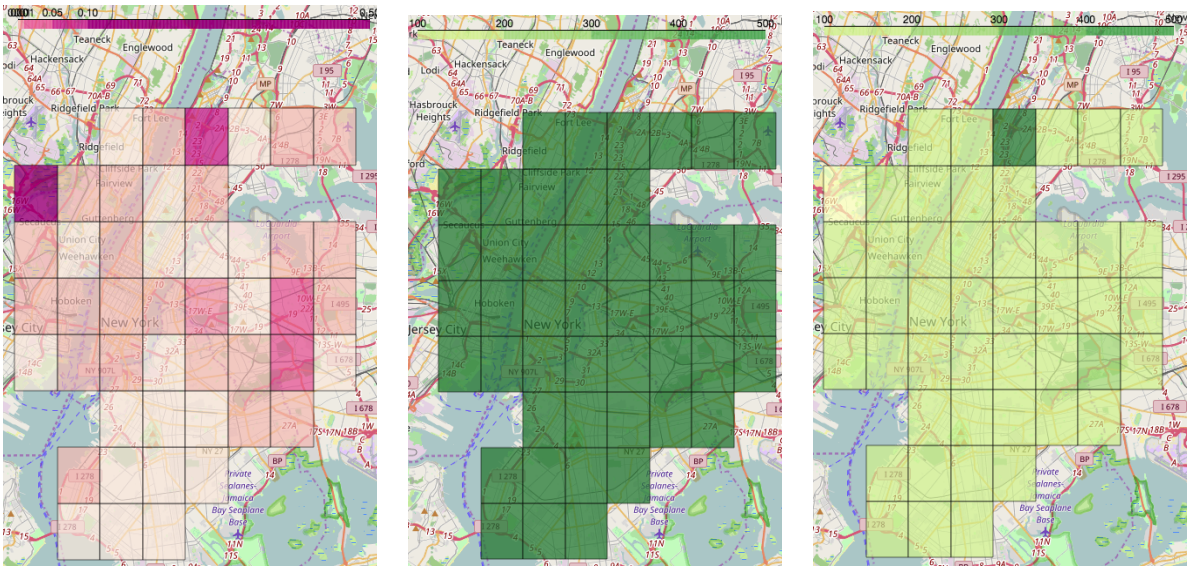


Figure 4.3: Ratio between avg. #pulls with a given ρ and with $\rho = \rho^*$.

We plot the average number of pulls each algorithm takes before termination varying with the number of arms n in Figure 4.1(b). On all the different configurations of n, IB takes a much larger number of pulls than WRR and RR, which makes it 1-3 orders of magnitude as costly as WRR and RR. At the same time, RR is also substantially slower than WRR, with the gap gradually increasing as n increases. This shows our design of additional pulls helps. Figure 4.1(c) further shows that in 80% of the test cases, WRR can save more than 40% of cost from RR; in about half of the test cases, WRR can save more than 60% of the cost.

Performance on Twitter. Figure 4.2(a) shows the correctness of different algorithms on Twitter data set. As one can see, both of our proposed algorithms qualify the correctness requirement, i.e., the probability of returning the exactly correct outlier set is higher than $1 - \delta$. The NRR baseline is far from reaching that bar. The IB baseline barely meets the bar, and the precision, recall and F1 measures show that its returned result is averagely a good approximate to the correct result, with an average F1 metric close to 0.95. This once again confirms that IB is a strong baseline.

We compare the efficiency of IB, RR and WRR algorithms in Figure 4.2(b). In this figure, we plot the cost reduction percentage for both RR and WRR in comparison with IB. WRR



(a) Ground-truth y_i's (b) # pulls by RR (c) # pulls by WRR

Figure 4.4: Case study on Twitter dataset for keyword "stadium." Each plotted square represents a region regarded as an arm in our experiments. Darker color indicates higher value.

is a clear winner. In almost 80% of the test cases, it saves more than 50% of IB's cost, and in about 40% of the test cases, it saves more than 75% of IB's cost. In contrast, RR's performance is comparable to IB. In approximately 30% of the test cases, RR is actually slower than IB and has negative cost reduction, though in another 40% of the test cases, RR saves more than 50% of IB's cost.

Tuning ρ. In order to experimentally justify our selection of ρ value, we test the performance of WRR on a specific setting of synthetic data set ($n = 15$, $k = 2.5$) with varying preset ρ values. Figure 4.3 shows the average number of pulls of 10 test cases for each ρ in $\{1.5, 2, \ldots, 5\}$, comparing to the performance with $\rho = \rho^*$ according to Eq. (4.6). It can be observed that the performance of $\rho = \rho^*$ is very close to the best performance when $\rho = 3$. A further investigation reveals that the $\frac{\mathbf{H}_1}{\mathbf{H}_2}$ of these test cases vary from 3 to 14. Although we choose ρ^* based on an extreme assumption $\frac{\mathbf{H}_1}{\mathbf{H}_2} = n$, its average performance is found to be close to the optimal even when the data do not satisfy the assumption.

Case study. We conduct a detailed case study to compare the behavior of RR and WRR.

For keyword "stadium", Figure 4.4(a) presents the parameters y_i's of different regions (arms). The largest value appears at a football stadium, which is MetLife stadium, followed by two baseball fields (Yankee stadium and Citi field) and a historic stadium site (Forest Hill). However, only the region containing MetLife stadium is an outlier by 2σ rule, probably

due to the difference of the size between a football stadium and a baseball field, and the tweeting frequency.

Figure 4.4(b) shows the number of pulls on each arms when the RR algorithm terminates. As expected, all the arms are pulled for almost the same number, which is more than 500. In comparison, as shown in Figure 4.4(c), when WRR terminates, only the arm containing Yankee stadium is pulled for around 500 times, while all the other arms are only pulled for fewer than 200 times. This is because the WRR algorithm focuses more on the arms closer to the outlier threshold, which are harder to be determined as outlier or normal. It saves a lot of iterations on other arms.

4.5 THEORETICAL ANALYSIS

4.5.1 Correctness (Theorem 4.1)

We start by confining our discussion into an event, where y_i and θ do not fall out of the given confidence intervals.

Lemma 4.1. *Suppose we are given confidence interval functions $\beta_i(m_i, \delta')$ for $\forall i$ and $\beta_\theta(\mathbf{m}, \delta')$ related to the number of pulls and an arbitrary error probability $0 < \delta' < 1$. They satisfy*

$$\mathbb{P}(|\hat{y}_i - y_i| > \beta_i(m_i, \delta')) < 2\delta'$$
$$\mathbb{P}(|\hat{\theta} - \theta| > \beta_\theta(\mathbf{m}, \delta')) < 2\delta'$$

Define the random event

$$\mathcal{E} = \left\{ |\hat{y}_i - y_i| \leq \beta_i(T, m_i) \bigwedge |\hat{\theta} - \theta| \leq \beta_\theta(T, \mathbf{m}), \forall i, \forall T \right\}$$

Suppose a sequence $S = [I_1, I_2, \ldots]$ is an infinite sequence where $1 \leq I_t \leq n$ is a integer, representing the arm pulled at iteration t. If we properly set the tolerance of confidence intervals δ' according to the current number of iterations, namely

$$\delta'(T) = \frac{3\delta}{\pi^2(n+1)T^2}$$

then for any S, we have

$$\mathbb{P}(\mathcal{E}|S) \geq 1 - \delta$$

Proof.

$$1 - \mathbb{P}(\mathcal{E}|S) \leq \sum_{T=1}^{\infty}\left[\frac{6\delta}{\pi^2(n+1)T^2} + \sum_{i=1}^{n}\frac{6\delta}{\pi^2(n+1)T^2}\right]$$
$$= \frac{6}{\pi^2}\sum_{T=1}^{\infty}\frac{\delta}{T^2} = \delta$$

Based on this, it is straightforward to prove Theorem 4.1.

4.5.2 Confidence Intervals of Bounded Reward Distributions

In this subsection, we show an instantiation of confidence interval when all the reward distributions are bounded. Without loss of generality, suppose they are bounded in $[a, b]$ and $R = b - a$.

We start by a direct application of Hoeffding's inequality to depict the concentration of each arm's expectation estimate.

Lemma 4.2 (Hoeffding)**.** *The probability that the difference between $\hat{y}_i$ and y_i is larger than a given constant t can be bounded as:*

$$\mathbb{P}(|\hat{y}_i - y_i| \geq t) \leq 2\exp\left(\frac{-2m_i t^2}{R^2}\right) \tag{4.8}$$

We then apply McDiarmid's inequality to describe the concentration of the threshold function estimator.

Lemma 4.3. *The probability that the difference between the estimated threshold function and the expected estimation of the value of the threshold function is larger than a given constant t can be bounded as:*

$$\mathbb{P}(|\hat{\theta} - \mathbb{E}\hat{\theta}| \geq t) \leq 2\exp\left(\frac{-2h(\mathbf{m})t^2}{R^2 g(k)}\right) \tag{4.9}$$

where $g(k) = (1 + k\sqrt{n-1})^2/n$.

Proof. Consider $\hat{\mu}_y$ to be a function of all the samples $\hat{\mu}_y(\mathbf{x}_1, \cdots, \mathbf{x}_n)$. Since

$$\hat{\mu}_y(\mathbf{x}_1, \cdots, \mathbf{x}_n) = \frac{1}{n}\sum_i \sum_j \frac{x_i^{(j)}}{m_i} \tag{4.10}$$

Hence

$$\sup_{x_i^{(j)}} \hat{\mu}_y(\mathbf{x}_1, \cdots, \mathbf{x}_n) - \inf_{x_i^{(j)}} \hat{\mu}_y(\mathbf{x}_1, \cdots, \mathbf{x}_n) \leq \frac{b-a}{nm_i} = \frac{R}{nm_i} \tag{4.11}$$

Consider $\hat{\sigma}_y$ to be a function of all the samples $\hat{\sigma}_y(\mathbf{x}_1, \cdots, \mathbf{x}_n)$. For any given pair of i and j, let

$$A_i = \frac{1}{n-1} \sum_{i' \neq i} \hat{y}_{i'}^2$$

$$B_i = \frac{1}{n-1} \sum_{i' \neq i} \hat{y}_{i'}$$

$$M_{ij} = \frac{1}{m_i - 1} \sum_{j' \neq j} x_i^{(j)}$$

where B_i is the mean of all the estimated $\hat{y}_{i'}$ other than $\hat{y}_{i'}$; $(A_i - B_i)^2$ is their standard deviation and therefore is no less than 0; M_{ij} is the mean of all but the j-th samples from the i-th arm.

We represent the estimated standard deviation $\hat{\sigma}_y$ as a function of random variable $x_i^{(j)}$ and the variables above:

$$\begin{aligned}
\hat{\sigma}_y(\mathbf{x}_1, \cdots, \mathbf{x}_n) &= \sqrt{\frac{\sum_i \hat{y}_i^2}{n} - \left(\frac{\sum_i \hat{y}_i}{n}\right)^2} \\
&= \sqrt{\frac{A_i(n-1) + \hat{y}_i^2}{n} - \left(\frac{B_i(n-1) + \hat{y}_i}{n}\right)^2} \\
&= \sqrt{\left(1 - \frac{1}{n}\right)\left(\frac{1}{n}(\hat{y}_i - B_i)^2 + (A_i - B_i^2)\right)} \\
&= \sqrt{1 - \frac{1}{n}} \sqrt{\frac{1}{n}\left(\frac{(m_i - 1)M_{ij} + x_i^{(j)}}{m_i} - B_i\right)^2 + (A_i - B_i^2)}
\end{aligned}$$

For any given $\mathbf{x}$, we want to bound the maximum possible difference of this function by keeping all variables fixed but only adjusting $x_i^{(j)}$

Let x_1, x_2 be the value of x_i^j when $\hat{\sigma}_y$ has the maximum and minimum value respectively.

For any given $\mathbf{x}_1 \cdots, \mathbf{x}_n$ except x_i^j, we have,

$$
\begin{aligned}
&\sup_{x_i^{(j)}} \hat{\sigma}_y(\mathbf{x}_1, \cdots, \mathbf{x}_n) - \inf_{x_i^{(j)}} \hat{\sigma}_y(\mathbf{x}_1, \cdots, \mathbf{x}_n) \\
\leq &\sqrt{1-\frac{1}{n}}\left[\sqrt{\frac{1}{n}\left(\frac{(m_i-1)M_{ij}+x_1}{m_i} - B_i\right)^2} - \sqrt{\frac{1}{n}\left(\frac{(m_i-1)M_{ij}+x_2}{m_i} - B_i\right)^2}\right] \\
\leq &\sqrt{\frac{1}{n}\left(1-\frac{1}{n}\right)}\left[\left|\frac{(m_i-1)M_{ij}+x_1}{m_i} - B_i\right| - \left|\frac{(m_i-1)M_{ij}+x_2}{m_i} - B_i\right|\right] \\
\leq &\sqrt{\frac{1}{n}\left(1-\frac{1}{n}\right)}\left(\frac{x_1-x_2}{m_i}\right) \\
\leq &\frac{b-a}{nm_i}\sqrt{n-1} = \frac{R}{nm_i}\sqrt{n-1}
\end{aligned}
$$

Since the threshold function is a linear combination of $\hat{\mu}_y$ and $\hat{\sigma}_y$, it also has bounded difference:

$$
\begin{aligned}
&\sup_{x_i^{(j)}} \hat{\theta}(\mathbf{x}_1, \cdots, \mathbf{x}_n) - \inf_{x_i^{(j)}} \hat{\theta}(\mathbf{x}_1, \cdots, \mathbf{x}_n) \\
&\leq \sup_{x_i^{(j)}} \hat{\mu}(\mathbf{x}_1, \cdots, \mathbf{x}_n) - \inf_{x_i^{(j)}} \hat{\mu}(\mathbf{x}_1, \cdots, \mathbf{x}_n) + \sup_{x_i^{(j)}} k\hat{\sigma}(\mathbf{x}_1, \cdots, \mathbf{x}_n) - \inf_{x_i^{(j)}} k\hat{\sigma}(\mathbf{x}_1, \cdots, \mathbf{x}_n) \\
&\leq \frac{R}{nm_i}\big(1 + k\sqrt{n-1}\big)
\end{aligned} \tag{4.12}
$$

Let $c_{ij} = \frac{R}{nm_i}\big(1 + k\sqrt{n-1}\big)$. We can have

$$
\begin{aligned}
\sum_i \sum_j c_{ij}^2 &= \sum_i m_i \frac{R^2}{n^2 m_i^2}\big(1 + k\sqrt{n-1}\big)^2 \\
&= \big(1 + k\sqrt{n-1}\big)^2 \frac{R^2}{n^2}\sum_i \frac{1}{m_i}
\end{aligned}
$$

According to McDiarmid's inequality, we can have the following concentration guarantee:

$$
\begin{aligned}
\mathbb{P}(|\hat{\theta} - \mathbb{E}\hat{\theta}| \geq t) \leq 2\exp\left(\frac{-2t^2}{\sum_i \sum_j c_{ij}^2}\right) &= 2\exp\left(\frac{-2n^2t^2}{R^2\big(1 + k\sqrt{n-1}\big)^2 \sum_i \frac{1}{m_i}}\right) \\
&= 2\exp\left(\frac{-2h(\mathbf{m})t^2}{R^2 g(k)}\right)
\end{aligned} \tag{4.13}
$$

where $g(k) = (1 + k\sqrt{n-1})^2/n$.

And since $\mathbb{E}\hat{\theta}$ is a biased estimator, we also need to bound its bias $|\mathbb{E}\hat{\theta} - \theta|$.

Lemma 4.4. *The bias of the threshold function estimator can be bounded as:*

$$\left|\mathbb{E}\hat{\theta} - \theta\right| \leq kR\sqrt{\frac{n-1}{4n}\frac{1}{h(\mathbf{m})}} \tag{4.14}$$

Proof. Notice that

$$\mathbb{E}\hat{\theta} - \theta = \mathbb{E}\hat{\mu}_y - \mu_y + k\mathbb{E}\hat{\sigma}_y - k\sigma_y = k\Big(\mathbb{E}\hat{\sigma}_y - \sigma_y\Big)$$

We only need to upper bound $\left|\mathbb{E}\hat{\sigma}_y - \sigma_y\right|$. Let $\hat{e}_i = \hat{y}_i - y_i$, and $\hat{\mu}_e = \hat{\mu}_y - \mu_y$, we have:

$$\begin{aligned}
\mathbb{E}\hat{\sigma}_y &= \mathbb{E}\sqrt{\frac{1}{n}\sum_i \Big(y_i - \mu_y + \hat{e}_i - \hat{\mu}_e\Big)^2} \\
&= \mathbb{E}\sqrt{\frac{1}{n}\left[\sum_i (y_i - \mu_y)^2 + \sum_i 2(y_i - \mu_y)(\hat{e}_i - \hat{\mu}_e) + \sum_i (\hat{e}_i - \hat{\mu}_e)^2\right]}
\end{aligned}$$

According to the CauchySchwarz inequality,

$$\left|\sum_i (y_i - \mu_y)(\hat{e}_i - \hat{\mu}_e)\right| \leq \sqrt{\sum_i (y_i - \mu_y)^2} \cdot \sqrt{\sum_i (\hat{e}_i - \hat{\mu}_e)^2}$$

Therefore, we can obtain the upper bound for $\mathbb{E}\hat{\sigma}_y$ as

$$\begin{aligned}
\mathbb{E}\hat{\sigma}_y &\leq \mathbb{E}\sqrt{\frac{1}{n}\left[\sum_i (y_i - \mu_y)^2 + 2\sqrt{\sum_i (y_i - \mu_y)^2} \cdot \sqrt{\sum_i (\hat{e}_i - \hat{\mu}_e)^2} + \sum_i (\hat{e}_i - \hat{\mu}_e)^2\right]} \\
&= \mathbb{E}\sqrt{\frac{1}{n}\left(\sqrt{\sum_i (y_i - \mu_y)^2} + \sqrt{\sum_i (\hat{e}_i - \hat{\mu}_e)^2}\right)^2} \\
&= \sqrt{\frac{1}{n}\sum_i (y_i - \mu_y)^2} + \mathbb{E}\sqrt{\frac{1}{n}\sum_i (\hat{e}_i - \hat{\mu}_e)^2} \\
&= \sigma_y + \mathbb{E}\sqrt{\frac{1}{n}\sum_i (\hat{e}_i - \hat{\mu}_e)^2}
\end{aligned}$$

Symmetrically, we can obtain the lower bound for $\mathbb{E}\hat{\sigma}_y$ as

$$\mathbb{E}\hat{\sigma}_y \geq \sigma_y - \mathbb{E}\sqrt{\frac{1}{n}\sum_i (\hat{e}_i - \hat{\mu}_e)^2}$$

Therefore, we only need to bound the term $\mathbb{E}\sqrt{\frac{1}{n}\sum_i (\hat{e}_i - \hat{\mu}_e)^2}$. By applying Jensen's inequality:

$$\begin{aligned}\mathbb{E}\sqrt{\frac{1}{n}\sum_i (\hat{e}_i - \hat{\mu}_e)^2} &\leq \sqrt{\frac{1}{n}\sum_i \mathbb{E}(\hat{e}_i - \hat{\mu}_e)^2} \\ &= \sqrt{\frac{1}{n}\sum_i \mathbb{E}(\hat{y}_i - y_i)^2 - \mathbb{E}\Big(\sum_i \frac{\hat{y}_i - y_i}{n}\Big)^2}\end{aligned}$$

Assuming the variance of the i-th arm's distribution is σ_i^2, when there are m_i samples collected to estimate $\hat{y}_i$, we have $\mathbb{E}[\hat{y}_i^2] = y_i^2 + \frac{\sigma_i^2}{m_i}$. Therefore, the equation becomes

$$\sqrt{\frac{1}{n}\sum_i \mathbb{E}(\hat{e}_i - \hat{\mu}_e)^2} = \sqrt{\frac{n-1}{n^2}\sum_i \frac{\sigma_i^2}{m_i}}$$

Further, since we have assume all the distributions are bounded, we can apply Popoviciu's inequality on variances, which states $\sigma_i^2 \leq R^2/4$. Hence,

$$\sqrt{\frac{n-1}{n^2}\sum_i \frac{\sigma_i^2}{m_i}} \leq R\sqrt{\frac{n-1}{4n}\frac{1}{h(\mathbf{m})}}$$

Combining the above equations, we have

$$\sigma_y - R\sqrt{\frac{n-1}{4n}\frac{1}{h(\mathbf{m})}} \leq \mathbb{E}\hat{\sigma}_y \leq \sigma_y + R\sqrt{\frac{n-1}{4n}\frac{1}{h(\mathbf{m})}} \tag{4.15}$$

From Lemma 4.3, we can derive with probability $1 - \delta'$,

$$|\hat{\theta} - \mathbb{E}\hat{\theta}| \leq R\sqrt{\frac{l(k)}{2h(\mathbf{m})}\log\Big(\frac{1}{\delta'}\Big)}$$

From Lemma 4.4, we have

$$|\mathbb{E}\hat{\theta} - \theta| \leq kR\sqrt{\frac{n-1}{4n}\frac{1}{h(\mathbf{m})}}$$

Combining these two equations, we have (with probability $1 - \delta'$)

$$\begin{aligned}
|\hat{\theta} - \theta| &\leq |\hat{\theta} - \mathbb{E}\hat{\theta}| + |\mathbb{E}\hat{\theta} - \theta| \\
&= R\sqrt{\frac{g(k)}{2h(\mathbf{m})}\log\left(\frac{1}{\delta'}\right)} + kR\sqrt{\frac{n-1}{4n}\frac{1}{h(\mathbf{m})}} \\
&\leq R\sqrt{\frac{1}{2h(\mathbf{m})}\log\left(\frac{1}{\delta'}\right)}\left(\sqrt{g(k)} + \sqrt{\frac{k^2}{2\log\left(\frac{1}{\delta'}\right)}}\right)
\end{aligned}$$

In our algorithm, the error δ' is set according to a function $\delta'(T)$ to guarantee the validity of the confidence interval through the entire sampling process. According to Eq. (4.1), we substitute its definition into the inequality above:

$$|\hat{\theta} - \theta| \leq R\sqrt{\frac{1}{2h(\mathbf{m})}\log\left(\frac{1}{\delta'(T)}\right)}\left(\sqrt{g(k)} + \sqrt{\frac{k^2}{2\log\left(\frac{\pi^2(n+1)T^2}{3\delta}\right)}}\right)$$

Notice that we pull each arm at least once as initialization, so $T \geq n$. Therefore, we can further obtain:

$$\begin{aligned}
|\hat{\theta} - \theta| &\leq R\sqrt{\frac{1}{2h(\mathbf{m})}\log\left(\frac{1}{\delta'(T)}\right)}\left(\sqrt{g(k)} + \sqrt{\frac{k^2}{2\log\left(\frac{\pi^2 n^3}{3\delta}\right)}}\right) \\
&\leq R\sqrt{\frac{l(k)}{2h(\mathbf{m})}\log\left(\frac{1}{\delta'(T)}\right)}
\end{aligned}$$

with probability $1 - \delta'$.

4.5.3 Upper Bound of Round-Robin Algorithm (Theorem 4.2)

We start by a lemma indicating how small the confidence intervals should be could we guarantee a certain arm is inactive.

Lemma 4.5. *At the T-th iteration, if $\mathcal{E}$ happens, and arm i is active, then*

$$\beta_i + \beta_\theta \geq \frac{1}{2}\Delta_i \tag{4.16}$$

where $\Delta_i = |y_i - \theta|$.

Proof. If arm i is active, then according to the algorithm, it satisfies the condition that

$$\begin{cases} \hat{y}_i - \beta_i < \hat{\theta} + \beta_\theta, & \text{if } \hat{y}_i > \hat{\theta}; \\ \hat{y}_i + \beta_i > \hat{\theta} - \beta_\theta, & \text{otherwise.} \end{cases} \tag{4.17}$$

Furthermore, if event $\mathcal{E}$ happens, we should have

$$\begin{aligned} \hat{y}_i - \beta_i \leq y_i \leq \hat{y}_i + \beta_i \\ \hat{\theta} - \beta_\theta \leq \theta \leq \hat{\theta} + \beta_\theta \end{aligned}$$

If $\hat{y}_i > \hat{\theta}$, then we have

$$\begin{aligned} y_i - \theta &\leq \hat{y}_i + \beta_i - (\hat{\theta} - \beta_\theta) \\ &= \hat{y}_i - \hat{\theta} + \beta_i + \beta_\theta \\ &\leq 2\beta_i + 2\beta_\theta \end{aligned} \tag{4.18}$$

Hence,

$$\beta_i + \beta_\theta \geq \frac{1}{2}\Delta_i \tag{4.19}$$

Symmetrically, for $\hat{y}_i \leq \hat{\theta}$ we can also obtain the same result.

Now we can give a proof of Theorem 4.2.

Proof. In Round-Robin algorithm, at any iteration, we have

$$|m_i - m_j| \leq 1, \forall 1 \leq i, j \leq n \tag{4.20}$$

where i and j are integers. According to Lemma 4.5, if $\mathcal{E}$ happens and arm i is still active, we have

$$\beta_i + \beta_\theta \geq \frac{1}{2}\Delta_i \tag{4.21}$$

Substituting the confidence intervals by their definitions,

$$\begin{aligned}
\frac{1}{2}\Delta_i &\leq R\sqrt{\frac{1}{2m_i}\log\left(\frac{1}{\delta'(T)}\right)} + R\sqrt{\frac{l(k)}{2h(\mathbf{m})}\log\left(\frac{1}{\delta'(T)}\right)} \\
\Delta_i &\leq R\sqrt{2\log\left(\frac{1}{\delta'(T)}\right)}\left(\sqrt{\frac{1}{m_i}} + \sqrt{\frac{l(k)}{h(\mathbf{m})}}\right) \\
&\leq R\sqrt{2\log\left(\frac{1}{\delta'(T)}\right)}\left(\sqrt{\frac{1}{m^*}} + \sqrt{\frac{l(k)}{m^*}}\right) \\
&\leq R\sqrt{2\log\left(\frac{1}{\delta'(T)}\right)}\left(\sqrt{\frac{1}{m^*}} + \sqrt{\frac{l(k)}{m^*}}\right) \qquad (4.22)
\end{aligned}$$

where $m^* = \min_{i'} m_{i'}$. By organizing the above inequality, we can obtain

$$m^* \leq \frac{2R^2}{\Delta_i^2}\log\left(\frac{1}{\delta'(T)}\right)\left(1 + \sqrt{l(k)}\right)^2 \qquad (4.23)$$

Hence, if the algorithm is not yet terminated, then there must be at least an arm i where the condition above holds.

And since in Round-Robin algorithm, for any i at any time we have $m_i \leq m^* + 1$, we can obtain

$$T = \sum_i m_i \leq n(m^* + 1) \qquad (4.24)$$

We analyze the situation when the algorithm is about to terminate. Right before the algorithm's last pull, denote the minimum number of pulls of a certain arm as $\tilde{m}^*$. Notice that the algorithm is not yet terminated, so we still have

$$\begin{aligned}
\tilde{m}^* &\leq \frac{2R^2}{\Delta_i^2}\log\left(\frac{1}{\delta'(T)}\right)\left(1 + \sqrt{l(k)}\right)^2 \\
&\leq \frac{2R^2}{\Delta_{i^*}^2}\log\left(\frac{1}{\delta'(T)}\right)\left(1 + \sqrt{l(k)}\right)^2
\end{aligned}$$

where $\Delta_{i^*} = \min_{i'} \Delta_{i'}$.

After the last pull, the minimum number of pulls of a certain arm $m^* \leq \tilde{m}^* + 1$. So we

have

$$\begin{aligned} T &\le n(m^*+1) \le n(\tilde{m}^*+2) \\ &\le \frac{2nR^2}{\Delta_{i^*}^2}\log\left(\frac{\pi^2(n+1)T^2}{3\delta}\right)\left(1+\sqrt{l(k)}\right)^2 + 2n \\ &= 2\mathbf{H}_1 R^2 \log\left(\frac{\pi^2(n+1)T^2}{3\delta}\right)\left(1+\sqrt{l(k)}\right)^2 + 2n \end{aligned}$$

According to Lemma 8 in [8], we have

$$T \le 8R^2\mathbf{H}_{\mathrm{RR}}\left[\log\left(\frac{4R^2\pi^2(n+1)\mathbf{H}_{\mathrm{RR}}}{3\delta}\right)+1\right] + 4n \tag{4.25}$$

4.5.4 Upper Bound of Weighted Round-Robin Algorithm (Theorem 4.3)

Lemma 4.6. *Suppose after T iterations of the algorithm WRR, the $(T+1)$-th iteration will be a regular pull. If random event $\mathcal{E}$ happens, and arm i has no less than T_i' additional pulls, then arm i is not in active set A ($i \notin A$). T_i' is defined as:*

$$T_i' = \frac{\rho-1}{\rho}\frac{2R^2}{\Delta_i^2}\log\left(\frac{1}{\delta'(T)}\right)\left(1+\sqrt{l(k)\rho}\right)^2 \tag{4.26}$$

where $\Delta_i = |y_i - \mathbb{E}\hat{\theta}|$.

Proof. According to Lemma 4.5, if arm i is active and $\mathcal{E}$ happens, we have $\beta_i + \beta_\theta \ge \Delta_i/2$. By substituting the confidence intervals by their definitions, we have:

$$\begin{aligned} \frac{1}{2}\Delta_i &\le R\sqrt{\frac{1}{2m_i}\log\left(\frac{1}{\delta'(T)}\right)} + R\sqrt{\frac{l(k)}{2h(\mathbf{m})}\log\left(\frac{1}{\delta'(T)}\right)} \\ \Delta_i &\le R\sqrt{2\log\left(\frac{1}{\delta'(T)}\right)}\left(\sqrt{\frac{1}{m_i}}+\sqrt{\frac{l(k)}{h(\mathbf{m})}}\right) \\ &\le R\sqrt{2\log\left(\frac{1}{\delta'(T)}\right)}\left(\sqrt{\frac{1}{m_i}}+\sqrt{\frac{l(k)}{m^*}}\right) \end{aligned} \tag{4.27}$$

where $m^* = \min_{i'} m_{i'}$. Among the m_i pulls for arm i, suppose $m_{i,r}$ are regular pulls, and $m_{i,a}$ are additional pulls, $m_{i,r} + m_{i,a} = m_i$. According to the algorithm, when the flag of

additional pull is cleared, and if arm i is still in the active set, then there must be

$$\rho(m_{i,r}-1) \le m_i = m_{i,r} + m_{i,a} < \rho(m_{i,r}-1) + 1 \tag{4.28}$$

And since the regular pulls follow a round-robin strategy, we should have $m^* \ge m_{i,r} - 1$. Hence,

$$\begin{aligned}
\Delta_i &\le R\sqrt{2\log\left(\frac{1}{\delta'(T)}\right)}\left(\sqrt{\frac{1}{m_i}} + \sqrt{\frac{l(k)}{m_{i,r}-1}}\right) \\
&\le R\sqrt{2\log\left(\frac{1}{\delta'(T)}\right)}\left(\sqrt{\frac{1}{m_i}} + \sqrt{\frac{l(k)\rho}{m_i}}\right) \\
m_i &\le \frac{2R^2}{\Delta_i^2}\log\left(\frac{1}{\delta'(T)}\right)\left(1+\sqrt{l(k)\rho}\right)^2
\end{aligned}$$

From Eq. (4.28) we can have

$$m_{i,a} < \frac{\rho-1}{\rho} m_i \tag{4.29}$$

which leads to

$$m_{i,a} < \frac{\rho-1}{\rho}\frac{2R^2}{\Delta_i^2}\log\left(\frac{1}{\delta'(T)}\right)\left(1+\sqrt{l(k)\rho}\right)^2 = T_i' \tag{4.30}$$

This is a contradiction to the condition $m_{i,a} \ge T_i'$. Therefore arm i should not be pulled additionally in the next iteration.

Next we bound the number of regular pulls for each arm. For the convenience of analysis, we analyze the situation when the algorithm just finishes a "total iteration", when the next $(T+1)$-th iteration will be a regular pull for arm 1. In this case the number regular pulls for all arms will be the same, denoted as m_r. Let $i^* = \arg\min_i \Delta_i$, we have the following theorem.

Lemma 4.7. *Suppose after T iterations the $(T+1)$-th iteration will be a regular pull for arm 1. If the algorithm is not yet terminated and event $\mathcal{E}$ happens, then the number of regular pulls for any arms should be no more than T_r', where*

$$T_r' = \frac{1}{\rho}\frac{2R^2}{\Delta_{i^*}^2}\log\left(\frac{1}{\delta'(T)}\right)\left(1+\sqrt{l(k)\rho}\right)^2 + 1 \tag{4.31}$$

Proof. Since the algorithm is not yet terminated, then the active set of arms A is non-empty.

Suppose arm $i \in A$, then

$$m_i \leq \frac{2R^2}{\Delta_i^2} \log\left(\frac{1}{\delta'(T)}\right)\left(1+\sqrt{l(k)\rho}\right)^2$$

And since the $(T+1)$-th iteration will be a regular pull, the inequality of Eq. (4.28) stands. Hence,

$$\begin{aligned} m_r = m_{i,r} &\leq \frac{m_i}{\rho} + 1 \\ &\leq \frac{1}{\rho}\frac{2R^2}{\Delta_i^2} \log\left(\frac{1}{\delta'(T)}\right)\left(1+\sqrt{l(k)\rho}\right)^2 + 1 \\ &\leq \frac{1}{\rho}\frac{2R^2}{\Delta_{i^*}^2} \log\left(\frac{1}{\delta'(T)}\right)\left(1+\sqrt{l(k)\rho}\right)^2 + 1 \end{aligned}$$

Now we prove Theorem 4.3

Proof. According to Theorem 4.1, as long as event $\mathcal{E}$ happens, the returned results will be correct. And since for any possible pulling sequences, the probability of event $\mathcal{E}$ is no less than $(1-\delta)$, thus the returned result will be correct with probability $(1-\delta)$.

As for bounding the total number of pulls T. Notice that

$$T = \sum_i m_i = \sum_i m_{i,a} + \sum_i m_{i,r} \tag{4.32}$$

we bound the number of additional and regular pulls respectively.

We start by bounding the final number of additional pulls for each arm. For any arm i, when the algorithm terminates, denote the iteration of its last regular pull when it is in the active set as $T_{i,a}$. Then we can apply Lemma 4.6 to the $(T_{i,a}-1)$-th iteration, as well as the fact that the following last several additional pulls on arm i will not exceed ρ, thus

$$\begin{aligned} m_{i,a} &\leq \frac{\rho-1}{\rho}\frac{2R^2}{\Delta_i^2} \log\left(\frac{1}{\delta'(T_{i,a}-1)}\right)\left(1+\sqrt{l(k)\rho}\right)^2 + \rho \\ &\leq \frac{\rho-1}{\rho}\frac{2R^2}{\Delta_i^2} \log\left(\frac{1}{\delta'(T)}\right)\left(1+\sqrt{l(k)\rho}\right)^2 + \rho \end{aligned}$$

We then bound the final number of regular pulls for each arm. Since the regular pulls follow the round-robin strategy, we denote the iteration of the last regular pull on arm 1 as $T_{1,r}$. Then we apply Lemma 4.7 to the $(T_{1,r}-1)$-th iteration, and using the fact that

there will be no more than 1 more regular pulls for each arm before the algorithm terminates. Thereby we obtain

$$\begin{aligned} m_{r,a} &\leq \frac{1}{\rho}\frac{2R^2}{\Delta_{i^*}^2}\log\left(\frac{1}{\delta'(T_{1,r}-1)}\right)\left(1+\sqrt{l(k)\rho}\right)^2+2 \\ &\leq \frac{1}{\rho}\frac{2R^2}{\Delta_{i^*}^2}\log\left(\frac{1}{\delta'(T)}\right)\left(1+\sqrt{l(k)\rho}\right)^2+2 \end{aligned}$$

Thus, when the algorithm terminates, we should have

$$\begin{aligned} T &= \sum_i m_{i,a} + \sum_i m_{i,r} \\ &\leq \left(\frac{\mathbf{H}_1}{\rho}+\frac{(\rho-1)\mathbf{H}_2}{\rho}\right)2R^2\log\left(\frac{1}{\delta'(T)}\right)\left(1+\sqrt{l(k)\rho}\right)^2+(\rho+2)n \end{aligned}$$

Substituting $\delta'(T)$ by its definition, we have

$$\begin{aligned} T &\leq \left(\frac{\mathbf{H}_1}{\rho}+\frac{(\rho-1)\mathbf{H}_2}{\rho}\right)2R^2\log\left(\frac{\pi^2(n+1)T^2}{3\delta}\right)\left(1+\sqrt{l(k)\rho}\right)^2+(\rho+2)n \\ &= 2R^2\mathbf{H}_{\mathrm{WRR}}\log\left(\frac{\pi^2(n+1)T^2}{3\delta}\right)+(\rho+2)n \end{aligned}$$

Therefore, according to Lemma 8 in [8], we have

$$T \leq 8R^2\mathbf{H}_{\mathrm{WRR}}\left[\log\left(\frac{4R^2\pi^2(n+1)\mathbf{H}_{\mathrm{WRR}}}{3\delta}\right)+1\right]+2(\rho+2)n \tag{4.33}$$

4.5.5 Confidence Intervals of Bernoulli Reward Distributions

In many real applications, each arm returns a binary sample 0 or 1, drawn from a Bernoulli distribution. In this subsection, we show an heuristic instantiation of confidence interval when all the reward distributions are Bernoulli distributions. We optimize the confidence interval on this specific case with some approximation.

We leverage a confidence interval presented in [5], defined as

$$\beta_i(m_i,\delta') = z_{\delta'}\sqrt{\frac{\tilde{p}(1-\tilde{p})}{m_i}} \tag{4.34}$$

where

$$\tilde{p} = \frac{m_i^+ + \frac{z_{\delta'}^2}{2}}{m_i + z_{\delta'}^2}$$

m_i^+ is the number of samples that equal to 1 among m_i samples, and $z_{\delta'}$ is value of the *inverse error function*:

$$z_{\delta'} = \text{erf}^{-1}(1 - \delta')$$

As for the confidence interval of the outlier threshold, we simply apply the propagation of uncertainty, where

$$\begin{aligned}
\beta_\theta(\mathbf{m}, \delta') &= \sqrt{\sum_i \left(\frac{\partial \theta}{\partial y_i}\right)^2 \beta_i^2} \\
&= \sqrt{\sum_i \left(\frac{k y_i}{n\sqrt{\sigma_y}} + \frac{1}{n}\right)^2 \beta_i^2} \\
&\approx \sqrt{\sum_i \left(\frac{k \hat{y}_i}{n\sqrt{\hat{\sigma}_y}} + \frac{1}{n}\right)^2 \beta_i^2}
\end{aligned}$$

4.6 SUMMARY

In this chapter, we study a novel problem of identifying the outlier arms with extremely high/low reward expectations compared to other arms in a multi-armed bandit. We propose a Round-Robin algorithm and a Weighted Round-Robin algorithm with correctness guarantee. We also upper bound both algorithms when the reward distributions are bounded. We conduct experiments on both synthetic and real data to verify our algorithms.

A future direction worth exploring is to adapt the algorithm to identifying outlier text objects where the text object is represented by a multi-dimensional vector in the embedding space, instead of a single numerical value. Other further extensions of this work include deriving a lower bound of this problem or extending the problem to a PAC setting.

CHAPTER 5: WEAKLY-SUPERVISED TEXT MINING FOR ASPECT-BASED SENTIMENT ANALYSIS

5.1 OVERVIEW

With word embedding, some text mining applications that traditionally rely on massive labeled data can potentially be handled with much less supervision, such as some seed words with labels. The semantic proximity of words captured by word embedding provides the signals necessary for performing the task.

In this chapter, we study a typical task, which is sentiment analysis. Identifying and classifying sentiment within documents is one of the most important text mining tasks, empowering numerous applications in recommendations [12, 18], stock prediction [76, 65] *etc.*

Aspect-based sentiment analysis [74] provides a fine-grained view of sentiment within documents. It aims to extract different aspects of the entity being reviewed and to determine the sentiment corresponding to each aspect individually. For example, "*The hotel has a reasonable price but the room is small*" presents two aspects: "price" and "room", with positive and negative sentiment respectively.

While many studies on this task adopt supervised methods [47, 57, 99, 34, 28], aspect-level sentiment labels are usually expensive to obtain. This triggers another series of research effort on weakly or distant supervised aspect-based sentiment analysis. However, most of the previous work utilizes external language resource or tools such as thesaurus information [32, 63, 38, 55, 70] or dependency parser [71, 52]. In reality, such resource is not always available or accurate in new domains or low-resource languages.

Therefore, we propose to study aspect-based sentiment with minimal user guidance. The goal is to perform aspect-based sentiment analysis without direct or indirect usage of training data or external language resource, but only a massive target corpus and very little user effort. More concretely, users are only required to provide a small set of seed aspect words and a small set of seed sentiment words with sentiment polarity. The objective is to output identified aspect mentions from each review document, as well as their corresponding sentiment polarity (positive or negative).

Example 5.1. *Figure 5.1 shows an example. With a massive set of hotel reviews, users only need to provide a small set of seed aspect words like {"room", "price", ...} as well as a small set of seed sentiment words {"good", "terrible", ...}. Users also need to provide sentiment polarity labels for the seed sentiment words (e.g. "good" is positive and "terrible" is negative). For each review document, the algorithm should be able to identify aspect mentions within*

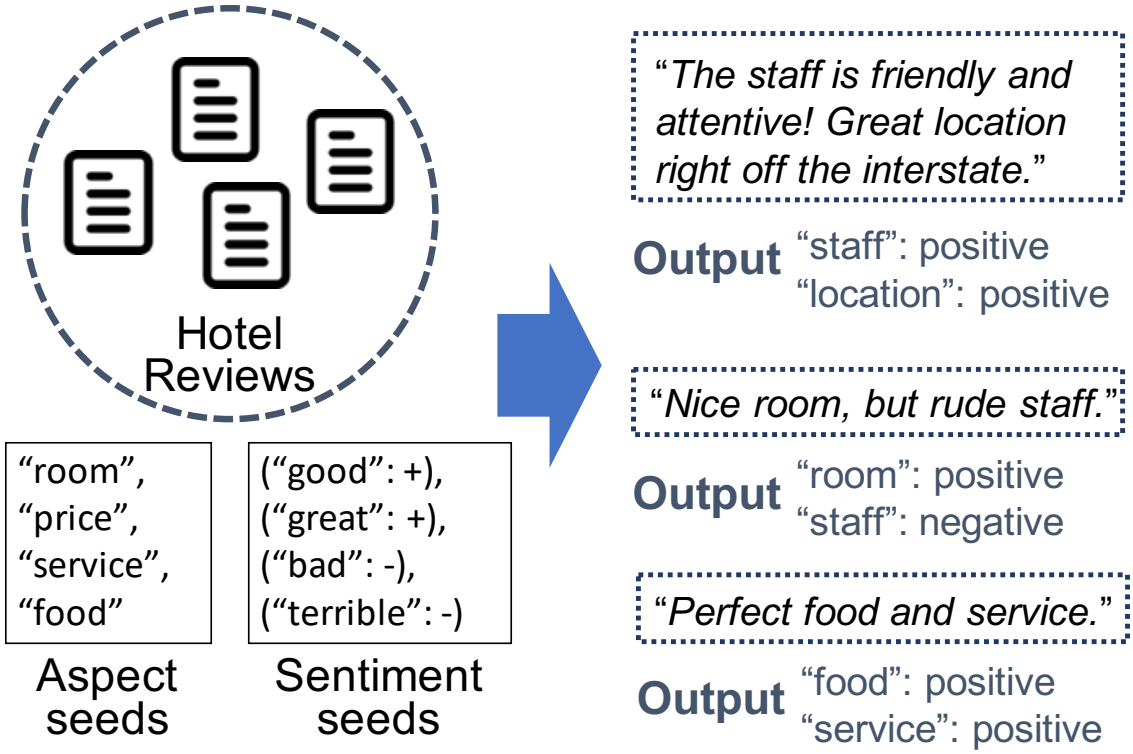


Figure 5.1: An example of aspect sentiment analysis with minimal guidance.

the document and output sentiment polarity label for each identified aspect mention. For example, for a review document "Nice room, but rude staff"*, the algorithm should output two identified aspect words "room" and "staff", even if "staff" is not a seed aspect word. The algorithm should also assigns positive label to "room" and negative label to "staff".*

This setting is usual when one needs to develop an aspect-based sentiment analysis tool for a new domain, such as online forums of a certain field. It is expensive to produce sufficient labeled data for training a supervised model. External language resource is usually only created for general field and thus does not necessarily have good coverage. Existing NLP tools are often inaccurate on specific domain without extra learning, especially for informal corpus like user generated content. However, our setting is much more realistic as users only need to specify around 10 seed aspect and sentiment words.

The challenges in this problem setting are: 1) how to leverage the limited user guidance to identify aspects as complete as possible; 2) how to classify sentiment for each aspect in a document with limited user guidance. The only signals we can leverage for both tasks are the user provided seed sets and the corpus.

We make the following contributions. First, we develop a method to expand the aspect and sentiment lexicons from given set of seed words based on features extracted by frequent pattern mining. Second, we propose a generative model to characterize the generation of

aspect and sentiment mentions, represented by their word embedding. By inferring the model, we can assign sentiment polarity to words in the sentiment lexicon. Finally, we can accordingly perform aspect-based sentiment analysis on each document. We verify the effectiveness on two real world data sets.

We present the detail of our work below.

5.2 PRELIMINARIES

In this section, we formalize the research problem, and briefly introduce the framework of our pipeline.

5.2.1 Problem formalization.

We denote a set of documents as $\mathcal{D} = \{d_i\}_{i=1}^n$. A document d_i consists of a sequence of sentence $d_i = (s_1, \cdots, s_{|d_i|})$. Each sentence s_j can be represented as a sequence of tokens $s_j = (w_1, \cdots, w_{|s_j|})$, where each w_k takes a value from a vocabulary $\mathcal{V}$.

For each word w in the vocabulary $\mathcal{V}$, one can derive an embedding vector from word embedding technique (*e.g.* [61]). Recall that word embedding provides a function $f : \mathcal{V} \mapsto \mathbb{R}^\nu$, where ν is the number of dimensions of the embedding space. The semantic proximity between two words can be reflected by the cosine similarity of their embedding vectors, defined in Eq. (3.1).

Notice that each $w \in \mathcal{V}$ is not necessarily a unigram word, but may also refer to a multigram phrase (*e.g.* "air conditioner", "mini bar"), or a subword like "n't" in "don't". We will use the term "word" to refer to any elements from the vocabulary $\mathcal{V}$ unless otherwise noted.

Each document d_i usually contains a few aspect mentions $\mathbf{a}_i = \{a_1, \cdots, a_{|\mathbf{a}_i|}\}$, where each aspect mention is a token in d_i. Each aspect mention is associated with a sentiment label, which can be represented as $\mathbf{y}_i = \{y_1, \cdots, y_{|\mathbf{a}_i|}\}$ collectively, where $y_k \in \mathcal{Y}$ represents the sentiment label for aspect mention a_k. In this chapter, we only focus on a binary setting, namely $\mathcal{Y} = \{0, 1\}$, where 0 stands for *negative* sentiment, and 1 stands for *positive* sentiment.

Users can provide guidance as seed aspect and sentiment words, which are essentially subsets of the vocabulary, denoted as $\mathcal{V}_A^{(0)}, \mathcal{V}_S^{(0)} \subset \mathcal{V}$ respectively. Moreover, for each seed sentiment word $w \in \mathcal{V}_S^{(0)}$, users also provide its sentiment label, denoted as $r_0(w) \in \mathcal{Y}$.

The problem can be formalized as:

Problem 5.1. *Given a corpus $\mathcal{D}$, a small set of seed aspect words $\mathcal{V}_A^{(0)}$ and a small set of seed sentiment words $\mathcal{V}_S^{(0)}$ with their labels $r_0 : \mathcal{V}_S^{(0)} \mapsto \mathcal{Y}$, we aim to identify the aspect mentions*

$\mathbf{a}_i$ *as well as their sentiment labels* $\mathbf{y}_i$ *for each* $d_i \in \mathcal{D}$.

Notice that there are studies in which an aspect is defined as a set or a distribution of aspect words. Although our output is only formalized as aspect mentions, we can always perform a clustering method to aggregate aspect words into more abstract aspects. However, this is beyond the scope of our study and we would address this in our future work.

5.2.2 Framework.

We tackle the problem in two steps. First, we expand the aspect and sentiment lexicons from very small sets of seed words. Second, we identify the aspect and sentiment mentions in each document and classify the sentiment polarity.

Lexicon expansion. Aspect and sentiment lexicons serve as strong tools to identify the most essential signals for aspect-based sentiment analysis.

For a given domain of reviews, the aspect lexicon $\mathcal{V}_A \subset \mathcal{V}$ should contain all the words characterizing possible factors of the entity to be reviewed, such as "location", "price", "service" in hotel reviews.

The sentiment lexicon $\mathcal{V}_S \subset \mathcal{V}$ should have all the words that express or imply an attitude or emotion. General sentiment words include "good", "great", "terrible", while some sentiment words can also be domain dependent. For example, words like "renovated", "air conditioned" imply positive sentiment in hotel reviews but not necessarily in other domains.

The goal of this stage is to construct the aspect lexicon $\mathcal{V}_A$ and the sentiment lexicon $\mathcal{V}_S$ from a given review corpus $\mathcal{D}$ as precise and complete as possible merely from the seeds $\mathcal{V}_A^{(0)}$ and $\mathcal{V}_S^{(0)}$.

Sentiment classification. With the aspect and sentiment lexicons available, we can identify aspect and corresponding sentiment mentions from documents. However, we do not have sentiment polarity labels for most of the sentiment words in $\mathcal{V}_S$. Hence, we need to perform sentiment polarity assignment before we can conduct sentiment classification of documents.

The objective is to derive a mapping function $r : \mathcal{V}_S \mapsto [0, 1]$, which assigns sentiment polarity rating $r(w)$ for all the words $w \in \mathcal{V}_S$. Again, only sentiment words in the seed set $\mathcal{V}_S^{(0)}$ are given polarity labels as input by $r_0(\cdot)$, while other sentiment words mined from the previous stage remain unlabeled.

Once we obtain the sentiment polarity for the entire sentiment lexicon, we can generate sentiment labels $\mathbf{y}_i$ for aspect mentions $\mathbf{a}_i$ in each document d_i.

5.3 LEXICON EXPANSION

In this section, we introduce a method based on frequent pattern mining to expand aspect and sentiment lexicons from given seed words $\mathcal{V}_A^{(0)}$ and $\mathcal{V}_S^{(0)}$.

The general idea is to select a few context patterns as features to characterize each word w. Then, we can train classifiers to extract new aspect and sentiment words iteratively.

There are three modules of our method:

- *Pattern feature mining.* Mining and selecting pattern features that can effectively characterize aspect and sentiment words.
- *Aspect lexicon expansion.* Using pattern features and currently mined sentiment lexicon to expand aspect lexicon.
- *Sentiment lexicon expansion.* Using pattern features and currently mined aspect lexicon to expand sentiment lexicon.

Notice that our method does not rely on NLP parsing tools such as Part of Speech (PoS) tagging or dependency parsing. Although NLP tools may provide strong syntactic signals in this task, they are usually trained on a more general corpus and often suffer from higher error rate on user generated review data. Furthermore, applying NLP pipeline is much slower than frequent pattern mining.

We will introduce each module in detail.

5.3.1 Pattern feature mining.

An important category of signals to characterize whether a word w is in aspect/sentiment lexicon are the frequencies of specific context patterns around w. For example, if the pattern "*the w is great*" occurs sufficiently frequent, w will probably be in the aspect lexicon. Similarly, if the pattern "*a very w bed*" is frequent, then w should be in the sentiment lexicon.

Precisely, a pattern feature p is an ordered sequence of tokens or aliases, represented as $p = (t_1, \cdots, t_l)$, where t_i is either a word from $\mathcal{V}$ or an alias that could be substituted by any from a set of words. In our setting, possible aliases are "[aspect]" or "[sentiment]", which can be substituted by any currently known aspect or sentiment words respectively.

It is intractable to enumerate all the possible context patterns of a word w, and most of them are not necessarily informative in determining whether w is in the aspect/sentiment lexicon. Apparently, patterns such as "a w" or "w of" are not very useful. Therefore, we adopt the following mechanism to mine and select informative pattern features, denoted as $P = \{p_1, p_2, \cdots\}$.

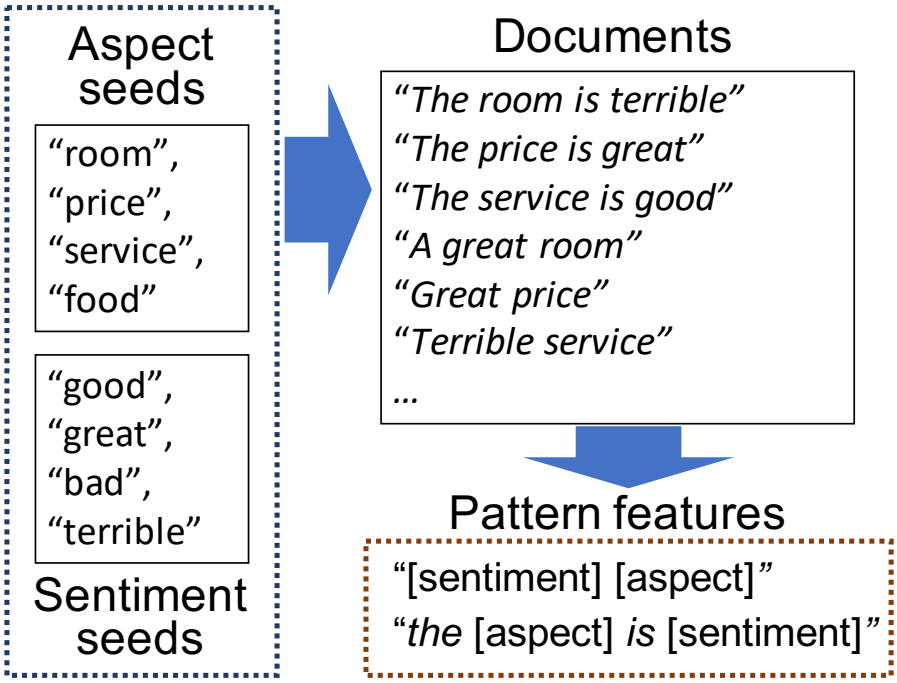


Figure 5.2: An example of mining pattern features.

Frequency. Intuitively, frequent patterns containing both (currently known) aspect and sentiment words would be a good candidate pool.

In order to mine such frequent patterns, we build a subset of sentences $\mathcal{S}$ containing both aspect and sentiment words from the seed set. As shown in Figure 5.2, only sentences containing words from both the aspect seeds and the sentiment seeds are selected.

Moreover, we treat all aspect (sentiment) words as a unified alias ("[aspect]", "[sentiment]") and turn all the words into lower case so that similar patterns can be merged together. As an example in Figure 5.2, the first three sentences "*The room is terrible*", "*The price is great*" and "*The service is good*" would all be converted into the same pattern "the [aspect] is [sentiment]".

We then perform a contiguous sequential frequent pattern mining algorithm based on the derived set $\mathcal{S}$. The relative minimum support is set to $\theta = 0.005$, namely only patterns with frequency larger than or equal to $\theta|\mathcal{S}|$ would be mined.

Representativeness. Some frequent patterns do not contain any aspect or sentiment aliases (*e.g.* "is a"), which cannot serve as pattern features in the following modules. Therefore, we only keep the patterns containing both a sentiment alias and an aspect alias.

Patterns crossing multiple clauses usually do not serve as good features. Sometimes patterns like "[sentiment], [aspect] is" would be mined as frequent patterns while they do

not necessarily reflect any interplay between the aspect and the sentiment word. We simply remove any patterns containing non-alphabetic tokens.

Concordance. Sometimes a pattern is frequent only because it has a frequent sub-pattern and does not necessarily provide novel information. For example, "[sentiment] [aspect]" is frequent and informative, but "[sentiment] [aspect] on" may be redundant even if it is frequent. The latter becomes frequent only because the former is so frequent that a co-occurred random word would produce another frequent pattern.

Therefore, we perform a test of independence to filter such redundant patterns. For each mined frequent pattern p, we test against the null hypothesis that it is merely generated randomly by attaching a word w to its immediate sub-pattern p'. By "immediate", we mean p and p' only differs by one word at the boundary, namely $p = p' \oplus w$ or $p = w \oplus p'$, where $\oplus$ means concatenation. We calculate a z-score as a test of independence [59]:

$$z(p, p') = \frac{c(p) - c(p')\mathbb{P}(w)}{\sqrt{c(p')\mathbb{P}(w)(1 - \mathbb{P}(w))}} \tag{5.1}$$

where $\mathbb{P}(w)$ is the relative frequency of w in $\mathcal{D}$.

We filter pattern p and all of its super-patterns if there exists any sub-pattern p' such that $z < \Phi^{-1}(1-\alpha)$ where Φ^{-1} is the probit function (*i.e.* inverse cumulative distribution function of a standard Gaussian distribution). We set $\alpha = 10^{-3}$ in our experiments.

Only patterns satisfying all the above criteria will be selected as pattern features in P.

5.3.2 Aspect/Sentiment Lexicon Expansion.

The basic idea of this module is to utilize the mined pattern features P to build a classifier to determine if a word is an aspect/sentiment word. We can then run the aspect lexicon expansion and sentiment lexicon expansion iteratively until convergence.

Both lexicons can be expanded in a similar mechanism. We only focus on describing aspect lexicon expansion below, while the sentiment lexicon expansion is symmetric.

Suppose we are at the t-th iteration and with mined aspect and sentiment lexicons $\mathcal{V}_A^{(t)}$ and $\mathcal{V}_S^{(t)}$. Our first step is to extract a set of candidate aspect words $\mathcal{U}_A^{(t+1)}$, where each candidate aspect word occurs at least once with a pattern feature $p_i \in P$, *i.e.* $\exists p_i \in P, c(p_i(w)) > 0$, where $c(\cdot)$ counts the frequency in $\mathcal{D}$ and $p_i(w)$ is a pattern by substituting w into "[aspect]" ("[sentiment]" for sentiment lexicon expansion) in p_i.

Then we adopt a supervised classifier to determine which candidate words in $\mathcal{U}_A^{(t+1)}$ should be an aspect word. We build feature vectors $\mathbf{x}_j$ for each word w_j by counting relative

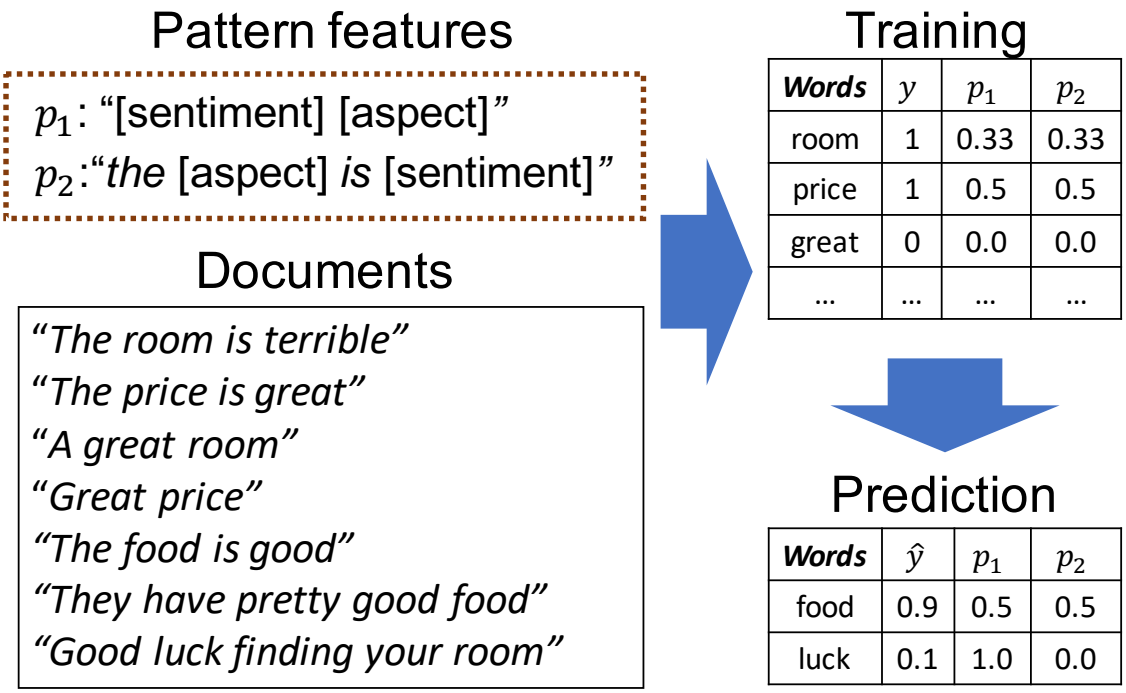


Figure 5.3: An example of expanding aspect lexicons from pattern features. Notice that y and $\hat{y}$ here represents whether a word is in the aspect lexicon. It is *not* its sentiment polarity label.

frequencies of all the patterns p_i in P with regard to w_j, namely $\mathbf{x}_j = [\frac{c(p_i(w_j))}{c(w_j)}]_{p_i \in P}$.

We treat words in $\mathcal{V}_A^{(t)}$ as positive samples and words in $\mathcal{V}_S^{(t)}$ as negative samples to train a random forest classifier. Then we apply the classifier to words in $\mathcal{U}_A^{(t+1)}$ to obtain the predicted value $\hat{Y}_A^{(t+1)}$, where $0 \leq \hat{y}_j \leq 1$ is the classifier's confidence value that candidate w_j is an aspect word.

Finally, we expand those candidate words w_j with $\hat{y}_j \geq \tau$ into the aspect lexicon $\mathcal{V}_A^{(t+1)}$, where $\tau = 0.8$ is a threshold. We also add candidate words w_j with embedding vectors close to any known aspect words $w_{j'} \in \mathcal{V}_A^{(t)}$ into the expanded lexicon, as long as $\text{sim}(f(w_j), f(w_{j'})) \geq \beta$ and $\hat{y}_j \geq 0.5$, where β is another threshold.

Example 5.2. *Figure 5.3 shows an example. With the given pattern features, we can first extract the candidate aspect words "food" and "luck" as they fit into the pattern features' aspect alias for at least once. We then build feature vectors for all the candidates and known aspect/sentiment words. We treat known aspect words ("room", "price", ...) as positive samples and sentiment words ("great", ...) as negative samples to train a random forest classifier. Finally, we can obtain the prediction results $\hat{y}$'s for candidate aspect words.*

The sentiment lexicon expansion module is symmetric to the aspect one.

5.3.3 Iterative expansion.

With a given set of pattern features P, we can perform aspect and sentiment lexicon expansion. Moreover, with the newly expanded aspect words, more candidate sentiment words can be extracted and values of pattern features for candidate sentiment words can be updated. This may trigger more sentiment words to be expanded and vice versa. Therefore, we iteratively perform aspect and sentiment word expansion until convergence.

5.4 ASPECT-BASED SENTIMENT CLASSIFICATION

In this section, we leverage the aspect and sentiment lexicons to perform aspect-based sentiment classification on documents.

The general idea is to utilize the embedding vector of sentiment words. First, we try to derive a "positive vector" and a "negative vector" in the semantic space modeled by word embedding based on the seed sentiment words and the corpus. Then, we can assign sentiment polarity to words in the sentiment lexicon $\mathcal{V}_S$ based on how close they are to the positive or the negative vector. Finally, polarity of sentiment mentions in each document can be aggregated to obtain sentiment classification results.

5.4.1 A simple baseline.

We start by introducing a relatively straightforward baseline.

Polarity assignment in sentiment lexicon. We introduce a naïve way to assign polarity labels to words in sentiment lexicon. Notice that for each word in the seed sentiment lexicon $w_j \in \mathcal{V}_S^{(0)}$, a polarity label $r_0(w_j) \in \{0, 1\}$ is given. We use $\mathcal{V}_{S+}^{(0)}$ and $\mathcal{V}_{S-}^{(0)}$ to represent subset of seed sentiment words with positive and negative polarity labels respectively.

We can derive a "positive vector" $\mathbf{v}_+$ and a "negative vector" $\mathbf{v}_-$ by taking the average embedding vectors of words with positive and negative labels respectively:

$$\mathbf{v}_+ = \frac{1}{|\mathcal{V}_{S+}^{(0)}|} \sum_{w_j \in \mathcal{V}_{S+}^{(0)}} f(w_j), \quad \mathbf{v}_- = \frac{1}{|\mathcal{V}_{S-}^{(0)}|} \sum_{w_j \in \mathcal{V}_{S-}^{(0)}} f(w_j)$$

where $f(w_j)$ represents the embedding vector of word w_j, as mentioned in Section 5.2.

Intuitively, words with embedding vector closer to $\mathbf{v}_+$ are more likely to convey positive sentiment, while words with embedding vector closer to $\mathbf{v}_-$ would be more likely negative. Therefore, we can assign sentiment polarity scores to all the other words in the sentiment

lexicon. Suppose for a word $w_j \in \mathcal{V}_S \setminus \mathcal{V}_S^{(0)}$, we can assign a sentiment rate $r(w_j) \in [0, 1]$:

$$r(w_j) = \varphi\big(\text{sim}(\mathbf{v}_+, f(w_j)) - \text{sim}(\mathbf{v}_-, f(w_j))\big)$$

where $\varphi(\cdot)$ is the standard logistic function. Meanwhile, if a word w_j is in seed sentiment words, it will still keep its ground-truth label $r(w_j) = r_0(w_j)$.

Based on the polarity scores of sentiment lexicon, we can further obtain the sentiment classification label of each document.

Aspect-based sentiment classification. For each document d_i, we extract all of the mentions of words in aspect lexicon, denoted as $\mathbf{a}_i = \{a_k \in d_i | a_k \in \mathcal{V}_A\}$. We also extract all of the sentiment mentions and map them into the closest aspect mention within the same sentence. Therefore, for the k-th aspect mention a_k, there is a set of sentiment mentions $\mathbf{o}_{ik}$. Aspect mentions with $|\mathbf{o}_{ik}| = 0$ are not included in practices.

We can aggregate the sentiment polarity scores for each aspect mentions $a_k \in \mathbf{a}_i$. We calculate y_k as the sentiment polarity score of the k-th aspect mention a_k in document d_i by averaging the sentiment polarity scores for all the words in $\mathbf{o}_{ik}$:

$$y_k = \mathbb{I}\Big[\frac{1}{|\mathbf{o}_{ik}|} \sum_{o_j \in \mathbf{o}_{ik}} r(o_j) > 0.5\Big]$$

where $\mathbb{I}(\cdot)$ is an indicator function.

Thereby we can obtain final output $\mathbf{a}_i$ and $\mathbf{y}_i = \{y_k\}_{k=1}^{|\mathbf{a}_i|}$ for document d_i.

5.4.2 Rectification of polarity assignment.

The positive and negative vectors $\mathbf{v}_+$, $\mathbf{v}_-$ utilized above are derived from a very small labeled lexicon. Hence, they may not be sufficiently accurate to reflect the actual distribution of positive and negative sentiment words in the embedding space.

We utilize a novel model to rectify their directions. We propose a graphical model to characterize the generation of aspect and sentiment words in review data, where the means of the two hidden sentiment word distributions corresponds to the "real" positive and negative vectors. By inferring the model, we can obtain more accurate positive and negative vectors.

Instead of using a mixture of multinomial distributions to generate words, we model the generation of each word's embedding vector directions by a mixture of von-Mises Fisher distributions. The von Mises-Fisher (vMF) distribution is a widely adopted distribution in directional statistics to model unit vectors in a spherical space and shows stronger

power [11, 97] than Gaussian in modeling embedding vectors in different applications. Its formalized definition can be found in Eq. (3.2).

Model. Our model assumes T hidden aspect vMF distributions and 2 hidden sentiment vMF distributions. For each document d_i, an aspect multinomial distribution θ_i^A and T sentiment multinomial distributions $\{\theta_{i,t}^S\}_{t=1}^T$ will be generated respectively. Each aspect mention $a_k \in \mathbf{a}_i$ will be assigned with a label z_k generated from θ_d^A, while each associated sentiment mention $o_j \in \mathbf{o}_{ik}$ will be generated from θ_{i,z_k}^S. Then, the unit vector of each word will be generated from corresponding vMF distributions as indicated by their labels.

To summarize:

$$
\begin{aligned}
\theta_i^A &\sim \text{Dirichlet}(\cdot|\boldsymbol{\alpha}_A), && d_i \in \mathcal{D} \\
\theta_{i,t}^S &\sim \text{Dirichlet}(\cdot|\boldsymbol{\alpha}_S), && d_i \in \mathcal{D}, t \in [T] \\
z_k &\sim \text{Categorical}(\cdot|\theta_i^A), && a_k \in \mathbf{a}_i \\
y_j &\sim \text{Categorical}(\cdot|\theta_{i,z_k}^S), && o_j \in \mathbf{o}_{ik} \\
\mathbf{x}_{a_k} &\sim \text{vMF}(\cdot|\boldsymbol{\mu}_{z_k}^A, \kappa_{z_k}^A), && a_k \in \mathbf{a}_i \\
\mathbf{x}_{o_j} &\sim \text{vMF}(\cdot|\boldsymbol{\mu}_{y_j}^S, \kappa_{y_j}^S), && o_j \in \mathbf{o}_{ik}
\end{aligned}
$$

We infer the model by Gibbs sampling. Since both the vMF distribution and the Dirichlet distribution have conjugate priors, we can accordingly develop a collapsed Gibbs sampler for labels z_k and y_j by integrating out parameters $\boldsymbol{\mu}_{A,t}$, $\boldsymbol{\mu}_{S,y}$, θ_i^A and $\theta_{i,t}^S$.

We can sample z_k according to:

$$
\begin{aligned}
&\mathbb{P}(z_k = t|\mathbf{Z}^{-ik}, \mathbf{Y}^{-ik\bullet}, \mathbf{X}, \boldsymbol{\kappa}; \boldsymbol{\Phi}) \\
\propto &\big(n_{i,-k}^{(t)} + \boldsymbol{\alpha}_A^{(t)}\big) \\
\times &\frac{\Gamma\big(\sum_y (m_{i,-k}^{(t,y)} + \boldsymbol{\alpha}_S^{(y)})\big)}{\Gamma\big(\sum_y (m_{i,-k}^{(t,y)} + m_{i,k}^{(\bullet,y)} + \boldsymbol{\alpha}_S^{(y)})\big)} \\
\times &\prod_{y \in \mathcal{Y}} \frac{\Gamma(m_{i,-k}^{(t,y)} + m_{i,k}^{(\bullet,y)} + \boldsymbol{\alpha}_S^{(y)})}{\Gamma(m_{i,-k}^{(t,y)} + \boldsymbol{\alpha}_S^{(y)})} \\
\times &\frac{C_\nu(\kappa_t^A) \cdot C_\nu(\|C_A \boldsymbol{\mu}_A + \kappa_t^A \mathbf{x}_{A,t}^{-ik}\|)}{C_\nu\big(\big\|C_A \boldsymbol{\mu}_A + \kappa_t^A (\mathbf{x}_{A,t}^{-ik} + \mathbf{x}_{a_k})\big\|\big)}
\end{aligned}
$$

where $\boldsymbol{\Phi} = [\boldsymbol{\mu}_A\ C_A\ \boldsymbol{\mu}_S\ C_S\ m_A\ \sigma_A^2\ m_S\ \sigma_S^2\ \boldsymbol{\alpha}_A\ \boldsymbol{\alpha}_S]$ is the collection of all prior parameters; $n_{i,-k}^{(t)}$ is the count of aspect mentions in document d_i with hidden variable $z_k = t$, without considering the k-th aspect mention in d_i; $m_{i,k}^{(\bullet,y)}$ is the count of sentiment mentions associated

to the k-th aspect mention with hidden variable $y_j = y$; $m_{i,-k}^{(t,y)}$ is the count of sentiment mentions associated with any aspect mentions with $z'_k = t$ but the k-th aspect mention, and with their own hidden sentiment variable as y; $\mathbf{x}_{A,t}^{-ik}$ is the sum of unit embedding vectors of aspect mentions with hidden variable $z_k = t$.

We can sample y_j according to:

$$\begin{aligned}&\mathbb{P}(y_j = y|\mathbf{Z}, \mathbf{Y}^{-ikj}, \mathbf{X}, \boldsymbol{\kappa}; \boldsymbol{\Phi})\\ \propto&\big(m_{ik,-j}^{(t,y)} + \boldsymbol{\alpha}_S^{(y)}\big)\frac{C_\nu(\kappa_y^S)\cdot C_\nu(\|C_S\boldsymbol{\mu}_S + \kappa_y^S\mathbf{x}_{S,y}^{-ikj}\|)}{C_\nu\big(\big\|C_S\boldsymbol{\mu}_S + \kappa_y^S(\mathbf{x}_{S,y}^{-ikj} + \mathbf{x}_{o_j})\big\|\big)}\end{aligned}$$

where $m_{ik,-j}^{(t,y)}$ is the count of sentiment mentions with their hidden variable as y and their associated aspect mentions' hidden variable as t, but without considering the j-th sentiment mentions of the k-th aspect in document i; and $\mathbf{x}_{S,y}^{-ikj}$ is the sum of unit embedding vectors of sentiment mentions with hidden variable as y but without considering the current sentiment mention.

We specify the prior for sentiment vMF distribution's mean vector $\boldsymbol{\mu}_y^S$ as vMF centered as the mean direction of seed sentiment words with label y as a guidance.

By estimating the mean direction $\hat{\boldsymbol{\mu}}_y^S$ for the sentiment vMF distributions, we can derive the rectified positive and negative vectors $\mathbf{v}_+ = \hat{\boldsymbol{\mu}}_1^S$ and $\mathbf{v}_- = \hat{\boldsymbol{\mu}}_0^S$:

$$\mathbf{v}_+ = \frac{C_S\boldsymbol{\mu}_S + \kappa_y^S\mathbf{x}_{S,1}}{\big\|C_S\boldsymbol{\mu}_S + \kappa_1^S\mathbf{x}_{S,1}\big\|},\ \mathbf{v}_- = \frac{C_S\boldsymbol{\mu}_S + \kappa_0^S\mathbf{x}_{S,1}}{\big\|C_S\boldsymbol{\mu}_S + \kappa_0^S\mathbf{x}_{S,0}\big\|}$$

where $\mathbf{x}_{S,1}$ and $\mathbf{x}_{S,0}$ are the sum of unit embedding vector of sentiment mentions with inferred hidden variable as 1 and 0 respectively.

We can use these rectified vectors to perform polarity assignment for sentiment lexicon and aspect-based sentiment classification as the baseline.

5.5 EXPERIMENTS

In this section, we verify the effectiveness of our proposed methods on real world review data sets.

5.5.1 Data set.

We introduce the data sets used in our experiments.

Table 5.1: Performance comparison of aspect lexicon expansion (%).

Data set	Method	P	R	F_1
Hotel	HU	60.95	43.90	51.04
	DP	51.56	99.89	68.01
	PF	80.27	72.99	76.46
Restaurant	HU	85.80	35.39	50.11
	DP	57.54	99.37	72.88
	PF	87.88	68.25	76.83

Table 5.2: Performance comparison of sentiment lexicon expansion (%).

Data set	Method	P	R	F_1
Hotel	HU	48.43	95.37	64.24
	DP	70.75	84.30	76.93
	PF	84.83	75.94	80.14
Restaurant	HU	38.37	97.33	55.04
	DP	59.36	93.27	72.55
	PF	84.71	70.58	77.00

Hotel. We utilize a hotel review data set from [84, 85]. In the hotel data, reviewers can provide 1 to 5 star ratings on several aspects such as room, service *etc.* We utilize reviews from 181 hotels, as hotels with less than 50 reviews are removed, which results in $17,865$ reviews.

Restaurant. We create a randomly sampled subset of $10,000$ public Yelp[1] restaurant reviews. For the purpose of evaluation, we also include 6,060 labeled sentences from restaurant reviews [28], where each sentence is labeled with a set of aspect mentions along with their corresponding sentiment orientation. Sentences without aspect words or with neutral sentiment are removed in our evaluation.

Similar to previous chapters, for both data sets, we preprocess with phrase mining [50] and then train a 200 dimension word embedding by word2vec [61]. Details can be found in Section 3.2.

5.5.2 Lexicon expansion.

We first evaluate the task of lexicon expansion.

Methods evaluated. We compare the following lexicon expansion methods.

- *Frequency-based method (HU).* A lexicon expansion method based on word frequencies

[1] `https://www.yelp.com/dataset/challenge`

and their PoS tags, proposed by Hu *et al.* [32]. We use an NLP pipeline[2] to parse sentences.

- *Double propagation (DP).* A double propagation method proposed by Qiu *et al.* [71] that relies on hand-crafted rules based on dependency information. We use the same NLP pipeline as above to obtain the syntactic information.

* *Pattern features (PF).* Our proposed method based on pattern feature mining.

Evaluation metrics. We use a pooling strategy to generate the ground-truth. We obtain the aspect and sentiment lexicons generated by all the evaluated methods, and label the union of all the lexicons. We only label words with frequency no less than 50 due to the large lexicon size. We evaluate the performance by weighted versions of precision (P), recall (R) and F_1-score (F_1), where each word is weighted by its frequency in the corpus. Similar measures are adopted in [53].

Experiment setup. We use 10 seed aspect words and 10 seed sentiment words for each data set. Notice that the size of seed in our experiments is substantially smaller than the seed sets in previous studies such as [71], where more than 1,000 seed sentiment words are used. We set β to 0.7 for both data sets.

Results. We present the results in Table 5.1 and Table 5.2. It can be observed that our method outperforms baselines in terms of F_1 score on both tasks and both data sets. While DP generally has higher recall, its rules are developed for product reviews and are likely to generate a lot of false positives. In comparison, our method does not rely on domain specific rules. We achieve the highest precision in both data sets and both tasks by our relative prudent expansion method, while keeping a decent recall.

5.5.3 Sentiment classification.

We also evaluate the performance of aspect-based sentiment classification.

Methods evaluated. The following methods are compared in our experiments.

- *Double propagation (DP).* The method proposed in [71], which includes both lexicon expansion and polarity assignment.
- *Aspect and sentiment unification model (ASUM).* The method proposed by [36].

[2] `https://spacy.io/`

Table 5.3: Performance comparison of aspect-based sentiment classification (%)

Data set	Method	P	R	F_1
Hotel	DP	37.17	8.45	13.77
	ASUM	24.82	74.41	37.23
	HU+EMB+R	16.05	57.43	25.08
	DP+EMB+R	45.58	33.72	38.76
	PF+EMB	37.44	50.02	42.83
	PF+EMB+R	44.60	52.04	48.03
Restaurant	DP	57.45	5.31	9.72
	HU+EMB+R	58.33	22.03	31.98
	DP+EMB+R	74.47	22.95	35.09
	PF+EMB	78.89	22.30	34.76
	PF+EMB+R	74.41	26.69	39.29

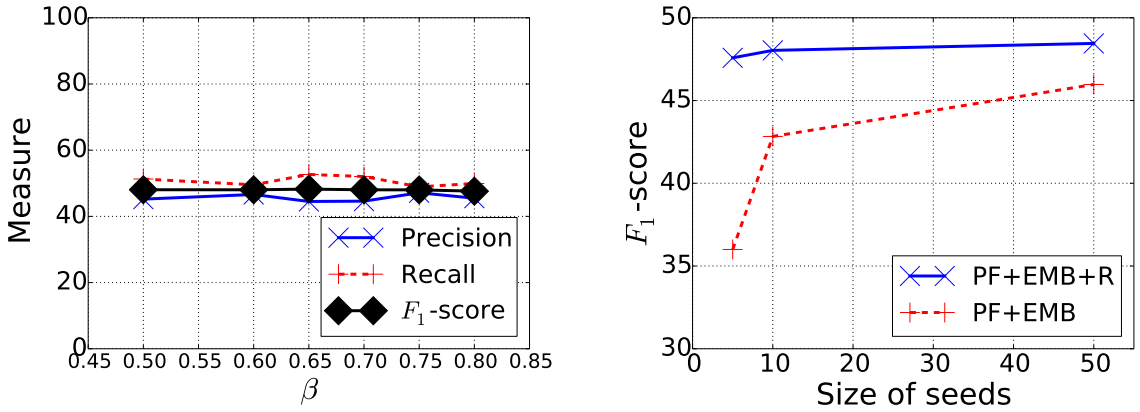
- *HU/DP+EMB+R*. Feeding the lexicon constructed by a previously mentioned baseline into our sentiment classification method.
- *PF+EMB*. Our proposed baseline method merely using embedding vector and seed sentiment words.
- * *PF+EMB+R*. Our proposed method with rectified polarity assignment of sentiment lexicon.

Evaluation metrics. For Hotel data set, we label documents with 1 or 2 star rating as negative sentiment, while 4 or 5 star rating as positive sentiment for each ground-truth aspect. Since user rating ground-truth is only available for more "general" aspects, we carefully pick a set of frequent aspect words A_k for each ground-truth aspect k. In evaluation, we take the average of the output sentiment polarity labels of aspect mentions corresponding to the same ground-truth aspect as the aggregated output.

If a document does not have user rating on ground-truth aspect k or does not contain words from A_k, then it is not evaluated on ground-truth aspect k. Moreover, documents with 3-star rating on k are also not evaluated on ground-truth aspect k.

We evaluate the performance by precision (P), recall (R) and F_1-score (F_1). Notice that reviews with positive sentiment are overwhelmingly more than reviews with negative sentiment in our data set, which makes the prediction a relatively trivial task. Hence we treat negative sentiment as "positive" label while calculating the evaluation measures.

Results. The results are shown in Table 5.3. It can be observed that our method performs the best in terms of F_1-score. On both data sets, our method constantly achieves around +5% improvement over the best performed baselines.



(a) β (b) Number of seeds

Figure 5.4: Performance w.r.t several parameters.

The results confirm that our lexicons are better than lexicons constructed by other baselines. Our method achieves +5-10% improvement in terms of F_1 on both data sets comparing to the same method with other baseline lexicons.

Another observation is that our rectification step substantially improves the performance. On both data sets, it achieves around +4% of improvement. This is because it combines the embedding signals with the information from the corpus.

Notice that our evaluation setting is much more challenging due to the minimal supervision and imbalance distribution of sentiment labels. Thus the performance is generally lower than typical sentiment analysis.

Parameter analysis. We first study the sensitivity of threshold parameter β in lexicon expansion. We measure the performance of aspect-based sentiment classification on Hotel data set based on lexicon constructed with β set to different values between 0.5 to 0.8. As Figure 5.4(a) suggests, the performance remains stable. The difference of F_1-score is within 1%.

We also study how the performance change w.r.t. the size of seed set. As Figure 5.4(b) shows, with only 5 seeds for each lexicon, the performance of our method is higher than our baseline with 50 seeds. This shows the power of our rectification method in the sentiment classification stage.

Case study. Table 5.4 shows a case study to provide an in-depth analysis of how our method outperforms other method. We majorly compare our PF+EMB+R method with our proposed baseline PF+EMB, as well as a variation with lexicon built by DP.

The first two sentences in Table 5.4 show how our rectification method improves the performance. In the first sentence, our method with sentiment rectification can correctly assign a positive polarity score to "large", while the baseline without rectification mistakenly

Table 5.4: Case study. For each sentence, we show the identified aspect mentions, sentiment mentions and their sentiment polarity assignment for each method.

Method	Aspect	(Sentiment, Polarity)
Sentence	*The bathroom is large.*	
PF+EMB+R	bathroom	(large, +)
PF+EMB	bathroom	(large, −)
Sentence	*Quiet room.*	
PF+EMB+R	room	(quiet, +)
PF+EMB	room	(quiet, −)
Sentence	*We found our bedding sooooo awful!*	
PF+EMB+R	bedding	(awful, −)
DP+EMB+R	bedding	(sooooo, +), (awful,−)
Sentence	*Reception staff were not friendly and occasion quite rude.*	
PF+EMB+R	staff	(not friendly, −), (rude, −)
DP+EMB+R	occasion	(not friendly, −), (rude, −)

mark it as negative. Similarly, the baseline recognizes "quiet" as a negative sentiment word, while the rectified version correctly identifies it as positive.

The other two sentences shows how the lower precision of lexicon affects the overall sentiment classification performance. In the sentence "*We found our bedding sooooo awful!*", the lexicon constructed by DP mistakenly take "sooooo" as a sentiment word, while it actually should be an intensifier with informal spelling. In the last sentence, the misspelled "occasion" is identified as an aspect mention by DP lexicon. It steals the sentiment mentions "friendly" and "rude" from the actual aspect mention "staff" in this sentence as we assign sentiment mentions to the closest aspect mention. However, our lexicon correctly outputs the only aspect mention "staff" in this sentence.

5.6 SUMMARY

We study to perform aspect-based sentiment analysis with minimal user guidance. We start with a lexicon expansion step and then develop a generative model to improve sentiment classification based on word embedding. Our experiments show the effectiveness of our method. This facilitates building a sentiment analysis tool for domains with limited resource. In principle, this work is language agnostic and can be seamlessly extended to other language, which would be an interesting direction for future work.

CHAPTER 6: JOINT WEAKLY-SUPERVISED TEXT MINING FOR ASPECT AND SENTIMENT ANALYSIS

6.1 OVERVIEW

In this chapter, we further study the aspect-based sentiment analysis, but further considering the joint classification of both the aspect and the sentiment. Different from the previous chapter which only outputs the aspect mentions with sentiment classes, we allow users to also provide seed words for each aspect, and output the aspect classes along with the corresponding sentiment classes. We focus on sentence-level analysis since each sentence usually belongs to an aspect.

The basic idea to attack the problem is to utilize an autoencoder to discover the aspect and sentiment structure hidden in sentences. By training the autoencoder to reconstruct sentences in the unlabeled corpus, we can learn a "dictionary" where each aspect and each sentiment can be characterized by a vector in the embedding space, reflecting the frequent words of the corresponding aspect and sentiment. We also design a regularization on the dictionary to instill the user guidance, such that the learned vectors in the dictionary remain close to the seed words in the embedding space. Moreover, we adapt the autoencoder structure so that the model is capable of learning a sentiment dictionary for each aspect, characterizing the aspect-specific sentiment in the embedding space.

More concretely, we make the following contributions:

- *Modeling the aspect and sentiment of sentences with user guidance.* We employ a structure with two parallel autoencoder to learn the aspect dictionary and sentiment dictionary of the corpus by reconstructing unlabeled sentences. We propose a regularization method to integrate user guidance into the modeling process. We also attach an attention layer to identify aspect and sentiment words in sentences.

- *Characterizing aspect-specific sentiment words.* We adapt the model by joining the aspect and sentiment encoder to reconstruct the sentiment of sentences. Thereby we can learn a sentiment dictionary for each aspect which captures the signals from aspect-specific sentiment words.

- *Conducting experiments on real data sets.* We verify the effectiveness of our proposed methods on two real world data sets from different domains.

We introduce the details of our studies below.

6.2 PRELIMINARIES

In this section, we start by introducing basic notations of our data set. Then we move on to formalize the research problem.

6.2.1 Notations

We represent a given corpus as a set of documents $\mathcal{D} = \{d_i\}_{i=1}^{n}$, where each document d_i can be represented as a sequence of sentences $d_i = (s_1, \ldots, s_{|d_i|})$. A sentence s_j consists of a sequence of words $s_j = (w_1, \ldots, w_{|s_j|})$, where each word w_k takes a value from a vocabulary $\mathcal{V}$.

Again, each "word" $w_k \in \mathcal{V}$ can be a unigram word, a multigram phrase (*e.g.*, "battery life", "chocolate cake") or a subword like "n't" in "don't". We simply use the term "word" for simplicity.

In this chapter, we denote the normalized word embedding of word w as $\mathbf{e}_w$, while the cosine similarity between word embedding is still denoted as $\text{sim}(\mathbf{e}_w, \mathbf{e}_{w'})$, as defined in Eq. 3.1.

In addition, there are K aspects in the given domain. Each sentence s_j in the corpus corresponds to a set of aspects $\mathbf{a}_j \subset \{1, \ldots, K\}$. In principle, $\mathbf{a}_j$ can be an empty set or a set with size larger than one, corresponding to no aspect mentioned or multiple aspects mentioned respectively. However, sentences without aspect mentioned are not targeted in this task, and sentences with multiple aspects are rare in the real world data. Hence, we assume that each sentence s_j can be classified into one and only one aspect a_j.

Moreover, each sentence s_j is also associated with a sentiment label y_j, where $y_j \in \{0, 1\}$. 0 stands for *Negative* sentiment, and 1 stands for *Positive* sentiment.

6.2.2 Problem Formalization

Based on the above notations, we describe our problem definition in a formalized way.

First, an unlabeled corpus of reviews $\mathcal{D}$ from a specific domain should be given. A domain refers to a relatively consistent category of products or services, such as the hotel domain, the restaurant domain, and the laptop domain. We assume users have a relatively complete set of aspects of interest in the given domain. For example, in the restaurant domain, the set of aspects would be {*Food*, *Service*, *Ambience*, *Location*, *Drinks*}, corresponding to aspects 1 to K respectively.

Users can provide some seed aspect words and seed sentiment words as guidance. Seed

aspect words are a group of word sets $\mathcal{V}_{A_1}, \ldots, \mathcal{V}_{A_K}$, where each word set $\mathcal{V}_{A_t} \subset \mathcal{V}$ corresponds to an aspect. Similarly, seed sentiment words can be denoted as $\mathcal{V}_{S_0}$ and $\mathcal{V}_{S_1}$, corresponding to *Negative* and *Positive* sentiments.

Notice that we do not require a ridiculously large set of seed words. For example, only 5 words for each aspect or each sentiment class would be sufficient. This setting can be easily fulfilled within minutes by any regular user with reasonable knowledge about the data without requiring any additional linguistic knowledge, language resources or exhaustive labor.

The problem can be formalized as:

Problem 6.1. *Given a corpus of review documents $\mathcal{D}$, some seed aspect words $\mathcal{V}_{A_1}, \ldots, \mathcal{V}_{A_K}$ for each of the K aspects, and seed sentiment words $\mathcal{V}_{S_0}, \mathcal{V}_{S_1}$ for* Negative *and* Positive *sentiments, we aim to develop a classifier such that for any sentence s_j, we can output its aspect label and corresponding sentiment labels (a_j, y_j).*

6.3 ASPECT SENTIMENT AUTOENCODER

In this section, we describe a neural model to characterize the aspect and sentiment for each sentence in a given corpus.

6.3.1 Overview

The model contains two major components: an attention module and an autoencoder module. Aspect and sentiment are modeled in two parallel autoencoders with attention.

The first part is the attention module, which aims to emphasize specific parts of the given sentence for aspect or sentiment classification. For example, in a simple sentence "The food is good", the word "food" should be emphasized more in terms of aspect classification, while the word "good" should be given more attention in sentiment classification.

The second part is the autoencoder structure to reconstruct the attention-weighted sentence representations of aspect and sentiment from two low dimension "dictionaries" initiated and regularized by user-provided seed words. Each row of the aspect or sentiment dictionary is a vector in the embedding space representing an aspect class or a sentiment class. By reduction and reconstruction of sentences, we can learn the dictionary which reflects the frequent words in each aspect and each sentiment. Meanwhile, the regularization from seed words maintains the learned dictionary to be well aligned with aspect classes and sentiment classes of interest. After the model is trained, we can obtain the classification results for any sentence from this module.

We describe the details of each module below.

6.3.2 Sentence Representation with Attention

For each sentence s_j, we first construct its vector representations for aspect and sentiment respectively, denoted as $\mathbf{z}_j^A$ and $\mathbf{z}_j^S$. The aspect vector representation $\mathbf{z}_j^A$ aims to summarize the aspect relevant information from the sentence in the embedding space, while the sentiment vector representation $\mathbf{z}_j^S$ should capture the sentiment related information.

Both vectors $\mathbf{z}_j^A$ and $\mathbf{z}_j^S$ are defined as a weighted summation of word embedding vectors $\mathbf{e}_{w_k}$ for $w_k \in s_j$ where w_k is valid, *i.e.* not a stop word, a number or a punctuation. More precisely,

$$\mathbf{z}_j^A = \sum_k \alpha_k^A \mathbf{e}_{w_k}, \qquad \mathbf{z}_j^S = \sum_k \alpha_k^S \mathbf{e}_{w_k}$$

where the weights α_k^A and α_k^S are non-negative attention scores derived from attention models. Generally, the weights α_k^A and α_k^S can be regarded as the probabilities of whether w_k should be focused on to determine the aspect and sentiment class respectively. Take aspect attention as example, the aspect attention weight α_k^A can be calculated based on the embedding of word w_k as well as the global context of the entire sentence. More concretely:

$$\begin{aligned}
\alpha_k^A &= \frac{\exp(u_k)}{\sum_k \exp(u_k)} \\
u_k &= \mathbf{e}_{w_k}^\top \mathbf{M}_A \mathbf{x}_j \\
\mathbf{x}_j &= \frac{1}{|m_j|} \sum_k \mathbf{e}_{w_k}
\end{aligned}$$

where $\mathbf{x}_j$ is the unweighted average word embedding, and m_j is the number of valid words in s_j. The matrix $\mathbf{M}_A \in \mathbb{R}^{\nu \times \nu}$ can be learned during the training process.

The attention for sentiment is similar, while the transformation matrix is replaced by another matrix $\mathbf{M}_S$.

6.3.3 Sentence Reconstruction

We have two vector representations $\mathbf{z}_j^A$ and $\mathbf{z}_j^S$ of the sentence s_j, which emphasize on the aspect and the sentiment of the sentence respectively. Now we learn to reconstruct these two representations $\mathbf{z}_j^A$ and $\mathbf{z}_j^S$ from the aspect dictionary and the sentiment dictionary respectively. A dictionary is a matrix where each row is a vector in the embedding space,

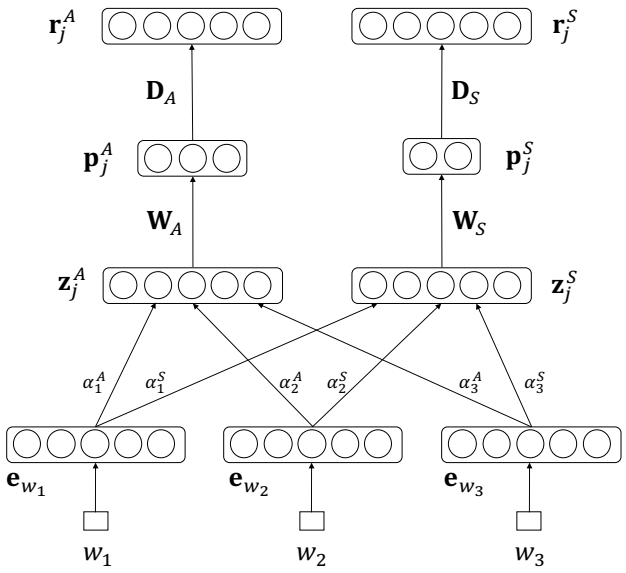


Figure 6.1: A graphical illustration of the aspect sentiment autoencoder model. Each word w_k is represented by its embedding $\mathbf{e}_{w_k}$. The aspect and sentiment representation of sentence $\mathbf{z}_j^A$ and $\mathbf{z}_j^S$ are weighted average of $\mathbf{e}_{w_k}$'s from two attention mechanisms. $\mathbf{p}_j^A$ and $\mathbf{p}_j^S$ are probability distributions over aspect and sentiment classes respectively. $\mathbf{r}_j^A$ and $\mathbf{r}_j^S$ are the reconstructed representations from aspect dictionary $\mathbf{D}_A$ and sentiment dictionary $\mathbf{D}_S$.

representing an aspect or a sentiment class. Learning the dictionary that best recovers sentences in the corpus is essentially capturing the most frequent semantic regions in the embedding space, which serves as a good summary of the corpus. Moreover, since the dictionary shares the same embedding space with words, it makes it straightforward to introduce regularization on the dictionary from user-provided seed words. The regularization will be introduced in Section 6.3.4.

Reconstructing the aspect vector. We first reduce the aspect vector representation $\mathbf{z}_j^A$ to a K-dimension vector by:

$$\mathbf{p}_j^A = \text{softmax}(\mathbf{W}_A \cdot \mathbf{z}_j^A) \tag{6.1}$$

where $\mathbf{W}_A \in \mathbb{R}^{K \times \nu}$ are model parameters to be learned. The softmax activation introduces non-linearity and ensures the resulting compressed vector $\mathbf{p}_j^A$ is non-negative with the ℓ_1-norm of 1.

The vector $\mathbf{p}_j^A$ can be viewed as a distribution over different aspects where each element represents the probability of whether the sentence s_j should belong to the corresponding aspect. In order to enforce this property, we try to reconstruct the original aspect vector

representation of the sentence $\mathbf{z}_j^A$ from $\mathbf{p}_j^A$, with the help of an aspect dictionary $\mathbf{D}_A$:

$$\mathbf{r}_j^A = \mathbf{D}_A^\top \cdot \mathbf{p}_j^A \tag{6.2}$$

where $\mathbf{D}_A \in \mathbb{R}^{K \times \nu}$ is the aspect dictionary to be learned. Each row of $\mathbf{D}_A$ can be regarded as an embedding vector of an aspect in the embedding space. Ideally, the vector of an aspect should be close to representative words frequently mentioned in this aspect.

Reconstructing the sentiment vector. Similarly, the sentiment vector reconstruction starts by reducing the sentiment vector representation $\mathbf{z}_j^S$ to a 2-dimension vector:

$$\mathbf{p}_j^S = \text{softmax}(\mathbf{W}_S \cdot \mathbf{z}_j^S) \tag{6.3}$$

where the model parameters $\mathbf{W}_S \in \mathbb{R}^{2 \times \nu}$ can be trained.

We then reconstruct the sentiment vector $\mathbf{z}_j^S$ from a sentiment dictionary $\mathbf{D}_S$:

$$\mathbf{r}_j^S = \mathbf{D}_S^\top \cdot \mathbf{p}_j^S \tag{6.4}$$

Some studies [27, 7] also adopt a similar autoencoder structure to learn the aspect dictionary. However, they do not separate the reconstruction of aspect and sentiment in sentences. They also do not utilize regularization to confine the learned dictionary to be aligned with user guidance, which will be described below.

6.3.4 Regularization

We place several regularization terms on the decoder parameters $\mathbf{D}_A$ and $\mathbf{D}_S$ to leverage the user-provided seed words.

Seed regularization. We leverage the information from seed words by applying a regularization on the dictionary parameters.

First, we describe the regularization on the aspect dictionary matrix $\mathbf{D}_A$. We create a "prior" matrix $\mathbf{R}_A \in \mathbb{R}^{K \times \nu}$ with the same size as parameter $\mathbf{D}_A$. The t-th row of $\mathbf{R}_A$ is assigned with the average embedding of seed words in the corresponding aspect, namely

$$\mathbf{R}_A^{(t)} = \frac{\sum_{w \in \mathcal{V}_{A_t}} \mathbf{e}_w^\top}{\| \sum_{w \in \mathcal{V}_{A_t}} \mathbf{e}_w^\top \|} \tag{6.5}$$

The objective is to penalize when the learned embedding of the t-th aspect (represented by the t-th row of $\mathbf{D}_A$) deviates too far away from the average embedding of seed words.

Accordingly, the regularization term can be written as:

$$C_A(\theta) = \sum_{t=1}^{K} \Big[1 - \text{sim}\Big(\mathbf{R}_A^{(t)}, \mathbf{D}_A^{(t)}\Big)\Big] \tag{6.6}$$

where $\mathbf{R}_A^{(t)}$ and $\mathbf{D}_A^{(t)}$ represents the t-th row of $\mathbf{R}_A$ and $\mathbf{D}_A$ respectively.

Similarly, the prior matrix for sentiment dictionary is a $2 \times \nu$ matrix, where

$$\mathbf{R}_S^{(y)} = \frac{\sum_{w \in \mathcal{V}_{S_y}} \mathbf{e}_w^\top}{\| \sum_{w \in \mathcal{V}_{S_y}} \mathbf{e}_w^\top \|}, \ \forall y \in \{0, 1\} \tag{6.7}$$

and the regularization term on the sentiment dictionary matrix is

$$C_S(\theta) = \sum_{y \in \{0,1\}} \Big[1 - \text{sim}\Big(\mathbf{R}_S^{(y)}, \mathbf{D}_S^{(y)}\Big)\Big] \tag{6.8}$$

Redundancy regularization. Another regularization typically adopted is the penalty on redundancy of learned aspect or sentiment dictionary. The intuition is to prevent the learned dictionary matrix from having almost identical rows. As an example, for a learned dictionary matrix $\mathbf{D}$, the redundancy regularization term is:

$$X(\mathbf{D}) = \|\mathbf{D}_n \mathbf{D}_n^\top - \mathbf{I}\| \tag{6.9}$$

where $\mathbf{D}_n$ is $\mathbf{D}$ with each row normalized to a unit length vector, and $\mathbf{I}$ is the identity matrix. This term is also utilized in [27].

In our work, we can apply the redundancy regularization term on all the dictionary matrices. We simply define $X_A(\theta) = X(\mathbf{D}_A)$ and $X_S(\theta) = X(\mathbf{D}_S)$.

6.3.5 Training Objective

The overall objective is to minimize the loss between the reconstructed vectors and the sentence representation vectors for both aspect and sentiment. The similarity between two vectors are measured by cosine similarity. For each sentence, we aim to maximize the similarity between the derived sentence representation vectors and the reconstructed vectors. In addition, we randomly sample m sentences as negative samples and minimize the similarity between the vector representations of negative samples and the reconstructed vectors. The

loss is formalized in a similar way to the objective functions proposed in [27]:

$$L_A(\theta) = \sum_{s_j \in \mathcal{D}} \sum_{i=1}^{m} \max\big(0, 1 - \text{sim}(\mathbf{r}_j^A, \mathbf{z}_j^A) + \text{sim}(\mathbf{r}_j^A, \mathbf{x}_{n_i})\big) \tag{6.10}$$

$$L_S(\theta) = \sum_{s_j \in \mathcal{D}} \sum_{i=1}^{m} \max\big(0, 1 - \text{sim}(\mathbf{r}_j^S, \mathbf{z}_j^S) + \text{sim}(\mathbf{r}_j^S, \mathbf{x}_{n_i})\big) \tag{6.11}$$

where $\mathbf{x}_{n_i}$ are the average embedding of words in the i-th randomly sampled sentence.

The final training objective is to minimize the overall loss along with the regularization term:

$$\begin{aligned} L(\theta) = L_A(\theta) + L_S(\theta) + \lambda_1\big(C_A(\theta) + C_S(\theta)\big) \\ + \lambda_2\big(X_A(\theta) + X_S(\theta)\big) \end{aligned} \tag{6.12}$$

where λ_1 and λ_2 are two hyperparameters given by users to weigh the effect of different regularization terms.

6.4 JOINT ASPECT SENTIMENT AUTOENCODER

The model described above consists of two parallel autoencoder structures for aspect and sentiment respectively. However, the occurrences of aspect and sentiment words in sentences are correlated.

In this section, we describe a joint autoencoder for aspect and sentiment in sentences. The model is designed based on the observation that some sentiment words specifically used for a certain aspect. For example, "delicious" is specifically used to express positive sentiment on *Food* aspect, while "rude" is often used to express negative sentiment on *Service* aspect.

We capture the aspect-specific sentiment words by expanding the universal sentiment dictionary into several aspect-based sentiment dictionaries. To learn the dictionaries, we join the aspect and sentiment encoder to generate the distributions over the space of aspect-sentiment pairs. Then we accordingly reconstruct the sentiment representation of sentences.

Since the attention and aspect reconstruction part is identical to the model described in Section 6.3, we only present the different parts of the model.

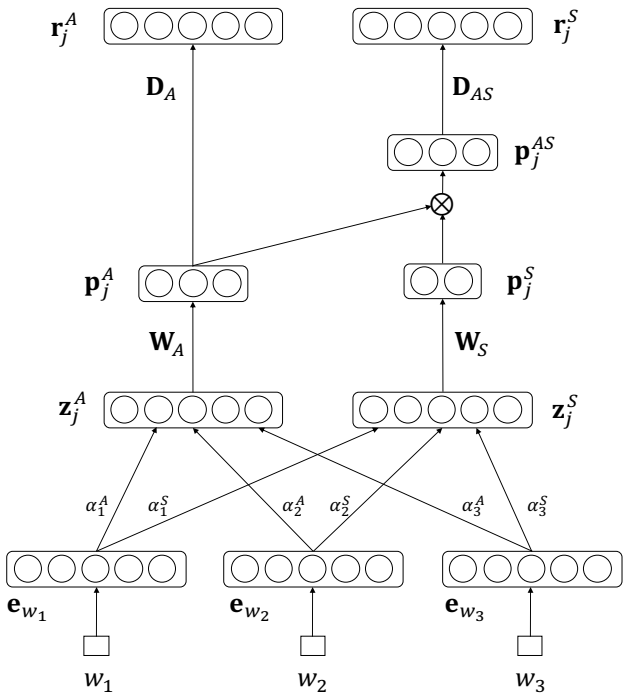


Figure 6.2: A graphical illustration of the joint aspect sentiment autoencoder model. The difference from Figure 6.1 is the joining operation of $\mathbf{p}_j^A$ and $\mathbf{p}_j^S$ which outputs a distribution $\mathbf{p}_j^{AS}$ over pairs of aspect and sentiment classes. Also, an aspect-specific sentiment dictionary $\mathbf{D}_{AS}$ is introduced to recover the sentiment representation.

6.4.1 Sentence Reconstruction

In this subsection, we only focus on the joint reconstruction of sentiment representation vector.

The intuition is to capture the aspect-specific sentiment words in sentiment dictionaries of each aspect. The occurring probability of an aspect-specific word in a sentence s_j is not only related to the sentiment class of the current sentence, but also the aspect class of the current sentence. Therefore, We start by joining the aspect distribution $\mathbf{p}_j^A$ and the sentiment distribution $\mathbf{p}_j^S$.

We derive the joint encoded vector $\mathbf{p}_j^{AS}$ by taking the outer product of $\mathbf{p}_j^S$ and $\mathbf{p}_j^A$ and then flattening it into a vector:

$$\mathbf{p}_j^{AS} = \text{vec}\Big(\mathbf{p}_j^S \otimes \mathbf{p}_j^A\Big) \tag{6.13}$$

where $\mathbf{p}_j^{AS}$ will be a $2K$-dimension vector. The $2t$-th element of $\mathbf{p}_j^{AS}$ can be interpreted as the probability that sentence s_j expresses *Negative* sentiment on the t-th aspect, while the $(2t+1)$-th correspond to *Positive* sentiment on the t-th aspect.

In order to reconstruct the sentiment representation from the joint aspect and sentiment distribution $\mathbf{p}_j^{AS}$, we need to learn a larger sentiment dictionary. More specifically, the sentiment dictionary has to be aspect-specific. Each aspect has its own sentiment dictionary. We denote the aspect-specific dictionary as $\mathbf{D}_{AS} \in \mathbb{R}^{2K \times \nu}$, where the $2t$-th and $(2t+1)$-th row form a sentiment dictionary for the t-th aspect.

Now we can reconstruct the sentiment vector $\mathbf{z}_j^S$ from the joint distribution over aspect and sentiment by:

$$\mathbf{r}_j^S = \mathbf{D}_{AS}^\top \cdot \mathbf{p}_j^{AS} \tag{6.14}$$

By minimizing the loss between $\mathbf{r}_j^S$ and $\mathbf{z}_j^S$, the $2t$-th and $(2t+1)$-th row of $\mathbf{D}_{AS}$ should become the vector representation of the *Negative* and *Positive* sentiment for the t-th aspect in the embedding space respectively. Notice that the learned sentiment dictionary for each aspect summarizes *all* possible sentiment words co-occurred with the aspect, which include both aspect-specific sentiment words as well as general sentiment words. For example, the embedding for *Positive* sentiment on *Food* aspect should still be close to general words like "good" and "great", while also relatively close to aspect-specific words like "delicious" and "yummy".

6.4.2 Regularization

Since we are learning the aspect-specific sentiment dictionary $\mathbf{D}_{AS}$, the regularization on the dictionary needs to be adapted accordingly. We construct an expanded prior matrix $\mathbf{R}_{AS} \in \mathbb{R}^{2K \times \nu}$ by repetitively tiling the prior matrix we utilized in Eq. (6.7) into the new prior matrix $\mathbf{R}_{AS}$. More specifically,

$$\mathbf{R}_{AS} = [\mathbf{R}_S^{(0)\top} \quad \mathbf{R}_S^{(1)\top} \quad \mathbf{R}_S^{(0)\top} \quad \mathbf{R}_S^{(1)\top} \quad \cdots \quad \mathbf{R}_S^{(1)\top}]^\top \tag{6.15}$$

Essentially, if we regard every two rows in $\mathbf{D}_{AS}$ as a sentiment dictionary for each aspect, then each of them is regularized in the same way as the general sentiment dictionary $\mathbf{D}_S$. However, if users are willing to give some seed words for each aspect-specific sentiment, one can easily instantiate a more informative prior regularization within this framework. We leave this idea for future extensions.

The regularization term on $\mathbf{D}_{AS}$ is similar:

$$C_{AS}(\theta) = \sum_{r=1}^{2K} \Big[1 - \text{sim}\Big(\mathbf{R}_{AS}^{(r)}, \mathbf{D}_{AS}^{(r)}\Big)\Big] \tag{6.16}$$

6.4.3 Training Objective

The overall loss function of this model would be:

$$\begin{aligned} L(\theta) = L_A(\theta) + L_S(\theta) + \lambda_1\big(C_A(\theta) + C_{AS}(\theta)\big) \\ + \lambda_2\big(X_A(\theta) + X_{AS}(\theta)\big) \end{aligned} \tag{6.17}$$

where $X_{AS}(\theta) = X(\mathbf{D}_{AS})$, which is defined in Eq. (6.9).

Although the major difference of the model structure is only reflected in the sentiment reconstruction part, the training results show substantial differences in the aspect reconstruction. The current model structure allows the aspect encoder to contribute substantially to the reconstruction of sentiment representations, especially those with aspect-specific sentiment words. As the model minimizes the reconstruction loss, the aspect autoencoder will also shift some attention to such words and utilize them in the aspect classification. We present some concrete examples in Section 6.5.

6.5 EXPERIMENTAL RESULTS

In this section, we verify the effectiveness of our proposed methods on real world review data sets.

6.5.1 Data Sets

We introduce the data sets used in our experiments.

Restaurant. We collect $47,239$ unlabeled reviews on restaurants from a public Yelp data set[1]. For the purpose of evaluation, we utilize sentences from SemEval-2016 [69] in the restaurant domain as ground-truth, where each sentence is labeled with target entities, entity types and attributes, as well as corresponding sentiment polarity (*Positive*, *Negative* and *Neutral*). We regard the entity types as aspect classes, while neglecting the entity type "Restaurant" since such sentences do not express aspect-specific opinions and are not our

[1] `https://www.yelp.com/dataset/challenge`

targets. We ignore the attributes of entities since they provide more fine-grained information. We also remove sentences with *Neutral* sentiment class to simplify the problem, but it can be seamlessly added with an extra set of seed words.

Laptop. We utilize 14,683 unlabeled Amazon reviews on merchandises in the laptop category, collected by [58, 29]. We also use labeled sentences on the laptop domain from SemEval-2016 [69] for evaluation. Similar to the Restaurant data set, each sentence is labeled with target entities, entity types, attributes and sentiment polarity. There are originally 21 different entity types. We remove some rare entity types and only keep the following 8 entity types as aspect classes: *Support*, *OS*, *Display*, *Battery*, *Company*, *Mouse*, *Software* and *Keyboard*. Again, we ignore the attributes and remove sentences with neutral sentiment.

6.5.2 Experiment Setup

Preprocessing. The unlabeled review documents serve as the training corpus $\mathcal{D}$. We use the sentences tokenizer and word tokenizer provided by *NLTK*[2] to tokenize the documents into list of sentences and to further tokenize each sentence into a list of words. We also tokenize the labeled sentences into lists of words with the same tool.

Next, we adopt a phrase mining technique, *SegPhrase* [50], to discover phrases such as "mini bar" and "front desk", such that they can be treated as a single semantic unit instead of several. SegPhrase can automatically segment sentences into chunks of unigram words and multigram phrases.

We then derive the word embedding by training *word2vec* [61] on our unlabeled set of documents. Therefore for each data set we train a different set of word embedding. Notice that the corpus used for word embedding contains multigram phrases from SegPhrase. Thus each multigram phrase has its own embedding. This is coherent with [50]. Notice that our method does not rely on a specific word embedding algorithm so it can be seamlessly replaced by any other embedding methods.

Methods evaluated. We compare the following methods.

- *Cosine similarity (CosSim).* Performing aspect and sentiment classification by simply calculating the cosine similarity between the average embedding of all words in the given sentence and the average embedding of the seed words of each aspect/sentiment class. Classifying the sentence into the aspect/sentiment class with the highest cosine similarity.

[2] `https://www.nltk.org/`

- *Aspect Sentiment Unification Model (ASUM).* A topic model specifically proposed for aspect-based sentiment analysis by Jo *et al.* [36]. The model also takes seed aspect and seed sentiment words as guidance.

- ⋆ *Aspect Sentiment Autoencoder (ASA).* Our model described in Section 6.3 with two parallel regularized autoencoder structures for aspect and sentiment.

- ⋆ *Joint Aspect Sentiment Autoencoder (JASA).* Our proposed model in Section 6.4 with a joint autoencoder for aspect and sentiment.

Evaluation measures. We evaluate the performance of aspect and sentiment classification respectively. For both the aspect and sentiment classification task, we evaluate the performance by accuracy (A), precision (P), recall (R) and F_1-score (F_1). To clarify, for the multi-class aspect classification task, we employ macro-averaged precision, macro-averaged recall and macro-averaged F_1-score as the evaluation measures.

For our neural network model, we run the experiments 10 times for each method on each data set and report the average performance to reduce the effect of randomness.

Configurations. For both data sets, we provide 5 seed words for each aspect and for each sentiment class. We utilize Adam [41] with default parameter setting to optimize the model. For both data sets, we set the weights for prior regularization and redundancy regularization to $\lambda_1 = 10$ and $\lambda_2 = 0.1$ respectively. The model is trained for 15 epochs, where each epoch contains $1,000$ batches. Each batch contains 50 randomly sampled sentences. Each randomly sampled sentence is paired with $m = 20$ negative sentences as negative samples.

6.5.3 Results

Now we present the experiment results.

Performance comparison. We proceed to evaluate the performance of aspect classification and sentiment classification separately on both Restaurant and Laptop data sets. The overall performance of aspect classification results are in Table 6.1 and the sentiment classification results are in Table 6.2.

In the task of aspect classification, both of our proposed methods ASA and JASA are able to achieve the best overall performances in terms of F_1 score, accuracy and recall. ASA and JASA achieve over 80% of accuracy in Restaurant data set and 76-77% of accuracy in Laptop data set. In comparison, the topic model-based baseline ASUM fails to identify most of the aspects and therefore has poor overall performance on both data set. CosSim achieves

Table 6.1: Performance of aspect classification (%).

Data set	Method	A	P	R	F_1
Restaurant	CosSim	78.67	**65.47**	57.39	59.83
	ASUM	30.79	27.01	26.25	24.75
	ASA	80.19	62.95	82.50	66.96
	JASA	**80.83**	63.67	**82.57**	**67.70**
Laptop	CosSim	55.81	67.46	60.35	56.81
	ASUM	34.24	26.81	32.01	28.21
	ASA	76.03	76.70	78.94	76.87
	JASA	**77.16**	**77.76**	**79.81**	**77.97**

78% accuracy in Restaurant data set with fewer aspects, but only reaches 55% of accuracy in Laptop data set which has more aspects.

Notice that JASA with the joint autoencoder structure can further improve the aspect classification performance on both data sets for +0.7% to +1.1% from the ASE model with parallel autoencoder structures. In order to measure the statistical significance of the improvement, we also compare the overall accuracy of both aspect and sentiment classification by JASA and ASA with 5 different seed sets. For each seed set we run the experiments for 5 times. We perform a paired Student's t-test and shows that the improvement from ASA to JASA is statistically significant on Restaurant and Laptop data set with significance level 0.1% and 0.5% respectively. This shows the benefit of exploiting the correlation between aspect and sentiment words in sentences.

For sentiment classification, our methods ASA and JASA still outperform all the other baselines in both data sets, in terms of all the evaluation measures. Both methods achieve more than 82% of accuracy in Restaurant data set and 74% of accuracy in Laptop data set, with +1% to +5% improvement from baseline methods.

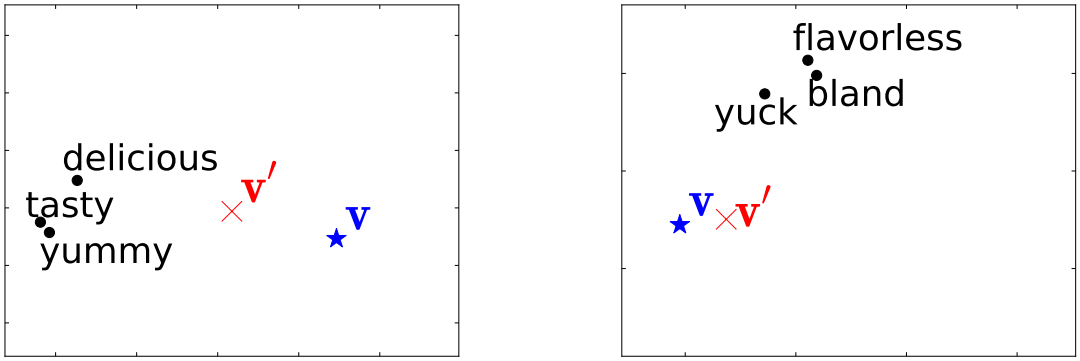
Notice that our reported results are average measures of 10 different results. The reported F_1-score is not directly calculated from the reported precision and recall.

Visualizing aspect-specific sentiment dictionary. We compare learned aspect-specific sentiment dictionary $\mathbf{D}_{AS}$ to the seed words to understand the effect of our dictionary learning module. For the t-th aspect with the sentiment $y \in \{0, 1\}$, we denote the average embedding of seed sentiment word $\mathbf{R}_S^{(y)}$ as $\mathbf{v}$, and the learned embedding in the aspect-specific dictionary $\mathbf{D}_{AS}^{(2t+y)}$ as $\mathbf{v}'$.

Figure 6.3 shows the visualization of $\mathbf{v}$ and $\mathbf{v}'$ for *Food* aspect on both *Positive* and *Negative* sentiment. It can be observed that the learned embedding $\mathbf{v}'$ in the aspect-specific dictionary shifts towards the embedding of aspect-specific sentiment. For *Positive* sentiment,

Table 6.2: Performance of sentiment classification (%).

Data set	Method	A	P	R	F_1
Restaurant	CosSim	77.23	75.57	74.92	75.20
	ASUM	71.31	69.47	70.16	69.72
	ASA	82.06	80.73	80.64	80.68
	JASA	**82.34**	**80.95**	**81.31**	**81.12**
Laptop	CosSim	73.03	74.31	74.40	73.03
	ASUM	67.26	66.67	66.75	66.70
	ASA	74.30	**74.83**	**75.23**	74.26
	JASA	**74.42**	74.46	74.94	**74.30**



(a) *Food* aspect, *Positive* sentiment

(b) *Food* aspect, *Negative* sentiment

Figure 6.3: Visualization of learned embedding $\mathbf{v}'$ in the dictionary $\mathbf{D}_{AS}$ on certain aspect and sentiment comparing to the average embedding of seed words $\mathbf{v}$.

as presented in Figure 6.3(a), $\mathbf{v}'$ drifts towards words like "delicious", "tasty" and "yummy", which are specifically used to compliment *Food* aspect. For *Negative* sentiment (Figure 6.3(b)), $\mathbf{v}'$ is more close to words like "bland" and "flavorless", which are common words to criticize *Food* aspect. On the other hand, $\mathbf{v}'$ does not completely distance itself from the average seed embedding $\mathbf{v}$ due to the regularization. This is because $\mathbf{v}'$ still needs to reflect the general sentiment words like "good" or "terrible".

We also try to identify the sentiment words that gain the most "density" by learning the aspect-specific sentiment dictionary. To be specific, we find words with the most boosted density from a kernel function centered at $\mathbf{v}$ to another kernel function centered at $\mathbf{v}'$. We instantiate the kernel function as

$$K_{\mathbf{v}}(\mathbf{e}) = \exp\Big(\text{sim}(\mathbf{e}, \mathbf{v})\Big)$$

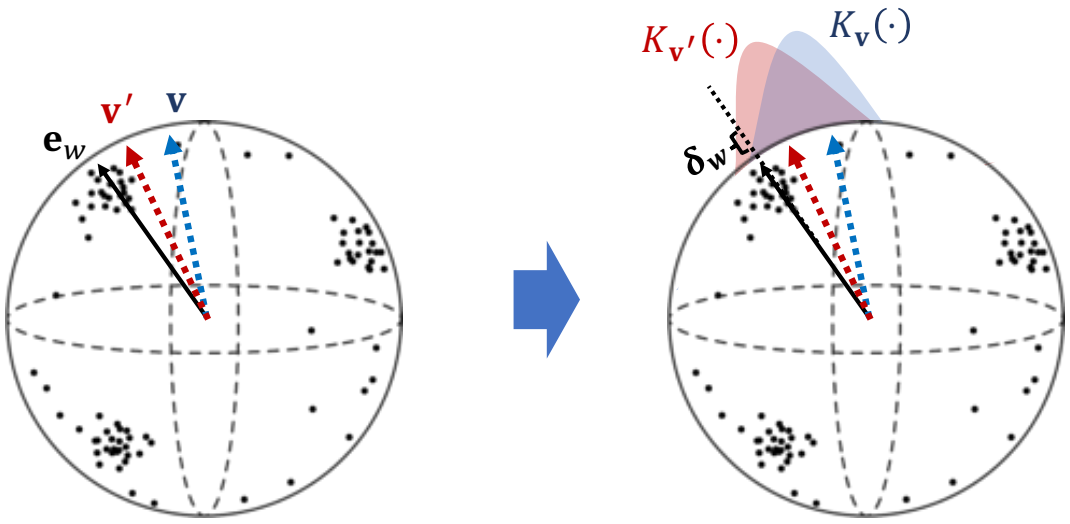


Figure 6.4: Example of how δ_w is calculated in the embedding space. Suppose the word w is "delicious". The black vector represents the embedding vector $\mathbf{e}_w$. The blue and red vectors represent the average embedding of seed words $\mathbf{v}$ and the learned embedding in the dictionary $\mathbf{v}'$ respectively. The boosted density δ_w is calculated by the difference between two kernel functions centered at $\mathbf{v}'$ and $\mathbf{v}$.

which is proportional to the probability density function of a von Mises-Fisher distribution with the concentration parameter as 1. For each word $w \in \mathcal{V}$, we calculate the difference of densities by:

$$\delta_w = K_{\mathbf{v}'}(\mathbf{e}_w) - K_{\mathbf{v}}(\mathbf{e}_w)$$

and rank the words by δ_w from the largest to the smallest.

Table 6.3 presents the top-3 words for each aspect-sentiment pair. As one can see, many words are aspect-specific sentiment words. For example, for aspect *Food* with *Positive* sentiment, the top-ranked words are "delicious", "tasty" and "yummy", while for its *Negative* sentiment, words like "bland", "flavorless" are ranked high.

Case study. We compare the output of JASA and ASA to take an in-depth look at how JASA outperforms ASA. Figure 6.5 shows the aspect attention weights α_k^A's on two sentences where JASA makes the correct classification while ASA fails.

The first sentence is "Be sure to try the seasonal, and always delicious, specials." As Figure 6.5(a) shows, ASA places the most attention on the word "specials", which is sufficient to narrow down the possible aspect classes to *Food* or *Drinks*, but not enough to pinpoint the correct aspect class. In contrast, JASA places much more attention on the word "delicious", which is usually a sentiment word, but can also imply the aspect class since it is specifically

Table 6.3: Comparing the learned embedding in dictionary $\mathbf{D}_{AS}$ and the average seed embedding. The table lists words with the most boosted densities δ_w in a cosine-similarity-based kernel function by moving the center from the average embedding of seed words $\mathbf{v}$ to the learned embedding in the dictionary $\mathbf{v}'$.

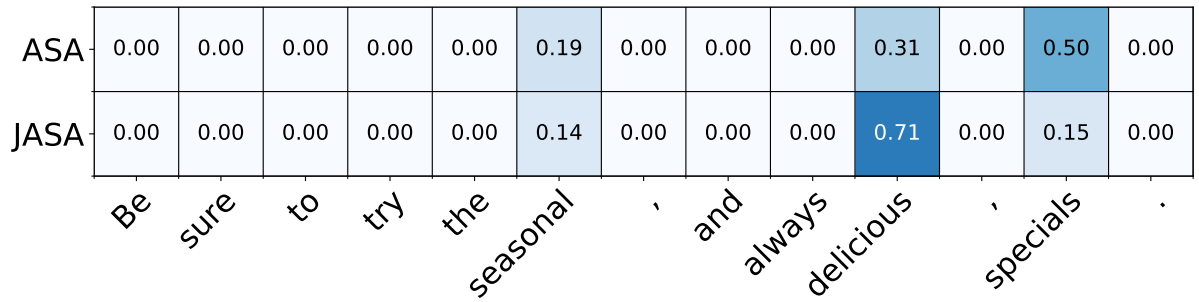
Aspect	Sentiment	Words with largest δ_w
Location	Positive	cozy, sultry, nice
	Negative	ventured, seattle, resident
Drinks	Positive	awesome, huckleberry, boozy
	Negative	vomit, substance, clump
Food	Positive	delicious, tasty, yummy
	Negative	yuck, bland, flavorless
Ambience	Positive	nice, cool, clean
	Negative	vomit, enemy, dishonest
Service	Positive	friendly, courteous, personable
	Negative	frustrated, time, acknowledgement

used to describe food. Hence, JASA correctly classify the sentence into *Food* aspect class with a high confidence (with predictive probability of 0.99). This is because the joint structure of aspect and sentiment in JASA allows the aspect attention mechanism and reconstruction mechanism to fit the training objective of sentiment representation reconstruction, which enables the aspect-specific sentiment words to benefit the aspect classification in such cases.

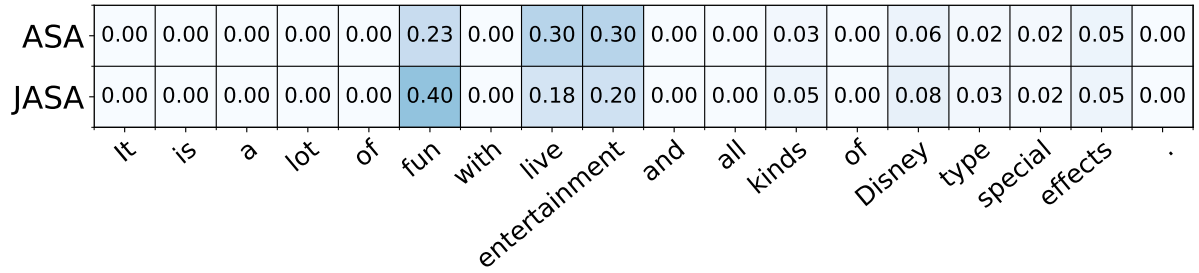
Another example is the sentence "It is a lot of fun with live entertainment and all kinds of Disney type special effects." This sentence is labeled with the *Ambience* aspect and the *Positive* sentiment. Figure 6.5(b) shows that ASA does not have a particularly strong attention on one specific word. Instead, its attention scores almost evenly distributed on words "fun", "live" and "entertainment", and mistakenly classifies the sentences into *Location* aspect. Again, JASA shifts more attention on the word "fun". Since "fun" is more frequently related to the *Ambience* aspect, JASA is able to output the correct aspect label *Ambience*.

6.6 SUMMARY

In this chapter, we study to develop sentence-level aspect-based sentiment analysis with minimal user guidance, where users only need to provide a small set of seed words for each aspect class and each sentiment class as well as an unlabeled corpus of reviews. We utilize the autoencoder structure with its dictionary regularized by user-provided seed words for both aspect and sentiment modeling. We also propose a model with joint aspect and sentiment encoder to capture aspect-specific sentiment words. The experimental results show the effectiveness of our model on two real world data sets.



(a) Aspect attention derived by ASA and JASA of a sentence on *Food* aspect with *Positive* sentiment. ASA mistakenly classifies the sentence into *Drinks* aspect.



(b) Aspect attention derived by ASA and JASA of a sentence on *Ambience* aspect with *Positive* sentiment. ASA mistakenly classifies the sentence into *Location* aspect.

Figure 6.5: Case study on aspect attention.

There are also numerous applications. For example, one can develop an aspect-based sentiment outlier analysis tool to discover potential outbreak of complaints from customer feedback of a certain product on a specific aspect.

CHAPTER 7: CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

We propose to leverage word embedding — a technique that maps each word to a vector representation — in multiple text mining scenarios. We propose several models based on word embedding to characterize the corpus and to extract meaningful knowledge from the text data. The models are designed for different scenarios, but share the following designing principles:

1. *Representing each text object as a bag of embedding vectors.* Instead of using the bag-of-word representation, leveraging the embedding vectors can provide more signals about the semantic proximity between words and allow more flexible corpus modeling in the embedding space.

2. *Focusing on the most informative part of the text data and filtering the noise.* Text data usually carries rich information, but not necessarily all signals are helpful for the target application.

3. *Identifying a mixture of vectors in the embedding space that can best represent the corpus.* They are usually derived by identified by maximizing the likelihood or minimizing the loss of recovering the corpus from these vectors. The identified vectors indicate the regions where embedding vectors in the corpus most frequently appear. If the target application has a given target class space (*e.g.* positive v.s. negative in sentiment analysis), each identified vector can be mapped to a target class.

A benefit of integrating word embedding into text mining tasks is the better handling of words with similar semantics. In traditional text mining, the frequencies of words with similar semantic meanings will be calculated separately. With word embedding, their frequencies can be considered collectively, leading to better modeling. On the other hand, the corpus representation in the embedding space allows words that are unseen in the modeling corpus to have nontrivial predictive probabilities determined by their positions in the embedding space.

Another benefit is that the embedding space provides a natural bridge for users to provide word-level supervision. The supervision usually serves as a guidance to shape the learned structure to be more aligned with human knowledge. Since the corpus representation and the user-provided words are both in the same embedding space, it is intuitive to leverage

word-level supervision from users in text mining tasks. Even with only few words from users, with the semantic similarity captured by word embedding as well as the frequency signals from the corpus, we can still develop text mining methods to achieve certain text mining tasks that traditionally requires massive labeled data for training.

We found that the three modules of representing text as bag of word embedding, filtering noise from textual signal, and representing corpus by a mixture of vectors in the embedding space serve as a pragmatic guideline to get rapid traction for novel or traditionally difficult text mining tasks. Notice that this three modules do not need to be performed specifically in order. Multiple parts can be jointly performed.

Discussions. The notation of "topic" in traditional text mining techniques is usually instantiated by a multinomial distribution over all the possible words. Words with different semantic meaning can have high probabilities in the same topic as long as they frequently co-occur in the same document. For example, "health" and "insurance" are not necessarily close in semantics, but may still appear in the same topic as they are related. However, the mixture of vectors representing the corpus in the embedding space usually depict several more fine-grained and specific clusters of words. More often, synonyms or words within the same category will be grouped together. For example, "health" and "fitness" will be grouped together.

This property helps in some text mining tasks such as lexicon expansion, while it may also imply potential "non-informative" vectors such as a vector close to "monday", "tuesday" and "friday". We emphasize the importance of focusing on the words relevant to the task by various methods, including removing non-informative semantic regions, or constructing and utilizing lexicons as filters.

Most of our proposed methods are not designed for a specific text object or domain, which means they can be applied to any text objects (such as sentences, paragraphs or documents) in any domains. In general, with a sufficiently large unlabeled corpus, it is always possible to train reasonable quality word embedding and perform our methods. Some specific technical part might target problems of a specific type of text object. For example, the outlierness measure specifically tackles the noise issue in longer text objects like documents.

7.2 FUTURE WORK

There are a number of interesting directions to explore in utilizing word embedding for text mining tasks. First, there could be a better way to handle words that are unseen in word embedding training corpus and hence do not have embedding vectors. Instead of simply

discarding these words, it might be more reasonable to estimate their embedding vectors by their surrounding words. Second, it might be worth to utilize a general background corpus to capture those corpus-specific informative words. Some words like "monday", "friday" likely have similar occurring probabilities in the background corpus as in the given corpus and therefore should be regarded as non-informative. Moreover, it is also worthwhile to explore the potential to generalize our methods for different tasks into one formalized framework, which could greatly facilitate the development of text mining algorithms for other tasks.

Specifically in the task of outlier document detection, there are also some potential future work. For example, if the algorithm can automatically generate an explanation for each mined outlier document, it may greatly help analysts understand the insights. In addition, there are also various downstream applications of outlier document detection to be explored. It may benefit fraud detection or spam filtering by enabling better exploitation of text data, or serve as an interesting exploration application.

For sentiment analysis, our methods are yet to leverage all the signals from the unlabeled corpus that can potentially benefit the aspect-based sentiment analysis. We list some possible future directions: 1) Leveraging the frequent patterns between aspect and sentiment words in the form of a sequence of word embedding vectors. Since aspect and sentiment words usually appears with semantically similar patterns (*e.g.*, *[aspect]* is *[sentiment]*, *[aspect]* was *[sentiment]*), utilizing these signals to enhance the traditional frequent pattern would be promising. 2) Fine-grained dictionary for aspect: With the improvement we observed by expanding the sentiment dictionary into aspect-specific sentiment dictionary, it is natural to consider the idea of further expanding the aspect dictionary to be more fine grained. In some cases, a user-defined aspect contains multiple concepts in the embedding space (*e.g.*, hardware aspect of laptop may include mouse, keyboard, graphic, CPU, *etc.*). This can be better characterized by a fine-grained aspect dictionary.

REFERENCES

[1] N. Abe, B. Zadrozny, and J. Langford, "Outlier detection by active learning," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2006, pp. 504–509.

[2] C. C. Aggarwal, *Outlier Analysis*. Springer, 2016.

[3] C. C. Aggarwal and P. S. Yu, "Outlier detection with uncertain data," in *Proceedings of the 2008 SIAM International Conference on Data Mining (SDM)*, 2008, pp. 483–493.

[4] C. C. Aggarwal and P. S. Yu, "On clustering massive text and categorical data streams," *Knowledge and Information Systems*, vol. 24, no. 2, pp. 171–196, 2010.

[5] A. Agresti and B. A. Coull, "Approximate is better than "exact" for interval estimation of binomial proportions," *The American Statistician*, vol. 52, no. 2, pp. 119–126, 1998.

[6] S. Angelidis and M. Lapata, "Multiple instance learning networks for fine-grained sentiment analysis," *Transactions of the Association of Computational Linguistics*, vol. 6, pp. 17–31, 2018.

[7] S. Angelidis and M. Lapata, "Summarizing opinions: Aspect extraction meets sentiment prediction and they are both weakly supervised," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018, pp. 3675–3686.

[8] A. Antos, V. Grover, and C. Szepesvári, "Active learning in heteroscedastic noise," *Theoretical Computer Science*, vol. 411, no. 29-30, pp. 2712–2728, 2010.

[9] M. Aouf and L. A. Park, "Approximate document outlier detection using random spectral projection," in *Advances in Artificial Intelligence*, 2012, pp. 579–590.

[10] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[11] K. Batmanghelich, A. Saeedi, K. Narasimhan, and S. Gershman, "Nonparametric spherical topic modeling with word embeddings," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016, pp. 537–542.

[12] K. Bauman, B. Liu, and A. Tuzhilin, "Aspect based recommendations: Recommending items with the most valuable aspects based on user reviews," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2017, pp. 717–725.

[13] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," in *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2001, pp. 601–608.

[14] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[15] M. M. Breunig, H. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2000, pp. 93–104.

[16] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 15:1–15:58, 2009.

[17] D. Chen and C. Manning, "A fast and accurate dependency parser using neural networks," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 740–750.

[18] L. Chen, G. Chen, and F. Wang, "Recommender systems based on user reviews: the state of the art," *User Modeling and User-Adapted Interaction*, vol. 25, no. 2, pp. 99–154, 2015.

[19] S. X. Chen and J. S. Liu, "Statistical applications of the poisson-binomial and conditional bernoulli distributions," *Statistica Sinica*, pp. 875–892, 1997.

[20] S. Chen, T. Lin, I. King, M. R. Lyu, and W. Chen, "Combinatorial pure exploration of multi-armed bandits," in *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 379–387.

[21] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th International Conference on Machine Learning (ICML)*, 2008, pp. 160–167.

[22] R. Das, M. Zaheer, and C. Dyer, "Gaussian LDA for topic models with word embeddings," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL)*, 2015, pp. 795–804.

[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[24] S. Gopal and Y. Yang, "Von mises-fisher clustering models," in *Proceedings of the 31th International Conference on Machine Learning (ICML)*, 2014, pp. 154–162.

[25] D. Guthrie, "Unsupervised detection of anomalous text," Ph.D. dissertation, University of Sheffield, 2008.

[26] M. Hauskrecht, I. Batal, M. Valko, S. Visweswaran, G. F. Cooper, and G. Clermont, "Outlier detection for patient monitoring and alerting," *Journal of Biomedical Informatics*, vol. 46, no. 1, pp. 47–55, 2013.

[27] R. He, W. S. Lee, H. T. Ng, and D. Dahlmeier, "An unsupervised neural attention model for aspect extraction," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017, pp. 388–397.

[28] R. He, W. S. Lee, H. T. Ng, and D. Dahlmeier, "Exploiting document knowledge for aspect-level sentiment classification," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018, pp. 579–585.

[29] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *Proceedings of the 25th International Conference on World Wide Web (WWW)*, 2016, pp. 507–517.

[30] V. J. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126, 2004.

[31] T. Hofmann, "Probabilistic latent semantic indexing," in *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 1999, pp. 50–57.

[32] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2004, pp. 168–177.

[33] B. Jiang and J. Pei, "Outlier detection on uncertain data: Objects, instances, and inferences," in *Proceedings of the 27th International Conference on Data Engineering (ICDE)*, 2011, pp. 422–433.

[34] W. Jin, H. H. Ho, and R. K. Srihari, "OpinionMiner: a novel machine learning system for web opinion mining and extraction," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2009, pp. 1195–1204.

[35] W. Jin, A. K. Tung, and J. Han, "Mining top-n local outliers in large databases," in *Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2001, pp. 293–298.

[36] Y. Jo and A. H. Oh, "Aspect and sentiment unification model for online review analysis," in *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining (WSDM)*, 2011, pp. 815–824.

[37] N. Kalchbrenner and P. Blunsom, "Recurrent continuous translation models," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2013, pp. 1700–1709.

[38] J. Kamps, M. Marx, R. J. Mokken, and M. d. Rijke, "Using WordNet to measure semantic orientations of adjectives," in *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC)*, 2004, pp. 1115–1118.

[39] S. P. Kasiviswanathan, G. Cong, P. Melville, and R. D. Lawrence, "Novel document detection for massive data streams using distributed dictionary learning," *IBM Journal of Research and Development*, vol. 57, no. 3-4, pp. 9:1–9:15, May 2013.

[40] S. P. Kasiviswanathan, H. Wang, A. Banerjee, and P. Melville, "Online l1-dictionary learning with application to novel document detection," in *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 2258–2266.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[42] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," in *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 3294–3302.

[43] E. M. Knorr and R. T. Ng, "A unified notion of outliers: Properties and computation," in *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD)*, 1997, pp. 219–222.

[44] E. M. Knorr and R. T. Ng, "Algorithms for mining distance-based outliers in large datasets," in *Proceedings of 24th International Conference on Very Large Data Bases (VLDB)*, 1998, pp. 392–403.

[45] G. Kollios, D. Gunopulos, N. Koudas, and S. Berchtold, "Efficient biased sampling for approximate clustering and outlier detection in large data sets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 5, pp. 1170–1187, 2003.

[46] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on International Conference on Machine Learning (ICML)*, 2014, pp. 1188–1196.

[47] F. Li, C. Han, M. Huang, X. Zhu, Y.-J. Xia, S. Zhang, and H. Yu, "Structure-aware review mining and summarization," in *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, 2010, pp. 653–661.

[48] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation forest," in *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM)*, 2008, pp. 413–422.

[49] H. Liu, Y. Zhang, B. Deng, and Y. Fu, "Outlier detection via sampling ensemble," in *IEEE International Conference on Big Data (Big Data)*, 2016, pp. 726–735.

[50] J. Liu, J. Shang, C. Wang, X. Ren, and J. Han, "Mining quality phrases from massive text corpora," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2015, pp. 1729–1744.

[51] P. Liu, S. Joty, and H. Meng, "Fine-grained opinion mining with recurrent neural networks and word embeddings," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015, pp. 1433–1443.

[52] Q. Liu, Z. Gao, B. Liu, and Y. Zhang, "Automated rule selection for aspect extraction in opinion mining," in *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI)*, 2015, pp. 1291–1297.

[53] Q. Liu, B. Liu, Y. Zhang, D. S. Kim, and Z. Gao, "Improving opinion aspect extraction using semantic similarity and aspect associations," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, 2016, pp. 2986–2992.

[54] X. Liu, Y. Shen, K. Duh, and J. Gao, "Stochastic answer networks for machine reading comprehension," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, vol. 1, 2018, pp. 1694–1704.

[55] Y. Lu, M. Castellanos, U. Dayal, and C. Zhai, "Automatic construction of a context-aware sentiment lexicon: an optimization approach," in *Proceedings of the 20th International Conference on World Wide Web (WWW)*, 2011, pp. 347–356.

[56] I. Mani, *Advances in automatic text summarization*. The MIT press, 1999.

[57] D. Marcheggiani, O. Täckström, A. Esuli, and F. Sebastiani, "Hierarchical multi-label conditional random fields for aspect-oriented opinion mining," in *Proceedings of European Conference on Information Retrieval (ECIR)*, 2014, pp. 273–285.

[58] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, "Image-based recommendations on styles and substitutes," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2015, pp. 43–52.

[59] M. L. McHugh, "The Chi-square test of independence," *Biochemia Medica*, vol. 23, no. 2, pp. 143–149, 2013.

[60] Q. Mei, X. Ling, M. Wondra, H. Su, and C. Zhai, "Topic sentiment mixture: modeling facets and opinions in weblogs," in *Proceedings of the 16th International Conference on World Wide Web (WWW)*, 2007, pp. 171–180.

[61] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2013, pp. 3111–3119.

[62] A. Mnih and G. E. Hinton, "A scalable hierarchical distributed language model," in *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2009, pp. 1081–1088.

[63] S. Moghaddam and M. Ester, "Opinion digger: an unsupervised opinion miner from unstructured product reviews," in *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM)*, 2010, pp. 1825–1828.

[64] A. Neelakantan, J. Shankar, A. Passos, and A. McCallum, "Efficient non-parametric estimation of multiple embeddings per word in vector space," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1059–1069.

[65] T. H. Nguyen, K. Shirai, and J. Velcin, "Sentiment analysis on social media for stock movement prediction," *Expert Systems with Applications*, vol. 42, no. 24, pp. 9603–9611, 2015.

[66] M. Nickel and D. Kiela, "Poincaré embeddings for learning hierarchical representations," in *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 6341–6350.

[67] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[68] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2018, pp. 2227–2237.

[69] M. Pontiki, D. Galanis, H. Papageorgiou, I. Androutsopoulos, S. Manandhar, A.-S. Mohammad, M. Al-Ayyoub, Y. Zhao, B. Qin, O. De Clercq et al., "Semeval-2016 task 5: Aspect based sentiment analysis," in *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval)*, 2016, pp. 19–30.

[70] A.-M. Popescu and O. Etzioni, "Extracting product features and opinions from reviews," in *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, 2005, pp. 339–346.

[71] G. Qiu, B. Liu, J. Bu, and C. Chen, "Opinion word expansion and target extraction through double propagation," *Computational Linguistics*, vol. 37, no. 1, pp. 9–27, 2011.

[72] X. Rong, Z. Chen, Q. Mei, and E. Adar, "Egoset: Exploiting word ego-networks and user-generated ontology for multifaceted set expansion," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining (WSDM)*, 2016, pp. 645–654.

[73] C. Scaffidi, K. Bierhoff, E. Chang, M. Felker, H. Ng, and C. Jin, "Red opal: product-feature scoring from reviews," in *Proceedings of the 8th ACM Conference on Electronic Commerce (EC)*, 2007, pp. 182–191.

[74] K. Schouten and F. Frasincar, "Survey on aspect-level sentiment analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 3, pp. 813–830, 2016.

[75] J. Shen, Z. Wu, D. Lei, J. Shang, X. Ren, and J. Han, "Setexpan: Corpus-based set expansion via context feature selection and rank ensemble," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)*, 2017, pp. 288–304.

[76] J. Si, A. Mukherjee, B. Liu, Q. Li, H. Li, and X. Deng, "Exploiting topic based twitter sentiment for stock prediction," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2013, pp. 24–29.

[77] R. Socher, J. Bauer, C. D. Manning et al., "Parsing with compositional vector grammars," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, vol. 1, 2013, pp. 455–465.

[78] X. Song, M. Wu, C. Jermaine, and S. Ranka, "Conditional anomaly detection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 5, pp. 631–645, 2007.

[79] M. Sugiyama and K. Borgwardt, "Rapid distance-based outlier detection via sampling," in *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2013, pp. 467–475.

[80] D. Tang, B. Qin, and T. Liu, "Aspect level sentiment classification with deep memory network," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016, pp. 214–224.

[81] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: extraction and mining of academic social networks," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2008, pp. 990–998.

[82] I. Titov and R. T. McDonald, "A joint model of text and aspect ratings for sentiment summarization," in *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, 2008, pp. 308–316.

[83] J. Turian, L. Ratinov, and Y. Bengio, "Word representations: a simple and general method for semi-supervised learning," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2010, pp. 384–394.

[84] H. Wang, Y. Lu, and C. Zhai, "Latent aspect rating analysis on review text data: a rating regression approach," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2010, pp. 783–792.

[85] H. Wang, Y. Lu, and C. Zhai, "Latent aspect rating analysis without aspect keyword supervision," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2011, pp. 618–626.

[86] J. Wang, L.-C. Yu, K. R. Lai, and X. Zhang, "Dimensional sentiment analysis using a regional CNN-LSTM model," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016, pp. 225–230.

[87] Y. Wang, M. Huang, X. Zhu, and L. Zhao, "Attention-based LSTM for aspect-level sentiment classification," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016, pp. 606–615.

[88] Y. Wang and S. Yang, "Outlier detection from massive short documents using domain ontology," in *IEEE International Conference on Intelligent Computing and Intelligent Systems*, vol. 3, 2010, pp. 558–562.

[89] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu, "Charagram: Embedding words and sentences via character n-grams," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016, pp. 1504–1515.

[90] M. Wu and C. Jermaine, "Outlier detection by sampling with accuracy guarantees," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2006, pp. 767–772.

[91] D. Yogatama, D. Gillick, and N. Lazic, "Embedding methods for fine grained entity type classification," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, vol. 2, 2015, pp. 291–296.

[92] C. Zhang, F. Tao, X. Chen, J. Shen, M. Jiang, B. Sadler, M. Vanni, and J. Han, "Taxogen: Unsupervised topic taxonomy construction by adaptive term embedding and clustering," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2018, pp. 2701–2709.

[93] J. Zhang, Z. Ghahramani, and Y. Yang, "A probabilistic model for online document clustering with application to novelty detection," in *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2004, pp. 1617–1624.

[94] L. Zhang, B. Liu, S. H. Lim, and E. O'Brien-Strain, "Extracting and ranking product features in opinion documents," in *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, 2010, pp. 1462–1470.

[95] Y. Zhang, J. Callan, and T. Minka, "Novelty and redundancy detection in adaptive filtering," in *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2002, pp. 81–88.

[96] Y. Zhao, B. Qin, S. Hu, and T. Liu, "Generalizing syntactic structures for product attribute candidate extraction," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2010, pp. 377–380.

[97] H. Zhuang, C. Wang, F. Tao, L. Kaplan, and J. Han, "Identifying semantically deviating outlier documents," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017, pp. 2748–2757.

[98] A. Zimek, M. Gaudet, R. J. Campello, and J. Sander, "Subsampling for efficient and effective unsupervised outlier detection ensembles," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2013, pp. 428–436.

[99] C. Zirn, M. Niepert, H. Stuckenschmidt, and M. Strube, "Fine-grained sentiment analysis with structural features," in *Proceedings of 5th International Joint Conference on Natural Language Processing (IJCNLP)*, 2011, pp. 336–344.