Open Classification and Change Detection in the Similarity Space

by

Geli Fei
B.S., Software Engineering, Harbin Institute of Technology, 2011

THESIS

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 Chicago, 2017

Chicago, Illinois

Defense Committee:
Bing Liu, Chair and Advisor
Barbara Di Eugenio
Piotr Gmytrasiewicz
Philip S. Yu
Jalal Mahmud, IBM Research - Almaden

This thesis is dedicated to my family, teachers and friends.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my Ph.D. advisor Professor Bing Liu for his dedication, encouragement and support during all of my Ph.D. years. Working with him to pursue my Ph.D. degree is my greatest honor. This dissertation and all my Ph.D. research would not have been done without his guidance and advice. His enthusiasm in pursuing unexplored areas in academic research, his kindness to his students, and his dedication to his family have always inspired me. I will always be grateful to his mentoring and advising without any reservation.

I would also like to thank Professor Philip S. Yu, Professor Barbara Di Eugenio, Professor Piotr Gmytrasiewicz and Doctor Jalal Mahmud for their valuable time. Their constructive comments and suggestions contributed a lot to this thesis. Special thanks to Professor Philip S. Yu and Doctor Jalal Mahmud, who have helped me in many ways during my Ph.D. study.

During my Ph.D. study, I have been fortunate enough to make many friends, who not only helped me through discussions and collaborations, but also made my Ph.D. life enjoyable and less stressful. I wish them all the best in pursuing their current studies and future careers.

Last but not least, I would like to thank my family for their unconditional love and support. I have been blessed to be born and raised up in a warm family, which gives me freedom to pursue my life in every aspect with full support and understanding. I want to thank my parents and grandparents for everything they have given to raise me up to the way I currently am.

GF

CONTRIBUTION OF AUTHORS

Chapter 2 presents a published manuscript (Fei and Liu, 2015) for which I was the primary author. My advisor Professor Bing Liu contributed to the discussions about the ideas and assisted in revising the manuscript.

Chapter 3 presents a published manuscript (Fei and Liu, 2016) for which I was the primary author. My advisor Professor Bing Liu contributed to the discussions with respect to the preliminary ideas and helped revise the manuscript.

Chapter 4 presents a published manuscript (Fei et al., 2016) for which I was the primary author. Shuai Wang and my advisor Professor Bing Liu contributed to the discussions about the preliminary ideas and assisted in revising the manuscript.

Chapter 5 presents a manuscript under review (Fei et al., 2017) for which I was the primary author. Shuai Wang, my advisor Professor Bing Liu, and Professor Leman Akoglu all contributed to the discussions about the ideas and helped revise the paper.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Text categorization has developed into one of the key techniques for handling and organizing text data. However, the rapid emergence of new topics and the highly diverse nature of online text data have brought new challenges to existing classification techniques. Among many, one of the main challenges is their lack of ability in handling unseen classes of documents due to the closed world assumption that has been commonly made by existing techniques. Under this assumption, all test classes are assumed to be known at training time. However, a more realistic scenario is to expect unseen classes during testing (open world). In this case, the goal is to design a learning system that classifies documents of the known classes into their respective classes and also to identify documents from unknown classes. This problem is called open (world) classification. In this thesis, we first study three closely related research problems to open classification.

We start with studying the problem of text classification under negative covariate shift (Fei and Liu, 2015), which can be seen as a special case of open classification, because the covariate shift in our setting is mainly caused by documents of unseen negative classes in testing. In order to solve the problem, we propose a novel learning technique, called center-based similarity (CBS) space learning. We show that in the similarity space, the covariate shift problem is significantly mitigated.

Then, we proceed to study the general problem of open classification and study it from an open space risk management perspective (Fei and Liu, 2016). In particular, we propose an

open space risk formulation and employ our proposed CBS learning technique as the underlying learner. Our method is able to significantly reduce the open space risk compared to existing methods, and the proposed classifier achieves superior results to the state-of-the-art approaches.

Often times, being able to detect unseen classes/topics is still insufficient for a multi-class classifier to handle the growing number of topics of interest of users. The detected unseen classes also need to be incorporated into the existing classifier system with minimal effort. Simply re-training the entire system from scratch is only feasible if the number of classes is small. For this reason, we propose cumulative learning (Fei et al., 2016), where unseen classes of documents are not only identified, but also added into the existing system in an efficient manner. The proposed cumulative learning strategy, as a form of lifelong machine learning, mimics the process of human concept learning and utilizes existing knowledge in helping learn the new class of data.

One of the key techniques used in the above research is the transformation of documents from the traditional n-gram document space to a similarity space to detect a special type of change in the test class distribution, i.e., unseen classes, in the supervised setting. As the last part of this thesis, we explore the use of similarity-based approaches in detecting a differnt type of change in social media accounts, especially online review accounts, in an unsupervised setting. In particular, we study the problem of detecting changed-hands (CH) online review accounts (Fei et al., 2017). We propose a novel algorithm, called CHAD, which relies on transforming a sequence of reviews in an account to a set of similarity sequences for CH accounts detection and to estimate the change point.

CHAPTER 1

INTRODUCTION

Applications using social media data, such as reviews, discussion posts, and (micro) blogs are becoming increasingly popular. Text classification, being a research topic for decades, has developed into one of the most widely used techniques in helping people organize and collect such data from the Web. However, the vast diversity and rapid emergence of new topics in social media also have posed new challenges to existing text classification techniques. Among many, one of the main challenges is their lack of ability in handling unseen classes of documents because of one commonly made assumption by existing techniques, which says all classes in testing have been seen at training time. However, such an assumption is often violated in reality, which results in inferior classifiers. A more realistic scenario is to expect the unseen classes during testing (open world), and the goal is to design a learning system that is able to classify documents of the known classes into their respective classes and also to detect documents from unseen classes. This problem is called open (world) classification. Compared to research on classification with the closed world assumption, there is relatively less work on open classification.

Thus, in this thesis, we focus on three research problems related to open classification, whose core issue is the handling of unseen classes of documents.

- We first study a special case of the open classification problem, the goal of which is to build a classifier system that is able to identify the documents of a particular topic of

interest from the Web. Due to the difference in training and test class distributions, the problem is also known as covariate shift.

- Then, we proceed to study the general open classification problem, and propose a solution from an open space risk management perspective.
- We propose and study the problem of cumulative machine learning for text classification, as a new form of lifelong machine learning, where unseen classes of documents are not only detected, but also cumulatively incorporated into the existing system in an efficient manner.

One of the key techniques used in the above research is the transformation of documents from their original document space to a similarity space to deal with a specific type of change in the test class distribution, i.e., the appearance of unseen classes. As the last part of this thesis, we explore the use of similarity-based approaches in detecting a type of change in social media accounts, especially online review accounts, in an unsupervised setting. In particular, we study the problem of detecting changed-hands online review accounts from a linguistic perspective.

In summary, this thesis consists of studies on the following four research problems:

1.1 Text Classification under Negative Covariate Shift

In Chapter 2, we focus on text classification under negative covariate shift. In a typical social media content analysis task, a user is interested in analyzing posts of a particular topic. Identifying such posts is often formulated as a binary classification problem. However, often times the resulting classifier may not be satisfactory. There may be many reasons. One key

reason is the covariate shift problem. That is, the training data is not fully representative of the test data.

We observed that in an application as we mentioned above, the covariate shift problem mainly occurs in the negative data because topics discussed in social media are highly diverse and numerous, although a user may label enough positive training data, the user-labeled negative training data may cover only a small number of topics, and many unseen negative classes appear in testing. Thus, we assume that the covariate shift problem occurs mainly in the negative training and test data, and no or minimum covariate shift exists in the positive training and test data, and thus we call it negative covariate shift as a special case of covariate shift. We propose a novel technique to solve this problem. The key novelty of the technique is in the transformation of document representation from the traditional n-gram feature space to a center-based similarity (CBS) space. We found that in the CBS space, the covariate shift problem is significantly mitigated, which enables us to build much better classifiers.

1.2 Open World Classification

In Chapter 3, we study the general problem of open world classification. Existing research on multi-class text classification mostly makes the closed world assumption, which focuses on designing accurate classifiers under the assumption that all test classes are known at training time. In reality, this assumption is often violated, which results in inferior classifiers. A more realistic scenario is to expect unseen classes during testing (open world). In this case, the goal is to design a learning system that classifies documents of the known classes into their respective classes and also to detect (reject) documents from unseen classes. This problem is called

open (world) classification. We approach the problem from an open space risk management perspective by reducing the open space risk while balancing the empirical risk, and propose to generalize the center-based similarity (CBS) space learning (or CBS learning) technique to provide a novel solution to the problem.

1.3 Cumulative Learning

In Chapter 4, we go one step further from open classification and propose the problem of cumulative machine learning in the context of text classification. In classic supervised learning settings, a learning algorithm takes a training data consisting of a fixed number of classes to build a classifier. We propose to study a new problem, i.e., building a learning system that learns cumulatively. As time goes by, the system sees and learns more and more classes of data and becomes more and more knowledgeable. We believe that this is similar to human learning. We humans learn continuously, retaining the learned knowledge, identifying and learning new things, and updating the existing knowledge with new experiences. Over time, we cumulate more and more knowledge. A learning system should be able to do the same. As algorithmic learning matures, it is time to tackle this cumulative machine learning (or simply cumulative learning) problem, which is a kind of lifelong machine learning problem. It presents two major challenges. First, the system must be able to detect data from unseen classes in the test set. Second, the system needs to be able to selectively update its models whenever a new class of data arrives without re-training the whole system from scratch using the entire past and present training data. Chapter 4 proposes a novel approach and system to tackle these challenges.

1.4 Detecting Changed-Hands Online Review Accounts

In Chapter 5, we propose and study the problem of detecting changed-hands social media accounts, especially online review accounts. Social media websites, especially review websites, have become important platforms for people to express opinions towards different products or services and to make purchase decisions. However, due to profits, such websites have become the targets of various types of spammers with different agendas. With the advances of spam filtering techniques, many types of spamming activities can be caught, which forces spammers to resort to other strategies. Having a reputable account or one with a clean history can be a good cover for spamming activities. In fact, it has become prevalent that spammers buy/sell such accounts openly on the Web. We call these sold/bought accounts *changed-hands* (CH) accounts. Such accounts, seemingly benign but having changed hands, exhibit behavioral and/or linguistic changes. They are hard to detect by existing spam detection algorithms as their spamming activities are under the disguise of clean histories or high reputation scores. Chapter 5 proposes a novel algorithm to determine whether an account has changed hands and estimate the change point.

CHAPTER 2

TEXT CLASSIFICATION UNDER NEGATIVE COVARIATE SHIFT

(This chapter was previously published as "Social Media Text Classification under Negative Covariate Shift", in *The 2015 Conference on Empirical Methods on Natural Language Processing (EMNLP'15)* (Fei and Liu, 2015).)

2.1 Introduction

Applications using social media data, such as reviews, discussion posts, and (micro) blogs are becoming increasingly popular. We observed from our collaborations with social science and health science researchers that in a typical application, a researcher first needs to obtain a set of posts of a particular topic that he/she wants to study, e.g., a political issue. Keyword search is often used as the first step. However, that is not sufficient due to low precision and low recall. A post containing the keyword "politics" may not be a political post while a post that does not contain the keyword may be a political post. Thus, text classification is needed to make more sophisticated decisions to improve accuracy.

For classification, the user first starts with manually labeling a set of relevant posts (positive data) about the political issue and many irrelevant posts (negative data) not about the political issue, and then builds a classifier by running a learning algorithm, e.g. SVM or naïve Bayes. However, the resulting classifier is often times not satisfactory. There may be many reasons.

One key reason we observed is that the labeled negative training data is not fully representative of the negative test data.

Let the user-interested topic be P (positive), and the set of all other irrelevant topics discussed in a social media source be $T = \{T_1, T_2, ..., T_n\}$, which forms the negative data. n is usually large. However, due to the labor-intensive effort of manual labeling, the user can label only a certain number of training posts. Then the labeled negative training posts may cover only a small number of irrelevant topics S of T ($S \subseteq T$) as negative. Further, due to the highly dynamic nature of social media, it is probably impossible to label all possible negative topics. In testing, when posts of other negative topics in $T - S$ show up, their classification can be unpredictable. For example, in an application, the training data has no negative examples about sports. However, in testing, some sports posts show up. These unexpected sports posts may be classified arbitrarily, which results in low classification accuracy. We propose a technique to solve this problem in Section 2.3.

In machine learning, this problem is called *covariate shift*, a type of *sample selection bias*. Classic machine learning assumes that the training and the test data are drawn from the same distribution. However, this assumption may not hold in many real-world scenarios such as in our case above, i.e., the training and the test distributions are different (Heckman, 1977; Shimodaira, 2000; Zadrozny, 2004; Huang et al., 2006a; Sugiyama et al., 2008; Bickel et al., 2009). There is no general solution for the sample selection bias problem because the training and test distributions can be very different from each other (Li et al., 2010). Thus, researchers attempted to deal with special cases of the problem by making different assumptions. One commonly made assumption,

which leads to the covariate shift problem, says that the conditional distribution of the class label given a data point is identical in the training and test data (Shimodaira, 2000; Bickel et al., 2009).

In this chapter, we focus on a special case of the covariate shift problem. We assume that the co-variate shift problem occurs mainly in the negative training and test data, and no or minimum covariate shift exists in the positive training and test data. This assumption is reasonable because the user knows the type of posts/documents that s/he is looking for and can label many of them.

Following the notations in (Bickel et al., 2009), our special case of the covariate shift problem can be stated formally as follows: let the set of training examples be $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_k, y_k)\}$, where $\boldsymbol{x}_i$ is the data/feature vector and y_i is the class label of $\boldsymbol{x}_i$. Let the set of test cases be $\{\boldsymbol{x}_{k+1}, \boldsymbol{x}_{k+2}, \ldots, \boldsymbol{x}_n\}$, which have no class labels. Since we are interested in binary classification, y_i is either $+1$ (positive class) or -1 (negative class). The labeled training data and the unseen test data have the same target conditional distribution $p(y|\boldsymbol{x})$ and the marginal distributions of the positive data in both the training and testing are also the same. But the marginal distributions of the negative data in the training and testing are different, i.e., $p_L(\boldsymbol{x}^-) \neq p_T(\boldsymbol{x}^-)$, where L, T, and $-$ represent the labeled training data, test data, and the negative class respectively.

Existing methods for addressing the covariate shift problem basically work as follows (more details in Section 2.2). First, statistical methods are employed to estimate the training data bias using the given test data. Then, a classifier is trained and the estimated bias of the original training set is taken into account by giving the training instances different weights. Requiring

the test data to be available in training is, however, a major weakness. In the social media post classification setting, the system needs to constantly classify the incoming data. It is infeasible to perform training constantly.

In this chapter, we propose a novel learning technique that does not need the test data to be available during training due to the specific nature of our problem, i.e., the positive training data does not have the covariate shift issue.

One obvious solution to this problem is one-class classification (Schölkopf et al., 2000; Tax and Duin, 1999b), e.g., One-class SVM. We simply discard the negative training posts/documents completely because they have the covariate shift problem. Although this is a valid solution, as we will see in the evaluation section, the models built based on One-class SVM perform poorly. Although it is conceivable to use an unsupervised method such clustering, SVD (Alter et al., 2000) or LDA (Blei et al., 2003), supervised learning usually give much higher accuracy.

In our proposed method, instead of performing supervised learning in the original document space based on n-grams, we perform learning in a similarity space. Thus, the key novelty of the method is the transformation from the original *document space* (DS) to a *center-based similarity* (CBS) space. In the new space, the covariate shift problem is significantly mitigated, which enables us to build more accurate classifiers. The reason for this is that in CBS based learning the vectors in the similarity space enable SVM (which is the learning algorithm that we use) to find a good boundary of the positive class data based on similarity and to separate it from all possible negative class data, including those negative data that is not represented in training.

We will explain this in greater detail in Section 2.3 after we present the proposed algorithm, which we call CBS-L (for CBS Learning). Our contributions can be summarized as follows:

1. First, it formulates a special case of the covariate shift problem. This case occurs frequently in social media data classification as we discussed above.
2. Second, it proposes a novel algorithm in the CBS space, CBS-L, which avoids the covariate shift problem to a large extent because it is able to find a good similarity boundary of the positive data.
3. Third, it experimentally demonstrates the effectiveness of the proposed method.

2.2 Related Work

Traditional supervised learning assumes that the training examples and the test instances are drawn from the same distribution. However, this assumption does not always hold in many applications. This is especially the case for social media data because of the high topic diversity and constant changes of topics. This problem is known as *covariate shift*, which is a form of sample selection bias. In this section, we will cover several related research areas including sample selection bias, one-class classification, document representations, and PU learning.

Heckman (1997) for the first time introduced the concept of sample selection bias in the field of econometrics. It drew the attention of researchers in the machine learning community thanks to the work of Zadrozny (2004). As we have mentioned earlier, the commonly used approach in machine learning is to first use statistical methods to estimate the training data bias using the given test data, and then train a classifier, taking into account the estimated bias of the original training set by giving the training instances different weights (Bickel et al., 2009). For example,

Shimodaira (2000) and Sugiyama and Müller (2005) employed kernel density estimation to estimate the training and test data distributions, which were subsequently used to compute a density ratio in order to generate weights for training examples. Alternative approaches include (Dudík et al., 2005; Bickel and Scheffer, 2007), which used maximum entropy density estimation. Huang et al. (2006a) proposed kernel mean matching. The idea of minimizing the Kullback-Leibler divergence between the test and the weighted training distributions was proposed in (Sugiyama et al., 2008; Tsuboi et al., 2009) in order to compute the weights for the training examples. And an integrated approach was proposed in (Bickel et al., 2009). As we can see, all the above techniques require test data at the training time, which is a major weakness of the above mentioned approaches for social media data classification. The proposed technique CBS-L doesn't have this restriction.

As mentioned in the introduction, one-class classification is a natural and suitable approach to solve the problem. Schölkopf et al. (1999) introduced the One-class SVM for one-class classification based on the SVM methodology, which assumes that the origin defined by a kernel function is the only data point in the negative class. The objective is to separate the positive data points from the origin with the largest margin. The final classification function f gives the value $+1$ in the region where the majority of the (positive) training data resides, and -1 elsewhere. Let $X = \{x_1, x_2, \ldots, x_m\}$ be the training data from a single class, Ψ be a

kernel function that transform the training data to a different space. One-class SVM solves the following quadratic programming problem for w and p to learn the classification function f.

$$\min \frac{1}{2}\|w\|^2 + \frac{1}{vm}\sum_{i=1}^{m}\xi_i - p, \tag{2.1}$$

subject to

$$(w \cdot \Psi(x_i)) \geq p - \xi_i, \quad \xi_i \geq 0, \quad i = 1, 2, \ldots, m \tag{2.2}$$

where p is the offset of the hyperplane in the transformed feature space defined by the kernel Ψ, and ξ_i is a slack variable for each training data point x_i. The regularization parameter $v \in (0, 1]$ is used to control the trade-off between training error and the smoothness of the model, and thus has an effect on the chosen support vectors. Among many, RBF and other Gaussian kernels are the most commonly used kernels for One-class SVM. Although they achieve better performance compared to using other kernels (Manevitz and Yousef, 2001), they often lead to over-fitting of the training data and are very sensitive to parameters.

Other than the One-class SVM, many other techniques have been proposed for one-class classification. Tax and Duin (1999a; 1999b) proposed Support Vector Data Description (SVDD). SVDD seeks to find a hypersphere around the positive data that encompasses the training data points with minimum radius. It has been shown that the use of Gaussian kernel makes SVDD and One-class SVM equivalent, and the results reported in (Khan and Madden, 2013) demonstrate that SVDD and One-class SVM are indeed comparable. In order to balance between

model over-fitting and under-fitting for one-class classifiers, Tax and Duin (2002) proposed a method that tries to use artificially generated outliers to optimize the model parameters. However, as suggested in their paper, the proposed procedure is only feasible for generating artificial outliers in a hyper-sphere with up to 30 dimensions (Khan and Madden, 2013). Also, as pointed out by Khan and Madden (2010; 2013), one drawback of their methods is that they often require a large dataset and the methods become very inefficient in high dimensional feature spaces. Since text documents are usually represented in a much higher dimensional space, these methods are less suitable for text applications. Manevitz and Yousef (2001) conducted extensive experiments of one-class classification on text documents using the One-class SVM proposed by Schölkopf et al. and a proposed variant called Outlier-SVM. The proposed Outlier-SVM is based on identifying outlier data that are representative of the second class. Instead of assuming the origin is the only member of the outlier class, it assumes those data points with few non-zero entries are also outliers. However, as reported in the paper, their method, Outlier-SVM, produced quite weak results. Another similar approach was proposed by Li et al. (2010) for detecting anomalies, which considers all data points that are close to the origin as outliers. Liu and Madden (2007) and Tian and Gu (2010) tried to refine Schölkopf's model by searching optimal parameters. Luo et al., (2007) proposed a cost-sensitive One-class SVM algorithm for intrusion detection. We will see in the experiment section that one-class classification is far inferior to our proposed CBS-L method.

Our work is also related to document representations, as we propose to represent documents in the similarity space. Many alternative document representations have been proposed

in the past and have shown to perform well in many applications (Radev et al., 2000; He et al., 2004; Lebanon, 2006; Ranzato and Szummer, 2008; Wang and Domeniconi, 2008). In (Radev et al., 2000), although the centroid sentence/document vector was computed, documents were not transformed into a similarity space vector representation. Wang and Domeniconi (2008) proposed to use external knowledge to build semantic kernels for documents in order to improve text classification. Recently, distributed document representations (Le and Mikolov, 2014) have drawn much attention from the research community and have achieved state-of-the-art performance on many tasks. In our problem, the main difficulty is that testing negative documents cannot be well covered in training. And it is not clear how the enriched document representations could help solve our problem.

This work is also related to learning from positive and unlabeled examples, also known as PU learning (Denis, 1998; Yu et al., 2002; Liu et al., 2003; Lee and Liu, 2003; Elkan and Noto, 2008; Li et al., 2010). In this learning model, there is a set of labeled positive training data and a set of unlabeled data, but there is no labeled negative training data. Clearly, their setting is different from ours too. There is also no guarantee that the unlabeled data has the same distribution as the future test data.

2.3 Center-based Similarity Space Learning

We now formulate the proposed supervised learning in the CBS space, called CSB-L. The key difference between CBS learning and the classic document space (DS) learning is in the document representation, which applies to both training and testing documents or posts. In the next subsection, we first give the intuitive idea and a simple example. The detailed algo-

rithm follows. In Section 2.3.4, we explain why CBS-L is better than DS-based learning when unexpected negative data appear in the test set.

2.3.1 Basic Idea

In the proposed CBS-L formulation, each document d is no longer represented as a feature vector based on textual n-grams. Instead, it represents a set of similarity values between document d and the center of the positive documents. Specifically, the learning consists of the following steps:

1. Each document d (in the positive or negative class) is first represented with a set of document representations, i.e., *document space vectors* (*ds*-vectors) based on the document itself as in traditional text classification. Each vector denotes one representation of the document. For example, one representation may be based on only unigrams, and another representation may be based on only bigrams. For simplicity, we use only one representation/vector $\boldsymbol{x}$ (e.g., unigrams) here to represent d. Note that we use bold lower case letters to represent vectors. Each feature in a *ds*-vector is called a *ds*-feature.
2. A center vector $\boldsymbol{c}$ is then computed for each document representation for the positive class documents using the *ds*-vectors of all positive and negative documents of that representation. $\boldsymbol{c}$ is thus also a *ds*-vector.
3. Each document d in the positive and negative class is then transformed to a *center-based similarity space vector* $\mathbf{s}_d$ (called a *cbs*-vector). $\mathbf{s}_d$ consists of a set of similarity values between document d's set of *ds*-vectors, i.e., $\{\boldsymbol{x}\}$ in our case here (since we use only one

representation), and the set of corresponding positive class center vectors, i.e., $\{\boldsymbol{c}\}$ in our case:

$$\boldsymbol{s}_d = Sim(\{\boldsymbol{x}\}, \{\boldsymbol{c}\}),$$

where *Sim* is a similarity function consisting of a set of similarity measures. Each feature in $\mathbf{s}_d$ is called an *cbs*-feature. $\mathbf{s}_d$ still has the same original class label as d. Let us see an actual example. We assume that our single center vector for the positive class has been computed (see Section 2.3.2) based on the unigram representation of documents:

$$\boldsymbol{c}\text{: 1:1 2:1 6:2,}$$

where $y : z$ represents a *ds*-feature y (e.g., a word) and its feature value (e.g., term frequency, *tf*). We want to transform the following positive document d_1, represented as $\boldsymbol{x}_1$, and negative document d_2, represented as $\boldsymbol{x}_2$, to their *cbs*-vectors (the first number is the class label):

$$\boldsymbol{x}_1\text{: +1 1:2 2:1 3:1} \quad \boldsymbol{x}_2\text{: -1 2:2 3:1 5:2}$$

Assuming we apply *cosine* similarity as the first similarity metric in *Sim*, then we are able to generate a *cbs*-feature $1 : 0.50$ for d_1 (as $cosine(\boldsymbol{x}_1, \boldsymbol{c}) = 0.50$) and a *cbs*-feature $1 : 0.27$ for d_2 (as $cosine(\boldsymbol{x}_2, \boldsymbol{c}) = 0.27$). With more similarity metrics, more *cbs*-features will be generated. The final *cbs*-vectors for d_1 and d_2, represented by $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ respectively, with their class labels $+1$ and -1, are:

$$\boldsymbol{s}_1\text{: +1 1:0.50 ...} \quad \boldsymbol{s}_2\text{: -1 1:0.27 ...}$$

4. We now have a binary classification problem in the CBS space. This step simply runs a classification algorithm, e.g., SVM, to build a classifier. We use SVM in our work.

2.3.2 CBS Learning

We are given a binary text classification problem. Let $D = \{(d_1, y_1), (d_2, y_2), \ldots, (d_n, y_n)\}$ be the set of training examples, where d_i is a document and $y_i \in \{+1, -1\}$ is its class label. Traditional classification directly uses D to build a binary classifier. However, in the CBS space, we learn a classifier that returns $+1$ for documents that are "close enough" to the center of the training positive documents and -1 for documents elsewhere.

We now detail the proposed technique. As we mentioned above, instead of using one single *ds*-vector to represent a document $d_i \in D$, we use a set R_i of p *ds*-vectors $R_i = \{\boldsymbol{x}_1^i, \boldsymbol{x}_2^i, \ldots, \boldsymbol{x}_p^i\}$. Each vector $\boldsymbol{x}_j^i$ denotes one document space representation of the document, e.g., unigram representation. We then compute the center of positive training documents, which is represented as a set of p centroids $C = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_p\}$, each of which corresponds to one document space representation in R_i. The way to compute each center $\boldsymbol{c}_j$ is similar to that in the Rocchio relevance feedback method in information retrieval (Rocchio, 1971; Manning et al., 2008), which uses the corresponding *ds*-vectors of all training positive and negative documents. The detail will be given below. Based on R_i for document d_i and the center C, we can transform a document d_i from its document space representations R_i to one center-based similarity vector $\boldsymbol{cbs\text{-}v}_i$ by applying a similarity function *Sim* on each element $\boldsymbol{x}_j^i$ of R_i and its corresponding center $\boldsymbol{c}_j$. We now detail document transformation.

Training document transformation: The training data transformation from *ds*-vectors to *cbs*-vectors performs the following two steps:

Step 1: Compute the set C of centroids for the positive class. Each centroid vector $\boldsymbol{c}_j \in C$ is for one document representation $\boldsymbol{x}_j^i$. And it is computed by applying the Rocchio method to the corresponding *ds*-vectors of all documents in both positive and negative training data.

$$\boldsymbol{c}_j = \frac{\alpha}{\mid D_+ \mid} \sum_{d_i \in D_+} \frac{\boldsymbol{x}_j^i}{\mid\mid \boldsymbol{x}_j^i \mid\mid} - \frac{\beta}{\mid D - D_+ \mid} \sum_{d_i \in D - D_+} \frac{\boldsymbol{x}_j^i}{\mid\mid \boldsymbol{x}_j^i \mid\mid} \tag{2.3}$$

where D_+ is the set of documents in the positive class and $\mid \cdot \mid$ is the size function. α and β are parameters, which are usually set empirically. It is reported that using *tf-idf* representation, $\alpha = 16$ and $\beta = 4$ usually work quite well (Buckley et al. 1994). The method reduces the weights of those terms that appear in both positive and negative documents, which are considered not discriminative, by using subtraction.

Step 2: Compute the similarity vector $\boldsymbol{cbs\text{-}v}_i$ (center-based similarity space vector) for each document $d_i \in D$ based on its set of document space vectors R_i and the corresponding centroids C of the positive documents.

$$\boldsymbol{cbs\text{-}v}_i = Sim(R_d, C) \tag{2.4}$$

Sim has a set of similarity measures, and each measure m_t is applied to p document representations $\boldsymbol{x}_j^i$ in R_i and their corresponding centers in C to generate p similarity features (*cbs*-features) in $\boldsymbol{cbs\text{-}v}_i$. We discuss the *ds*-features and similarity measures for computing *cbs*-features in the next two subsections.

Complexity: The data transformation step is clearly linear in the number of examples, i.e., n.

Test document transformation: For each test document d_i, we can use step 2 above to produce a *cbs*-vector for d_i.

2.3.3 Features

DF Features. In order to compute *cbs*-features (center-based similarity space features) for each document, we need to have the *ds*-features of a document and the center of the positive class. We discuss *ds*-features first, which are extracted from each document itself.

Since our task is document classification, we use the popular unigram, bigram and trigram with *tf-idf* weighting as the *ds*-features for a document. These three types of *ds*-features also give us three different document representations.

CBS Features. *Ds*-vectors are transformed into *cbs*-vectors by applying a set of similarity measures on each document space vector and the corresponding center vector. In this work, we employed five similarity measures in (Cha, 2007) to gauge the similarity of two vectors. Based on these measures, we produce 15 CBS features using the unigram, bigram, and trigrams representations of each document. The similarity measures we used are listed below, where P and Q are two vectors and d represents the dimension of P and Q.

$$s_{cos} = \frac{\sum_{i=1}^{d} P_i Q_i}{\sqrt{\sum_{i=1}^{d} P_i^2}\sqrt{\sum_{i=1}^{d} Q_i^2}} \tag{2.5}$$

$$s_{gow} = 1 - \frac{1}{d}\sum_{i=1}^{d} \mid \frac{P_i}{\sqrt{\sum_{i=1}^{d} P_i^2}} - \frac{Q_i}{\sqrt{\sum_{i=1}^{d} Q_i^2}} \mid \tag{2.6}$$

$$s_{lor} = 1 - \sum_{i=1}^{d} \ln(1+ \mid P_i - Q_i \mid) \tag{2.7}$$

$$s_{dice} = \frac{2\sum_{i=1}^{d} P_i Q_i}{\sum_{i=1}^{d} P_i^2 + \sum_{i=1}^{d} Q_i^2} \tag{2.8}$$

$$s_{jac} = \frac{\sum_{i=1}^{d} P_i Q_i}{\sum_{i=1}^{d} P_i^2 + \sum_{i=1}^{d} Q_i^2 - \sum_{i=1}^{d} P_i Q_i} \tag{2.9}$$

2.3.4 CBS Learning for Negative Covariate Shift

We now try to explain why CBS learning (CBS-L) can deal with the covariate shift problem, and thus can perform better than document space learning. The reason is that due to the use of similarity features, CBS-L is essentially trying to generate a boundary for the positive training data because similarity is not directional and thus covers all directions in a spherical shape in the space. In classification, the negative data from anywhere or direction outside the spherical shape can be detected. The covariate shift problem will not affect the classification much. Many types of documents that are not represented in the negative training data will still be detected due to their low similarity. For example, in Figure 1, we want to build a SVM classifier to separate positive data represented as black squares and negative data represented as empty

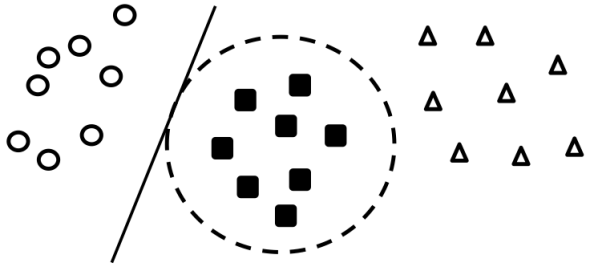


Figure 1. CBS Learning vs. DS Learning

circles. The constructed CBS-L classifier would look like a circle (in dashed line) in the original document space covering the positive data. The size of this (boundary) circle depends on the separation margin between the two classes. Although data points represented by empty triangles are not represented in the negative training data (which has only empty circles) in building the classifier, our classifier is able to identify them as *not positive* at the test time because they are outside the boundary circle.

If we had used the document space (DS) features to build a SVM classifier, the classifier would be a line between the positive data (black squares) and the negative data (empty circles). This line unfortunately will not be able to identify data points represented as empty triangles as not positive because the triangles actually lie on the positive side and would be classified as positive, which is clearly wrong.

2.4 Experiments

In this section, we evaluate the proposed learning in the center-based similarity space and compare it with baselines.

2.4.1 Dataset

As stated at the beginning of the chapter, this work was motivated by the real-life problem of identifying the right social media posts or documents for specific applications. For an effective evaluation, we need a large number of classes in the data to reflect the topic richness and diversity of the social media. The whole data also has to be labeled for evaluation. Using online reviews of a large number of products is a natural choice because there are many types of products and services and there is no need to do manual labeling, which is very labor intensive, time consuming, and error prone. We used the Amazon review dataset in (Chen and Liu, 2014a), which contains 100 types of products or domains. We randomly select 50 of them, which we also call 50 topics. Each topic has 1000 reviews. For each class in the dataset, we randomly sampled 700 of documents for training, and the rest 300 for testing. Note that although we use this product review collection, we do not perform sentiment classification. Instead, we still perform the traditional topic based classification. That is, given a review, the system decides what type of product the review is about. In our experiments, we use every topic as the positive class. This gives us 50 classification results.

2.4.2 Baselines

Now we introduce three baseline methods that will be used in our evaluation.

Document space One-class SVM (*ds*-osvm): As we discussed earlier, due to the co-variate shift problem in the negative training data, one obvious solution is to drop the negative training data completely to build a one-class classifier. As One-class SVM is the most representative and one of the most widely used one-class classification algorithms, we apply it to

the documents in the document space as one of the baselines. One-class SVM assumes that the origin is the only member of the second class. The data is first mapped into a transformed feature space via a kernel and then a standard two-class SVM is employed to construct a hyperplane that separates the data and the original with maximum margin. As we mentioned earlier, the support vector data description (SVDD) (Tax and Duin, 1999a) is another formulation for one-class classification, and it gives comparable results to One-class SVM when Gaussian kernel is used. Thus in the experiments, we only use One-class SVM with Gaussian kernel. Its implementation is in the LIBSVM library (version 3.20) (Chang and Lin, 2011).

Center-based similarity space One-class SVM (*cbs*-osvm): Instead of applying One-class SVM to documents in the original document space, this baseline applies it to documents in the CBS space. Documents are first transformed to CBS vectors before One-class SVM is applied. Note that this baseline uses the negative data in the document transformation step, although One-class SVM does not.

Document space SVM: This baseline is the standard binary classification using SVM, where SVM is applied in the original document space. Although in this case, there is covariate shift problem, we hope to evaluate how serious the problem might be, and see how the proposed CBS-L technique is able to deal with the problem. We also use the SVM implementation in the LIBSVM package (Chang and Lin, 2011).

2.4.3 Kernels and Parameters

As Khan and Madden (2013) pointed out that One-class SVM performs the best when Gaussian kernel is used, we use Gaussian kernel as well. Manevitz and Yousef (2001) applied

One-class SVM to text classification, and the authors reported that One-class SVM works the best with binary feature weighting scheme compared to *tf* or *tf-idf* weighting schemes. Also, they reported that a small number of features (10) with highest document frequency performed the best with Gaussian kernel. We also use binary representation, but found that 10 features are already too many in our case. In fact, 5 features give the best results. Using a small number of features is intuitive because to find the boundary of a very high dimensional space is very difficult. We also tried more features but they were poorer.

For SVM classification in the document space, we use the linear kernel as it has been shown by many researchers that the linear kernel performs the best (Joachims, 1998; Colas and Brazdil, 2006). We experimented with RBF kernels extensively, but they did not perform well with the traditional document representation. The term weighting scheme is *tf-idf* (Colas and Brazdil, 2006) with no feature selection.

For our proposed method CBS-L, we use *tf-idf* term weighting scheme of unigram, bigram and trigram to represent a document in three ways in the document space. As mentioned earlier, five document similarity functions are used to transform document space vectors to CBS space vectors. And in order to filter out less useful features for the center vector of the positive class, we performed feature selection in the document space using the classic information gain method (Yang and Pedersen, 1997) to empirically choose the most effective 100 features for the positive class.

2.4.4 Results

We now present the experiment results. As mentioned above, we treat each topic as the positive class. This gives 50 test results. To test the effect of covariate shift, we also vary the number of topics in the negative class. We used 10, 20, 30, and 40 topics in the training negative class. The test set always has 49 topics of negative data.

For each setting, we give three sets of results for the positive class, which is the target topic data that we are interested in obtaining through classification. Each set of results includes the standard measures of precision, recall, and F1-score for the positive class. The three sets are:

1. **In-training**: In this case, the test negative data contains only data from those topics used in training. This is the classical supervised learning setting where the training and test data are randomly drawn from the same distribution.
2. **Not-in-training**: In this case, the test negative set contains only data from the other topics not used in training. The classical setting of supervised learning does not deal with this problem. This represents covariate shift.
3. **Combined**: In this case, the test data contains both in-training and not-in-training negative topics. Due to the use of not-in-training test data, this is also not the classical setting.

Due to a large number of experiment results, we only summarize them in Table I by focusing on the F1 scores of all settings. Notice that for *ds*-osvm, it does not make sense to have in-training and not-in-training results because it does not use any training negative data. Thus, there is only one set of results for "Combined", which is duplicated in the table for easy

	In-training			***Out-of-training***			***Combined***		
	prec.	recall	F1	prec.	recall	F1	prec.	recall	F1
	10 topics are used in the training negative class								
ds-osvm	N/A			N/A			0.154	0.498	0.205
cbs-osvm	0.664	0.453	0.514	0.357	0.442	0.339	0.343	0.452	0.330
SVM	0.678	0.811	0.736	0.176	0.803	0.282	0.160	0.819	0.262
CBS-L	0.796	0.766	0.776	0.384	0.768	0.491	0.368	0.754	0.481
	20 topics are used in the training negative class								
ds-osvm	N/A			N/A			0.154	0.498	0.205
cbs-osvm	0.561	0.477	0.466	0.430	0.445	0.390	0.364	0.457	0.344
SVM	0.566	0.753	0.643	0.304	0.753	0.422	0.254	0.758	0.371
CBS-L	0.761	0.700	0.723	0.557	0.702	0.608	0.485	0.693	0.558
	30 topics are used in the training negative class								
ds-osvm	N/A			N/A			0.154	0.498	0.205
cbs-osvm	0.451	0.491	0.393	0.488	0.524	0.407	0.378	0.487	0.355
SVM	0.508	0.721	0.591	0.450	0.722	0.547	0.323	0.726	0.439
CBS-L	0.723	0.650	0.678	0.721	0.644	0.667	0.569	0.649	0.598
	40 topics are used in the training negative class								
ds-osvm	N/A			N/A			0.154	0.498	0.205
cbs-osvm	0.423	0.482	0.379	0.590	0.511	0.444	0.372	0.486	0.347
SVM	0.456	0.689	0.544	0.641	0.685	0.658	0.374	0.695	0.481
CBS-L	0.697	0.613	0.644	0.848	0.616	0.699	0.639	0.613	0.619

TABLE I

Result summary for 50 topics.

comparison. However, note that *cbs*-osvm uses negative data for training in order to compute the center for the positive class.

From Table I , we can make the following observations.

1. The proposed CBS-L method performs markedly better than all baselines. For the results of in-training, not-in-training, and combined, CBS-L is consistently better in all cases than

Topic	**ds-osvm**	**cbs-osvm**	**SVM**	**CBS-L**
Amplifier	0.125	0.360	0.406	0.597
Automotive	0.041	0.031	0.240	0.383
Battery	0.266	0.425	0.433	0.656
Beauty	0.079	0.401	0.470	0.618
Cable	0.131	0.028	0.231	0.500
Camera	0.376	0.361	0.433	0.523
CD Player	0.154	0.274	0.344	0.585
Clothing	0.046	0.234	0.292	0.486
Computer	0.117	0.225	0.328	0.455
Fan	0.408	0.581	0.581	0.724
Flashlight	0.273	0.487	0.528	0.744
Graphics Card	0.419	0.473	0.552	0.631
Headphone	0.298	0.338	0.432	0.533
Home Improvement	0.039	0.032	0.178	0.233
Jewelry	0.362	0.579	0.632	0.800
Kitchen	0.042	0.118	0.197	0.261
Lamp	0.091	0.249	0.374	0.487
Magazine Subscriptions	0.406	0.597	0.796	0.858
Mattress	0.435	0.562	0.603	0.702
Memory Card	0.134	0.256	0.367	0.508
Microphone	0.103	0.223	0.25	0.417
Microwave	0.378	0.577	0.637	0.735
Monitor	0.136	0.345	0.312	0.513
Movies TV	0.146	0.507	0.641	0.682
Musical Instruments	0.073	0.241	0.446	0.575
Network Adapter	0.164	0.483	0.481	0.596
Office Products	0.040	0.193	0.327	0.346
Patio Lawn Garden	0.043	0.226	0.295	0.483
Pillow	0.491	0.640	0.781	0.888
Printer	0.549	0.557	0.624	0.859
Projector	0.230	0.459	0.482	0.805
Shoes	0.224	0.524	0.585	0.793
Speaker	0.241	0.251	0.253	0.410
Subwoofer	0.147	0.268	0.346	0.401
Tablet	0.069	0.234	0.142	0.424
Telephone	0.099	0.034	0.144	0.167
Toys	0.088	0.029	0.331	0.449
Video Games	0.424	0.387	0.508	0.705
Wall Clock	0.401	0.582	0.607	0.777
Webcam	0.155	0.304	0.372	0.645

TABLE II

Complete results for 40 topics under the *combined* setting.

all baselines. Even for in-training, CBS-L performs better than SVM. This clearly shows the superiority of the proposed CBS-L method.

2. *Ds*-osvm performs poorly. *Cbs*-osvm is much better because it uses the negative data in feature selection and center computation.
3. SVM in the document space performed poorly (Combined) when only a small number of negative topics are used in training. It gets better than both One-class SVM baselines when more negative topics are used in training (see the reason in the next point).
4. Finally, we can also see that with the number of training negative topics increases, the results of the combined case of both SVM and CBS-L improve. This is expected because with the increased number of negative topics for training, the number of not-in-training negative topics for testing decreases and the covariate shift problem gets smaller. We can also see that *cbs*-osvm, SVM and CBS-L's F1-scores for not-in-training improve with the increased training negative topics due to the same reason. However, their F1-scores drop for in-training because with more negative topics, the data becomes more skewed, which hurts in-training classification.

To give a flavor of the detailed results for each topic (product), we give the full results for one setting with 40 randomly selected topics as the training negative data (Table II). The results in the table are F1-scores of the combined case.

CHAPTER 3

OPEN WORLD TEXT CLASSIFICATION

(This chapter was previously published as "Breaking the Closed World Assumption in Text Classification", in *The 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL'16)* (Fei and Liu, 2016).)

3.1 Introduction

With the rapid growth of online information, text classifiers have become one of the most important tools for people to track and organize information. And the emergence of social media platforms has brought increasing diversity and dynamics to the Web. Many social science researchers rely on the collected online user generated content to carry out research on different social phenomenon. In this case, multi-class text classifiers are widely used to gather information of several topics of interest. However, most existing research on multi-class text classification makes the *closed world* assumption, meaning that all the test classes have been seen in training. However, in a more realistic scenario where people use a multi-class classifier to collect information of several topics from a data source that covers a much broader range of topics, it is normal to break the closed world assumption and to see the arrival of documents from unknown classes that have never been seen in training. In this case, a multiclass classifier should not always assign a document to one of the known classes. Instead, it should identify

unknown classes of documents and label them as unknown or reject. This is called *open* (*world*) classification.

More precisely, in the traditional multi-class classification setting, the learner assumes a fixed set of classes $Y = \{C_1, C_2, \ldots, C_m\}$, and the task is to construct a m-class classifier using the training data. The resulting classifier is tested/applied on the data from only the m classes. The assumption is often violated in reality and documents not belonging to any of the m classes are forced to fall into one of the m classes. While in *open* classification, we allow the classifier to predict labels/classes from the set of C_1, C_2, ..., C_m, C_0 classes, where class C_0 represents the unknown which covers documents of all unknown or unseen classes or topics. In other words, every test instance may be predicted to belong to either one of the known classes $y_i \in Y$, or C_0 (unknown).

It is thus not sufficient for a classifier to just return the most likely class label among the m known classes. An option to reject must be provided. An obvious approach to predicting the class label $y \in Y \cup \{C_0\}$ for an n-dimensional data point $\boldsymbol{x} \in R^n$ is to incorporate a posterior probability estimator $p(y|\boldsymbol{x})$ and a decision threshold into an existing multi-class learning algorithm (Kwok, 1999; Fumera and Roli, 2002; Huang et al., 2006b; Bravo et al., 2008). There are many reasons this technique would not achieve good results in open classification. As we will discuss in the following sections, one of the most important reasons is that the underlying classifier is not robust or is not informed enough to reject unseen classes of documents due to its significant open space risk.

Traditional multi-class learners optimize only on the known classes under the closed world assumption, while a potential learner for open classification has to optimize for both the known classes and the unknown classes. Some recent research in the field of computer vision studied the problem, which they call *open set recognition* (Scheirer et al., 2013; Scheirer et al., 2014; Jain et al., 2014) for facial recognition. Classic learners define and optimize over empirical risk, which is measured on the training data. For open classification, it is crucial to measure the risk of classifying the unseen as positive by preventing overgeneralization or overspecialization. In order to tackle this problem, Scheirer et al. (2013) introduced the concept of *open space risk* and formulated an extension of the existing one-class and binary SVMs to address the open classification problem. However, as we will see in Section 3.3, their proposed method is weak as the positively labeled open space is still an infinite area.

In this chapter, we propose a solution to reduce the open space risk while also balancing the empirical risk for open classification. Intuitively, given a positive class of documents, our open space for the positive class is considered as the space that is sufficiently far from the center of the positive documents. In the multi-class classification setting, each of the m target classes is surrounded by a ball covering the positively labeled (the target class) area, while any document falling outside of all the m balls is considered belonging to the unknown class.

In Chapter 2 we introduced a learning technique called center-based similarity (CBS) space learning to deal with the problem of covariate shift in binary classification. We found that it is also suitable for open world classification. Unlike SVM learning in the document space that bounds the positive class only by an infinite half-space formed with the decision hyperplane,

which has a huge open space risk, CBS learning finds a closed boundary for the positive class covering only a finite area, which is a spherical area in the original document space and thus reduce the open space risk significantly. Our final multi-class classifier is called *cbsSVM*, as it uses SVM as the underlying learner.

In summary, this work makes the following contributions:

1. To the best of our knowledge, this work is the first attempt to study multi-class open classification in text from the open space risk management perspective.
2. It gives an open space risk formulation for the problem and applies CBS learning technique as an initial solution based on our formulation.
3. Extensive experiments show that the proposed *cbsSVM* for multi-class open classification produces superior classifiers to existing state-of-the art methods.

3.2 Related Work

Comparing to research on multi-class classification with the closed world assumption, there is relatively less work on open world classification. This work is related to existing research on one-class classification, which we have discussed in detail in Section 2.2. In this section we will first focus on reviewing two areas of work in open world classification. Then, we discuss related work in the semi-supervised learning setting.

3.2.1 SVM Decision Score Calibration

Early work on SVM-based open world classification relied on a calibration process to transform raw SVM scores to probabilities. As the decision score produced by SVM is not a probability distribution, several techniques have been proposed to convert a raw decision score

to calibrated probabilistic output (Bravo et al., 2008; Duan and Keerthi, 2005; Huang et al., 2006b; Platt, 1999; Zadrozny and Elkan, 2002). Usually a parametric distribution is assumed (Gaussian models are the most popularly used) for the underlying distribution, and raw scores are mapped based on the learned model. For the task of score calibration, the most widely used technique is proposed in (Platt, 1999), which has been extended and applied in many other learning systems. It is based on the observation that data is in general "roughly sigmoidal" by analyzing specific empirical data instances. In particular, a sigmoid function is first used to fit the raw SVM scores during the training phase, and then the raw SVM scores for test instances are transformed into calibrated probabilities based on the model. Provided with a threshold, a test instance can be rejected if its highest probability of belonging to any class is lower than the threshold. In this work, we also apply Platt's (1999) method for decision score calibration due to its simplicity. Its implementation is included in the LIBSVM package (Chang and Lin, 2011).

Recent work in computer vision (Scheirer et al., 2011) suggested that the assumptions made in the statistical extreme value theory (EVT) (Kotz and Nadarajah, 2000) is coherent to recognition problems, and compared to existing techniques, the EVT allows us to calculate probabilities without the overall distribution of the data. The extreme scores given by any recognition algorithm can be modeled using the EVT. More specifically, a reverse Weibull distribution can be used to fit the data if it is bounded from above, and a Weibull distribution is suitable if the data is bounded from below (Scheirer et al., 2014). Thus, new SVM score calibration techniques based on the EVT were proposed (Jain et al., 2014; Scheirer et al.,

2014). In particular, Scheirer et al. (2014) proposed W-SVM, which is a combination of a One-class SVM and a binary SVM for each target class. The One-class SVM is used as a conditioner to reject test instances that have small likelihood of belonging to some target class. And the EVT is applied to the calibration process for both the One-class SVM and the binary SVM. A similar idea was also used in (Jain et al., 2014), where the EVT is used to estimate the unnormalized posterior probability of inclusion for each class by fitting a Weibull distribution over the positive class scores from a 1-vs-rest multi-class RBF SVM classifier. For both of the above methods, decision thresholds need to be chosen based on the prior knowledge of the ratio of unseen classes in testing, which is a weakness of the methods.

3.2.2 Open Space Risk Management

Recently, researchers made several attempts to solve open world classification (which they call *open set recognition*) for visual learning from new angles (Scheirer et al., 2013; Scheirer et al., 2014). In particular, Scheirer et al. (2013) introduced the concept of *open space risk*, and defined it as a relative measure. The proposed technique, 1-vs-set machine, reduces the open space risk of traditional SVM by replacing the positively labeled half-space with a positive region bounded by two parallel hyperplanes. While the positively labeled region for a target class is reduced compared to the half-space in traditional linear SVM, the open space risk is still infinite. Scheirer et al. (2014) introduced the Compact Abating Probability (CAP) model, which explains how thresholding the probabilistic output of RBF One-class SVM manages the open space risk. However, due to the fact that One-class SVM produces relatively weak results, the overall performance of the technique is also negatively affected.

3.2.3 Multi-class Semi-supervised Learning

Apart from supervised learning, researchers also tried to deal with unseen/unknown classes in the semi-supervised learning setting. Dalvi et al. (2013) proposed Exploratory Learning in the multi-class semi-supervised learning (SSL) setting. In their work, an "exploratory" version of expectation-maximization (EM) is proposed to extend traditional multi-class SSL methods. In the new setting, the algorithm is only given seed documents from a subset of the classes in the data, either because of the lack of complete knowledge of the data or special interest of the user. It automatically explores different numbers of new classes in the EM iterations. The underlying assumption is that when an instance x has close to uniform probability of belonging to the existing classes, a new class should be created. This is quite different from our work and objective. Firstly, it works in the semi-supervised setting and assumes that test data is available during training. Secondly, it only focuses on improving accuracy on the classes with seed examples.

3.3 Proposed Method

In this section, we propose our technique for the open classification problem. First we introduce our open space risk formulation for the case of binary classification. Then we show how our proposed CBS learning technique (Chapter 2) can be used as an initial solution according to the proposed open space risk formulation. Lastly, we combine all the binary classifiers in the 1-vs-rest fashion to produce our final multi-class open world classifier.

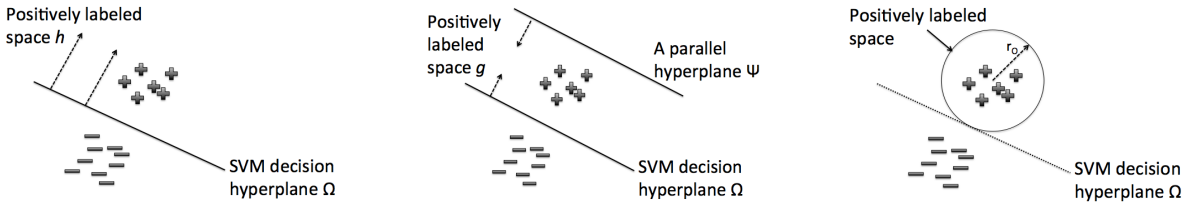
3.3.1 Open Space Risk Formulation

Consider the risk formulation by Scheirer et al. (2013), where apart from the empirical risk, there is risk in labeling the open space (space away from positive training examples) as "positive" for any known class. Due to lack of information on a classification function on the open space, open space risk is approximated by a relative Lebesgue measure (Shackel, 2007). Let S_O be a large ball of radius r_O that contains both the positively labeled open space O and all of the positive training examples; and let f be a classification function where $f_y(x) = 1$ for classifying y as positive and $f_y(x) = 0$ otherwise. The probabilistic open space risk $R_O(f)$ of function f for a class y is defined as the ratio of the positive open space (the positively labeled space that is far from most positive examples) to the overall space that is positively labeled (which includes the space where most positive examples reside).

$$R_O(f) = \frac{\int_O f_y(x)dx}{\int_{S_O} f_y(x)dx} \tag{3.1}$$

Equation 3.1 indicates that the more we label open space as positive, the greater open space risk is. However, it does not suggest how to specify the positively labeled open space O.

In this work, we formulate O as the positively labeled area that is sufficiently far from the center of the positive training examples. Let $B_{r_y}(cen_y)$ be a closed ball of radius r_y centered around the center of positive class y (cen_y), which ideally contains all positive examples of class y; S_O be a larger ball $B_{r_O}(cen_y)$ of radius r_O with the same center cen_y. Let classification function $f_y(x) = 1$ when $x \in B_{r_O}(cen_y)$, and $f_y(x) = 0$ otherwise. Also let h be the positive



(a) SVM (b) 1-vs-set Machine (c) Ours

Figure 2. Illustration of the positively labeled open space.

half space defined by a binary SVM decision hyperplane Ω obtained using positive and negative training examples, and let the size of ball B_{r_O} be bounded by Ω, $B_{r_O} \cap h = B_{r_O}$. We define open space as

$$O = S_O - B_{r_y}(cen_y) \tag{3.2}$$

where the radius of S_O, r_O, needs to be determined from the training data for each known positive class.

This open space formulation greatly reduces the open space risk compared to traditional SVM and the 1-vs-set Machine in (Scheirer et al., 2013), which is illustrated in Figure 2. In particular, Figure 2(a) shows the positively labeled space for traditional SVM, whose classification function $f_y^{\text{SVM}}(x) = 1$ when $x \in h$. Thus, its positive open space is approximately $h - B_{r_y}(cen_y)$, which is only bounded by the SVM decision hyperplane Ω. Figure 2(b) shows the positively labeled space for 1-vs-set Machine in (Scheirer et al., 2013), whose classification function $f_y^{\text{1-vs-set}}(x) = 1$ when $x \in g$, where g is a slab area with thickness δ bounded by two parallel hyperplanes Ω and Ψ ($\Omega \parallel \Psi$) in h. And its positive open space is approximately

$g - B_{r_y}(cen_y)$. Figure 2(c) illustrates our proposed open space formulation, in which the positively labeled space is a finite ball bounded by the SVM decision hyperplane Ω. Given the open space formulations of traditional SVM and the 1-vs-set Machine, we can see that both of these methods label an unlimited area as positively labeled space, while our formulation reduces it to a bounded spherical area.

Given our open space formulation, the question is how we estimate the radius r_O of the spherical area S_O for each positive class. We show that our proposed center-based similarity space learning (CBS learning) in Chapter 2 follows the proposed open space risk formulation and is able to give an estimation, although the technique was original proposed to deal with the negative covariate shift problem in binary text classification.

3.3.2 CBS Learning for r_O Estimation

Due to learning in the similarity space with similarities as features, CBS learning generates a boundary based on similarities to separate the positive and negative training data in the similarity space, which is essentially a ball encompassing the positive training data in the original document space. In other words, instead of explicitly minimizing the positively labeled open space risk, CBS learning approximates the radius r_O by learning a score based on similarities in the similarity space, which is equivalent to a limited spherical area in the original document space. The generated model thus not only limits the positively labeled open space on the positive side of Ω (SVM decision hyperplane), but also balances the empirical risk from the positive and negative training examples. In fact, r_O is approximately the distance from the center of positive class to Ω measured in similarities (Figure 2(c)). The positively labeled/classified

region produced by CBS learning is the circle in the original document space, while SVM learning produces a half space bounded by its decision line. Note that as multiple similarity features are used, the spherical area is formed by an integrated similarity produced by SVM, which combines all similarity features.

In order for the method to work well for our multi-class classification, ideally two assumptions should be made about the data. First, the target classes of documents are generated by a mixture model, where each mixture component is responsible for one class of documents. Secondly, after feature normalization each target class of documents is generated by a Gaussian distribution, where the Gaussian mean resides at the center of the class, and its $n \times n$ covariance matrix has equal eigenvalues so that the positive class can have a spherical shape boundary or a ball. This assumption may be too strict and may be violated, but empirical results show it works well in reality. Note that we do not make any assumptions about data from non-target classes.

3.3.3 Final Open World Classifier

The preceding discussion is based on binary open classification. We follow the standard technique of combining a set of 1-vs-rest binary classifiers to perform multi-class classification with a rejection option for the unknown. The SVM scores for each classifier are first converted to probabilities based on a variation of Platt's (1999) algorithm, which is supported in LIBSVM (Chang and Lin, 2011). Let $P(y|\mathbf{x})$ be a probably estimate, where $y \in Y$ is a class label and $\mathbf{x}$ is a feature vector, and let λ be the decision threshold (usually 0.5). Let Y be the set of known

classes, C_0 be the unknown class, and y^* is the final predicted class for $\mathbf{x}$. The final classifier, called *cbsSVM*, uses this following for classification:

$$y^* = \begin{cases} \text{argmax}_{y \in Y} P(y|\mathbf{x}) & \text{if } P(y|\boldsymbol{x}) \geq \lambda \\ C_0 & \text{otherwise} \end{cases} \tag{3.3}$$

3.4 Experiments

In this section, we show the results of the proposed method *cbsSVM* and compare it with the state-of-the-art baselines across two datasets. We first introduce the baseline methods and the experimental datasets.

3.4.1 Baselines

This subsection lists a set of start-of-the-art baselines for open world classification. In Chapter 4, they will be used again for comparison.

1-vs-rest multi-class SVM (*1-vs-rest-SVM*). This is the standard 1-vs-rest multi-class SVM with Platt's Probability Estimation (Platt, 1999), and it is implemented based on LIBSVM [1] (version 3.20) (Chang and Lin, 2011). It works in the same way as the proposed *cbsSVM* in Section 3.3.3 except that it uses the document space classification. Linear kernel is used as it is shown by many researchers that linear SVM performs the best for text classification (Joachims, 1998; Colas and Brazdil, 2006).

---

[1] http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/

1-vs-set Machine (*1-vs-set-linear*). This is the baseline method proposed in (Scheirer et al., 2013), and we use all the default parameter settings in the original paper. That is, the near and far plane pressures are set at $p_A = 1.6$ and $p_\Omega = 4$ respectively, which are the two parameters to allow users to specify the importance of the open space near the two decision hyperplanes. Regularization constant is set at $\lambda_r = 1$, which gives equal weights on the empirical risk and the open space risk. And no explicit hard constraints are used on the training error ($\alpha = 0$, $\beta = 1$).

W-SVM (*wsvm-linear* and *wsvm-rbf*). These are the two baselines proposed in (Scheirer et al., 2014), which combine a RBF One-class SVM with a binary SVM with linear and RBF kernel respectively. For thresholding the output, two parameters δ_τ and δ_R are required. We set $\delta_\tau = 0.001$, which is used to control what data is considered to be relevant by the One-class SVM classifier. δ_R is the final decision threshold not only for W-SVM, but also for the next two baselines (P_I-SVM, P_I-OSVM). Two ways of setting δ_R were suggested by the authors. We set it as the prior probability of the number of unseen classes during evaluation (testing). An alternative way is to set it based on an openness score computed using the number of training and testing classes. We tried both methods and found the former gave better results.

P_I-SVM (*P_I-svm-linear* and *P_I-svm-rbf*). This baseline is from (Jain et al., 2014), which estimates the probability of inclusion by applying the EVT on the positive class scores from the output of binary SVMs. Both linear and RBF kernels are tested. The threshold δ

serves the same purpose as the δ_R in W-SVM, and is set as the prior probability of the number of unseen classes in test data.

P_I-OSVM (*P_I-osvm-linear* and *P_I-osvm-rbf*). This baseline is similar to P_I-SVM. The only difference is that P_I-OSVM (Jain et al., 2014) uses a multi-class One-class SVM before fitting an Extreme Value Theory distribution to estimate the probability of inclusion. Again, two kernel functions are tested and the prior probability of the number of unseen classes is used to set δ. As P_I-OSVM is a variant of the traditional One-class SVM, we do not use One-class SVM as a baseline.

Exploratory Seeded K-Means (*Exploratory-EM*). In (Dalvi et al., 2013), three well-known multi-class semi-supervised learning methods were extended under the exploratory learning framework. We compare with exploratory version of Seeded K-Means due to its superior performance on 20newsgroup dataset reported by the authors. We also applied the criteria that work the best in the original paper for creating new classes and for model selection, i.e., the MinMax criterion and the AICc criterion. Note that ExploratoryEM works in the semi-supervised setting and uses both the training and test data as labeled and unlabeled data in training. As more than one new class can be introduced during training, for comparison we lump together all instances assigned to new classes as being rejected (unknown). In the experiments, we set the max number of iterations to be 50. Little changes in results are observed after 50 iterations.

All documents use *tf-idf* term weighting scheme with no feature selection. Source code for different baselines (1-vs-set Machine [1], W-SVM and P_I-SVM [2], and Exploratory learning [3]) was provided by the authors of their original papers.

3.4.2 Datasets

We perform experimental evaluation using two public available datasets: the 20-newsgroup in (Joachims, 1996) and the Amazon reviews (Chen and Liu, 2014a). For the Amazon dataset, we use the same 50 domains/products of reviews as in Chapter 2 to perform the traditional topic based classification. The 20-newsgroup data contains 20 non-overlapping classes with a total of 18828 documents. For each class in both datasets, we randomly sampled 700 of documents for training, and the rest 300 for testing.

3.4.3 Experiment Settings

Following (Jain et al., 2014; Dalvi et al., 2013), we conduct open world cross-validation style analysis, which means some classes are held out during training and are mixed back during testing. We also vary the number of training and test classes. In particular, we set the number of test classes at 10, 20, 30, 40, 50 for Amazon dataset, and at 10, 20 for 20-newsgroup dataset. For each chosen number of test classes, we randomly use 25%, 50%, 75% and 100% of the test classes in the training step. Varying the ratio of the number of training classes to

---

[1] https://github.com/Vastlab/liblinear.git

[2] https://github.com/ljain2/libsvm-openset

[3] http://www.cs.cmu.edu/ bbd/ExploreEM_package.zip

test classes is to test the robustness of different systems in handling different "openness" of the problem. Since there are many different combinations on the subset of training classes we can select, for each setting we repeat the experiments for 5 times and report the average results. The only exception is when 100% of the test classes are chosen in training, we only conduct the experiment once.

When 100% of test classes are used in training, the problem reduces to the closed world classification. As most of our baselines such as W-SVM, P_I-OSVM and P_I-SVM use prior knowledge to set decision threshold, i.e., $\delta_R = 0$ for W-SVM and $\delta = 0$ for P_I-OSVM and P_I-SVM, in the closed world setting, for fair comparison we also set the threshold to $\lambda = 0$ for both 1-vs-rest-SVM and our proposed *cbsSVM* in the closed world classification setting. By doing this, we always assign a known class label to a test instance. For Exploratory Seeded K-Means, we use an option supported in the exploratory learning package that does not allow any new classes to be introduced in learning.

For each setting, we first compute precision, recall and F1 score for each class and then macro-average the results across all classes. Final results are given by averaging the results across 5 random training and test classes combinations, except for the case of using 100% of the test classes in training. Since there are many results, we only report F1 scores.

For all the methods that use the RBF kernel, the parameters are empirically chosen through cross validation on the training data. In particular, we set $C = 5$ and $\gamma = 0.2$ for Amazon, and $C = 10$ and $\gamma = 0.5$ for 20newsgroup. We do not perform over-sampling or under-sampling all methods for two reasons. First, as we have discussed earlier, our experiments have multiple

settings that use different number of classes, it is hard to select the optimal sampling number every time, and it is also not the focus of our work. Secondly, since this strategy applies to all the methods, we do not bias against any one.

3.4.4 Results and Discussion

In this subsection, we first show the main results on two datasets, and then conduct further analysis on the performance of our technique and discuss its weakness. Open classification results on the Amazon dataset is given in Table III, and those on 20-newsgroup are given in Table IV. As we can see across two datasets, in most situations (23 of 28 settings) our proposed cbsSVM method performs the best. Even when 100% of the test classes are used for training (the traditional closed world classification), cbsSVM still performs the best in almost all settings (6 out of 7) except for 20-newsgroup with 20 classes. In this case, it lost to ExploratoryEM by 1.12%. In fact, it is unfair to compare cbsSVM with ExploratoryEM because ExploratoryEM uses the test data in training.

We also analyze the cases where our technique does not perform well. By looking at the sub-tables where 10 domains are tested in Table III and Table IV, we see that our method loses to 1-vs-set-linear, wsvm-linear and P_I-svm-linear on both datasets when training on 2 classes (25%) and testing on 10 classes, though in other cases training on 25% known classes can still yield good results. By inspecting the results, we found that in both settings our technique achieves very high recall but low precision on the known classes, while achieving high precision but low recall on the unknown classes. After careful investigation, we found this is caused by

	25%	50%	75%	100%	25%	50%	75%	100%
	10 Domains				20 Domains			
cbsSVM	0.450	**0.715**	**0.775**	**0.873**	0.566	**0.695**	**0.695**	**0.760**
1-vs-rest-SVM	0.219	0.658	0.715	0.817	0.466	0.610	0.616	0.688
ExploratoryEM	0.386	0.647	0.704	0.854	**0.571**	0.561	0.573	0.691
1-vs-set-linear	0.592	0.698	0.700	0.697	0.506	0.560	0.589	0.620
wsvm-linear	0.603	0.694	0.698	0.702	0.553	0.618	0.625	0.641
wsvm-rbf	0.246	0.587	0.701	0.792	0.397	0.502	0.574	0.701
P_I-osvm-linear	0.207	0.590	0.662	0.731	0.453	0.531	0.589	0.629
P_I-osvm-rbf	0.061	0.142	0.137	0.148	0.143	0.079	0.058	0.050
P_I-svm-linear	**0.600**	0.695	0.701	0.705	0.547	0.620	0.628	0.644
P_I-svm-rbf	0.245	0.590	0.718	0.774	0.396	0.546	0.675	0.714
	30 Domains				40 Domains			
cbsSVM	**0.565**	**0.645**	**0.630**	**0.686**	**0.541**	**0.633**	**0.619**	**0.650**
1-vs-rest-SVM	0.463	0.568	0.545	0.627	0.463	0.543	0.515	0.584
ExploratoryEM	0.500	0.511	0.569	0.659	0.467	0.496	0.562	0.628
1-vs-set-linear	0.462	0.511	0.542	0.585	0.429	0.489	0.526	0.558
wsvm-linear	0.521	0.574	0.578	0.598	0.499	0.554	0.560	0.565
wsvm-rbf	0.372	0.444	0.502	0.651	0.351	0.402	0.464	0.609
P_I-osvm-linear	0.428	0.510	0.553	0.605	0.413	0.483	0.533	0.571
P_I-osvm-rbf	0.108	0.047	0.043	0.047	0.078	0.043	0.047	0.049
P_I-svm-linear	0.520	0.575	0.581	0.602	0.497	0.554	0.563	0.568
P_I-svm-rbf	0.379	0.517	0.629	0.680	0.371	0.505	0.602	0.634

	50 Domains			
	25%	50%	75%	100%
cbsSVM	**0.557**	**0.615**	0.586	**0.634**
1-vs-rest-SVM	0.460	0.533	0.502	0.568
ExploratoryEM	0.348	0.467	0.534	0.618
1-vs-set-linear	0.420	0.483	0.514	0.551
wsvm-linear	0.488	0.545	0.549	0.559
wsvm-rbf	0.317	0.367	0.436	0.584
P_I-osvm-linear	0.403	0.489	0.535	0.578
P_I-osvm-rbf	0.066	0.039	0.047	0.050
P_I-svm-linear	0.487	0.546	0.551	0.562
P_I-svm-rbf	0.360	0.509	**0.632**	0.630

TABLE III

Open classification results on the Amazon dataset.

	25%	50%	75%	100%	25%	50%	75%	100%
	10 Domains				20 Domains			
cbsSVM	0.417	**0.769**	**0.796**	**0.855**	**0.593**	**0.701**	**0.720**	0.852
1-vs-rest-SVM	0.246	0.722	0.784	0.828	0.552	0.683	0.682	0.807
ExploratoryEM	0.648	0.706	0.733	0.852	0.555	0.633	0.713	**0.864**
1-vs-set-linear	**0.678**	0.671	0.659	0.567	0.497	0.557	0.550	0.577
wsvm-linear	0.666	0.666	0.665	0.679	0.563	0.597	0.602	0.677
wsvm-rbf	0.320	0.523	0.675	0.766	0.365	0.469	0.607	0.773
P_I-osvm-linear	0.300	0.571	0.668	0.770	0.438	0.534	0.640	0.757
P_I-osvm-rbf	0.059	0.074	0.032	0.026	0.143	0.029	0.022	0.009
P_I-svm-linear	0.666	0.667	0.667	0.680	0.563	0.599	0.603	0.678
P_I-svm-rbf	0.320	0.540	0.705	0.749	0.370	0.494	0.680	0.767

TABLE IV

Open classification results on the 20newsgroup dataset.

the relatively poor approximation of radius r_O when positive and negative training examples are far apart.

To verify the cause, we conducted more experiments on the 20-newsgroup data using the same setting (10 classes for test and 2 for training). The 10 classes are listed in Table V. We show the results for two sets of experiments. In each set of the experiments, we keep one known class unchanged in training and select different classes as the second class. We show how the results change on the unchanged class as well as the unknown (reject) classes. Table VI illustrates two examples. In particular, the top sub-table lists the precision, recall, and F1 score for *comp.windows.x* and for the unknown classes. Similarly, the bottom sub-table gives the results for *rec.motorcycles* and for the unknown classes. The first column in Table VI is the different second classes used in training. We can see that in both sets of experiments, the

Domain names	
rec.motorcycles	comp.graphics
comp.os.ms-windows.misc	alt.atheism
comp.sys.mac.hardware	comp.windows.x
misc.forsale	comp.sys.ibm.pc.hardware
rec.autos	rec.sport.baseball

TABLE V

10 chosen domains in 20newsgroup for analysis.

	comp.windows.x			Unknown (reject)		
	Prec.	Recall	F1	Prec.	Recall	F1
rec.motorcycles	0.260	0.963	0.410	0.972	0.168	0.287
comp.graphics	0.380	0.850	0.525	0.966	0.482	0.643
comp.sys.mac.hardware	0.286	0.972	0.442	0.977	0.356	0.522
comp.os.ms-windows.misc	0.418	0.877	**0.567**	0.976	0.513	0.672
misc.forsale	0.244	0.959	0.389	0.966	0.201	0.334
rec.autos	0.226	0.979	0.367	0.976	0.162	0.277
	rec.motorcycles			Unknown (reject)		
	Prec.	Recall	F1	Prec.	Recall	F1
comp.sys.mac.hardware	0.284	0.956	0.438	0.962	0.198	0.328
rec.autos	0.459	0.892	**0.606**	0.974	0.470	0.634
comp.windows.x	0.260	0.963	0.410	0.972	0.168	0.287
comp.graphics	0.289	0.953	0.444	0.964	0.177	0.299
comp.sys.ibm.pc.hardware	0.284	0.953	0.438	0.958	0.169	0.288
alt.atheism	0.194	0.973	0.324	0.980	0.333	0.498

TABLE VI

Results on example classes vs. unknown classes.

precision and F1 score on the unchanged known classes (*comp.windows.x* and *rec.motorcycles*) are better when a more similar class (closer in distance) is selected in training. In particular, *comp.windows.x* achieves the best result when *comp.os.ms-windows.misc* is the second known class, and *rec.motorcycles* achieves the best result when *rec.autos* is the second known class. This is because the radius r_O for each positively labeled space is determined based on the distance between the positive and negative training examples. As related classes are closer in distance, a tighter boundary with smaller r_O can be learned. However, our results show in the cases when only 2 known classes are available, a tight boundary is harder to achieve for either class for the proposed *cbsSVM*.

CHAPTER 4

CUMULATIVE LEARNING

(This chapter was previously published as "Learning Cumulatively to Become More Knowledgeable", in *The 22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'16)* (Fei et al., 2016).)

4.1 Introduction

Supervised learning has been very successful in research and in applications. However, existing supervised learning research has focused on developing effective individual statistical algorithms that learn accurate models or classifiers given a fixed dataset. Relatively little research has been done on how to build continuous learning systems that *learn cumulatively* and become more and more knowledgeable as the system sees more and more classes of data over time.

Let us use an example to motivate this research. The 2016 presidential election in the USA has been a hot topic on social media and many social science researchers rely on the collected online user discussions to carry out political science research. During the long campaign, every new proposal made by a presidential candidate is followed by a huge amount of discussions on social media. A multiclass classifier is thus needed to track and to organize the discussions from the general public. As the campaign goes on, the initially built model or classifier will inevitably and frequently encounter new topics (e.g. Donald Trump's plan for immigration

reform or Hillary Clinton's new proposal for tax increase) that have not been covered in previous learning. In this case, the classifier must first be able to detect these new topics when they occur rather than classifying them into some existing classes or topics. Second, after enough training examples of new/unseen topics are collected by human users, the existing classifier should incorporate the new classes or topics in the classification system in a manner that does not require rebuilding the whole classification system from scratch.

Based on this example, we can see two inter-related challenges in building a multi-class cumulative supervised learning system:

1. The ability to continuously detect new/unseen classes of data that have not been covered in training by the current classification system.
2. The ability to cumulatively add new classes to the system without having to re-train the entire system from scratch using all the past training data.

In this chapter, we aim to solve these two problems in the context of text classification. We call this *cumulative machine learning* or simply *cumulative learning*, which is a special form of *lifelong machine learning* (Carlson et al., 2010; Pentina and Lampert, 2014; Silver et al., 2013; Thrun and Mitchell, 1995). Formally, cumulative learning is stated as follows:

Problem Statement (Cumulative Learning): At any time point t, the system maintains a classifier, H^t, learned from a set of past training datasets $D^t = \{D_1, D_2, \ldots, D_t\}$ labeled with corresponding classes (labels) $Y^t = \{l_1, l_2, \ldots, l_t\}$, where every example in each dataset $D_i \in D^t$ is labeled with the same class $l_i \in Y^t$. H^t is able to classify each test instance to either one of the known classes in Y^t or the unknown class l_0, which represents all new or

unseen topics. In this case, H^t is said to perform *open world classification* (Chapter 3). Once enough training data D_{t+1} has been collected for an unknown topic/class l_{t+1} by the user, H^t is updated to cover the new class l_{t+1} to produce a new classifier H^{t+1} and H^{t+1} is also able to perform open world classification. We want to build H^{t+1} upon H^t with minimal effort and without re-training the entire system.

Let us now explain why cumulative learning is a form of lifelong machine learning (Chen et al., 2015), which is defined as follows:

Lifelong Machine Learning (LML) Definition: At any time point t, a learner has performed a sequence of t learning tasks, T_1, T_2, ..., T_t, and has accumulated the knowledge K learned in these past t tasks. At time point $t+1$, it is faced with a new learning task T_{t+1}. The learner is able to make use of the past knowledge K to help perform the new learning task T_{t+1}.

Cumulative learning is a form of lifelong machine learning because we can treat task T_{t+1} as the task of learning a multi-class classifier H^{t+1} using all the past and the current data, D_1, D_2, ..., D_t, D_{t+1} labeled with corresponding classes, l_1, l_2, ..., l_t, l_{t+1}, as well as the past classifier H^t as the knowledge to help train H^{t+1}.

In Chapter 3, we have introduced the problem of open world classification, where unseen classes in testing are expected and can be detected by a classifier. We have also described our proposed open classifier, cbsSVM, by generalizing our center-based similarity space learning (CBS learning) technique in Chapter 2 following an open space risk formulation.

However, being able to detect unseen classes/topics is still insufficient for a multi-class classifier to handle the growing number of topics of interest. We need to incorporate the detected new classes into the system with minimal effort. A naive approach to solving this problem is to re-train the entire system including the new class of data from scratch. However, such a solution is only feasible if the number of classes is small. It is inappropriate and impracticable when the number of classes grows very large.

In this chapter, we propose a new learning strategy, which is inspired by the process of human concept learning, to make an attempt to tackle the *cumulative learning* problem. Human beings are exposed to new concepts all the time. One particular way we learn a new concept is by searching from the already known concepts, looking for similar ones, and then trying to find the difference between these known concepts and the new one without using all the known concepts. For example, assume we have already learned concepts like "movie", "furniture", and "soccer". Now we are presented with the concept of "basketball" and its documents. We find that "basketball" is similar to "soccer", but very different from "movie" and "furniture". Thus we just need to accommodate the new concept "basketball" into our old knowledge base by focusing on distinguishing the "basketball" and "soccer" concepts. We do not need to worry about the difference between "basketball" and "movie" or "furniture", because the existing perception or concepts of "movie" and "furniture" can easily tell that documents from "basketball" do not belong to either of them. Based on the above possible human learning process, the proposed learning process adds a new class of documents to the system that only disturbs a small subset of the past classes.

As we indicated above, cumulative learning is related to lifelong learning (Carlson et al., 2010; Pentina and Lampert, 2014; Silver et al., 2013; Thrun and Mitchell, 1995) because we aim to perform learning continuously to make the system more and more knowledgeable, which is analogous to human learning. However, it is different from current lifelong machine learning (Silver et al., 2013) and transfer learning (Pan and Yang, 2010) methods because existing works in these areas mainly focus on knowledge transfer, i.e., how to make use of the past data or knowledge to help new learning tasks. None of the methods is able to detect unseen classes or incrementally update an existing classifier, which make cumulative learning require a different type of algorithms. Our problem is also different from existing research in online learning and incremental learning (Blum, 1998; Cauwenberghs and Poggio, 2000; Zhao et al., 2011). Online learning aims to handle new instances of the known classes, while we focus on handling unseen/new classes of documents by recognizing them and updating the existing classification system.

In summary, this work makes the following contributions:

1. It proposes the new learning problem of *cumulative learning*, which presents a new form of lifelong learning. It involves two unique challenges, detecting and learning new knowledge (classes or concepts) over time so that the system becomes more and more knowledgeable. The first challenge is called open world classification and we have addressed the problem in Chapter 3.
2. In order to address the second problem of learning new knowledge (classes or concepts) over time, it proposes a learning strategy for cumulatively adding new classes of documents

into the existing classification system without requiring re-training the whole system from scratch.

3. Extensive experiments show that the proposed method gives superior results compared to state-of-the-art baselines in terms of both classification accuracy and learning efficiency.

4.2 Related Work

This work addresses an issue that is related to and has received attention from various machine learning paradigms such as open world classification, lifelong learning, transfer learning, multi-task learning, and online learning. We have covered open world classification in Section 3.2. Now we focus on the rest of the topics.

4.2.1 Lifelong Machine Learning

Lifelong machine learning (or lifelong learning) (Carlson et al., 2010; Pentina and Lampert, 2014; Silver et al., 2013; Thrun and Mitchell, 1995) is a machine learning paradigm that learns continuously, accumulating the knowledge learned in previous tasks and using it to help future tasks. In the process, the learner becomes more and more knowledgeable. However, most existing machine learning techniques learn in isolation. For a given training dataset, a machine learning algorithm learns a model on the dataset. However, it does not try to retain any learned knowledge and use it for future learning tasks.

In the context of supervised learning, early work on lifelong learning focused on transferring invariances in neural networks. For example, memory-based and explanation-based neural networks (EBNN) based methods were proposed in (Thrun, 1996; Thrun and Mitchell, 1995), which transferred knowledge across multiple learning tasks. Their learning task was similar to

ours, but they mainly focused on helping the classification of the new class. Also, their methods were inefficient. Silver and Poirier (2004) made some improvements in terms of efficiency to those in (Thrun, 1996; Thrun and Mitchell, 1995), but the framework was similar. Ruvolo and Eaton (2013b) proposed the Efficient Lifelong Learning Algorithm (ELLA). Ruvolo and Eaton (2013a) further enhanced ELLA through active selection of the next task to learn. However, each of ELLA's learning task is independent of others, i.e., each task's learning and testing are not related to others. Thus, it solves a different problem. Also, none of previous works detects new or unseen classes. Our work is complementary to existing research. Lifelong learning has also been conducted in reinforcement learning (Carlson et al., 2010), and unsupervised topic modeling (Chen and Liu, 2014b; Chen et al., 2015), where past documents of many domains are used to extract knowledge and the knowledge is subsequently used to improve topic discovery in future tasks.

4.2.2 Online and Incremental Learning

Online learning and incremental learning (Blum, 1998; Cauwenberghs and Poggio, 2000; Crammer et al., 2006; Fink et al., 2006; Kakade et al., 2008; Zhao et al., 2011) mainly aim at handling new instances of known classes. In both scenarios, new data instances belonging to the known classes and their class labels are incrementally revealed. The goal of online learning is to make a sequence of accurate predictions in an online manner given the knowledge of the correct answers to previous prediction tasks. However, our problem has a different setting, in which a new class of documents arrives together and online updating is not required. We also detect new classes and update the learned classifier without re-training the entire system. Although

(Xiao et al., 2014) allows new classes of data to be incrementally added, the paper does not detect new/unseen classes, which makes the system less applicable in real-world applications.

4.3 Cumulative Learning

This section presents the proposed learning strategy and process to solve the cumulative learning problem. As discussed in the introduction section, the proposed learning process is similar to that of human concept learning. It cumulates knowledge and uses the cumulated knowledge to help update the existing classification model and to accommodate the new class, so that the new classification model can classify both existing classes and the new class, as well as detecting unseen classes constantly. This section focuses on the overall algorithm and how to incorporate the new class with minimum effort in training by exploiting the existing classification model and the past data as prior knowledge.

4.3.1 Training a Cumulative Classification Model

Assuming we already have an open world classification system at time t with its classification model $H^t = \{h_1, h_2, \ldots, h_t\}$ built for the past t classes $Y^t = \{l_1, l_2, \ldots, l_t\}$ using their corresponding training sets $D^t = \{D_1, D_2, \ldots, D_t\}$. At time $t+1$, the new dataset D_{t+1} of class l_{t+1} arrives, and the classification model H^t needs to be updated or extended to produce a new classification model H^{t+1} to perform open world classification. Each h_i in H^t or H^{t+1} is a 1-vs-rest SVM based on the CBS learning technique introduced in Chapter 2, which is built for class l_i by treating l_i as the positive class. In other words, H^t and H^{t+1} are both open world classifiers, cbsSVM (Chapter 3).

The learning system goes through the following two steps to update the current state of the classification system H^t to build a new one H^{t+1} that can classify test data from all classes in $\{l_1, l_2, \ldots, l_t, l_{t+1}\}$ as well as detect unseen classes of documents denoted by l_0.

1. Searching for a set of similar classes, denoted by SC, that are similar to the new class data D_{t+1} with label l_{t+1}.
2. Learning to separate the new class l_{t+1} from the classes in SC.

In order to perform the first step, we need a way to measure the similarity between classes. There are many possible ways. One way is to perform clustering every time when a new class arrives and see which cluster the new class falls in. However, it is difficult to set the number of clusters as the number of classes changes over time. It is also hard to know how the classes in the same cluster are related in the overall classification problem. Another way to measure the similarity between classes is by computing the similarity between the centers of each class of documents. However, this approach does not know the spread of each class of documents and again, it is not clear how this distance is related to the final classification.

In this work, we propose to quantify the similarity between a new class l_{t+1} and the existing ones l_1, l_2, $\ldots$, l_t by running each of the 1-vs-rest binary classifiers in $H^t = \{h_1, h_2, \ldots, h_t\}$ to classify instances in D_{t+1}. Those past classifiers that accept (classify as positive) a certain number/percentage λ_{sim} of instances from D_{t+1} are regarded as similar classes and are denoted by SC. This method is intuitive because if a past classifier h_i classifies many instances in D_{t+1} as positive, it means that the two classes of data are close to each other and need to be separated subsequently.

Separating the new class l_{t+1} and classes in SC actually involves two steps:

1. Building a binary classifier h_{t+1} for the new class l_{t+1}. It is intuitive to build h_{t+1} for class l_{t+1}, using D_{t+1} as the positive training data and the data of the classes in SC as the negative training data.
2. Updating the existing classifiers for the classes in SC. The reason for the updating is that the addition of class l_{t+1} confuses those classifiers in SC. To re-build each existing classifier h_i, the system needs to use the original negative data employed to build the existing classifier h_i and the new data D_{t+1} as the new negative training data. We still need the old negative training data because we want the new classifier still to be able to separate class l_i from those old classes.

The detailed algorithm is given in Algorithm 1. Line 1 initializes SC to the empty set while Line 3 initializes the variable CT (for count) to record the number of instances in D_{t+1} that are classified as positive by classifier h_i. Lines 4-9 use h_i to classify each instance in D_{t+1} and record the number of instances that are classified (or accepted) as positive by h_i. Lines 10-12 check whether there are too many instances in D_{t+1} that have been classified as positive by h_i to render class l_i as similar to class l_{t+1}. λ_{sim} is a threshold controlling how many percent of instances in D_{t+1} should be classified to class l_i before considering l_{t+1} as similar/close to class l_i. Lines 14-17 build a new classifier h_{t+1} and update all the classifiers for classes in SC.

In summary, the proposed learning process uses the set SC of similar classes to the new class l_{t+1} to control both the number of classifiers need to be built/updated at time $t+1$ and also the number of negative instances used in building the new classifier h_{t+1}. It thus greatly

Algorithm 1 Cumulative Learning

Input: Classification model $H^t = \{h_1, h_2, \ldots, h_t\}$ till time t
Past dataset $\{D_1, D_2, \ldots, D_t\}$ till time t
New dataset D_{t+1} at time $t+1$
Similarity threshold λ_{sim}
Output: Updated classification model $H^{t+1} = \{h_1, \ldots, h_t, h_{t+1}\}$

```
1: SC = ∅
2: for each classifier h_i ∈ H^t do
3:     CT = 0
4:     for each instance d_j ∈ D_{t+1} do
5:         class = h_i(d_j) // classify doc d_j using h_i
6:         if class = l_i then // wrongly classified
7:             CT = CT + 1
8:         end if
9:     end for
10:    if CT > λ_sim × |D_{t+1}| then
11:        SC = SC ∪ {l_i}
12:    end if
13: end for
14: build h_{t+1} and add to H^{t+1}
15: for each h_i of class l_i ∈ SC do
16:     update h_i
17: end for
18: Return H^{t+1}
```

reduces the time compared to that of training a 1-vs-rest multi-class classifier using all the data. However, running existing classifiers to classify instances from the new class will cause some overhead. But the overhead is small compared to the training time needed when the number of classes is very large.

Applying the above cumulative update algorithm together with the cbsSVM open world classifier (Chapter 3), our final system, which is able to handle both challenges in cumulative learning, is called *CL-cbsSVM* (CL stands for Cumulative Learning).

4.3.2 Final Classification Model

Following the cumulative training algorithm discussed in the above subsection, we end up with an updated classifier system $H^{t+1} = \{h_1, h_2, \ldots, h_t, h_{t+1}\}$ that is able to perform open world classification. Since H^{t+1} is essentially a cbsSVM open world classifier, for testing we use the same technique of cbsSVM introduced in Section 3.3.3. That is, for a test instance x, each binary classifier $h_i \in H^{t+1}$ is used to produce a probability $P(l_i|x)$. If none of the probabilities is greater than a decision threshold λ (= 0.5), the document represented by x is regarded as unseen or belonging to l_0; otherwise it is classified to the class with the highest probability.

4.4 Evaluation

In this section, we evaluate our proposed system and compare it with extensive baselines in terms of both classification results and computational speed.

4.4.1 Datasets

For evaluation, we again use the Amazon product reviews dataset (Chen and Liu, 2014a) and the 20-newsgroup dataset (Joachims, 1996). However, in order to mimic real life scenarios and show the superiority of our approach in its efficiency, we need a larger number of classes. Thus, for experiments we use the complete set of 100 classes of Amazon product reviews. For each class/domain in both datasets, we randomly keep 700 of the documents for training and the rest 300 for testing.

4.4.2 Baselines

Our supervised learning baselines can be classified into two categories depending on the strategy they use to update the system given new classes of documents. Most of our supervised learning baselines except for CL-1-vs-rest-SVM are based on the rebuilding strategy, and they have been introduced and used in Section 3.4 as baselines. CL-1-vs-rest-SVM is a variant of our proposed CL-cbsSVM, which is able to cumulatively update the system given a new class of documents. For more complete evaluation, we also include the semi-supervised learning method, Exploratory-EM, which has also been introduced in Section 3.4 as a baseline. All the baseline methods are summarized below. All documents use tf-idf term weighting scheme with no feature selection.

1. **Supervised learning baselines based on rebuilding strategy**. This set of baselines are based on the rebuilding strategy, which means they take all the past data and the new class of data to re-train the system from scratch. We have introduced them in Section 3.4, which include 1-vs-rest multi-class SVM (1-vs-rest-SVM), 1-vs-set Machine (1-vs-set-linear), W-SVM (wsvm-linear and wsvm-rbf), P_I-SVM (P_I-svm-linear and P_I-svm-rbf). We do not compare with P_I-OSVM (P_I-osvm-linear and P_I-osvm-rbf), as they have shown poor results in Section 3.4.
2. **Cumulative Learning with 1-vs-rest SVM (CL-1-vs-rest-SVM)**. Intuitively, the choice of the underlying learner should affect cumulative learning in both classification result and running time. As our proposed cumulative learning process is independent of the learner used in building the classifier, instead of using cbsSVM as the underlying

learner, we apply 1-vs-rest-SVM in the cumulative learning process as one baseline. This is the only supervised learning baseline that supports cumulative update of the model without rebuilding the system from scratch given a new class of documents.

3. **Exploratory Seeded K-Means (Exploratory-EM)**. This baseline is the multi-class semi-supervised method introduced in Section 3.4.

One thing to note is that we did not include any 1-vs-1 SVM based multi-class classification methods as baselines. This is because although a 1-vs-1 SVM technique for multi-class classification can support learning cumulatively by adding t new 1-vs-1 classifiers for the arrival of the $(t+1)$th class, to the best of our knowledge, none of the existing such methods can support open world classification.

4.4.3 Experimental Settings

Similar as in Section 3.4, we conduct open world cross-validation style analysis. In particular, we vary the number of test classes $n = 50, 75, 100$ for Amazon, and $n = 20$ for 20-newsgroup. For each of these choices on the number of test classes, we also select $m = 33\%, 66\%$ and 100% of the number of test classes for training.

When $m = 100\%$ of test classes are used for training, the problem reduces to the closed world classification. In this case, we force all the methods to perform closed world classification by setting $\delta_R = 0$ for W-SVM, $\delta = 0$ for P_I-SVM, and $\lambda = 0$ for methods 1-vs-rest-SVM, CL-1-vs-rest-SVM, cbsSVM and CL-cbsSVM. By doing this, we always assign a known class label to a test instance. For Exploratory-EM, we use its "semisup" mode instead of the "explore" mode, which allows no new classes to be introduced in learning.

For methods that support the proposed cumulative update (CL-cbsSVM and CL-1-vs-rest-SVM), we build a k-class classifier system starting from only 2 classes and cumulatively add new classes to the system one at a time till k classes. We set $\lambda_{sim} = 2\%$ for all methods, as it gives the best results, and we will discuss its effect in Section 4.4.5. For methods that use rebuilding strategy (1-vs-rest-SVM, 1-vs-set-linear, wsvm-linear, wsvm-rbf, P_I-svm-linear, and P_I-svm-rbf, cbsSVM), instead of simulating the whole process of building classifiers starting from 2 classes till k classes by rebuilding the system over and over again, we only build a k-class classifier once using all the k classes to simulate what happens when the kth class arrives.

Same parameters are used for the RBF kernel as in Section 3.4, with $C = 5$, $\gamma = 0.2$ for Amazon and $C = 10$, $\gamma = 0.5$ for 20-newsgroup.

In the following subsections, we first show the classification results of all the methods discussed in this chapter, and then conduct running time analysis in Section 4.4.5 to compare their efficiency. Finally, we perform qualitative analysis of the proposed cumulative learning.

4.4.4 Classification Results

In order to compare the classification results of different systems, for each train-test partition, we first compute precision, recall and F1 score for each class and then macro-average the results across all classes. We only show F1 scores due to too many results.

We show the results of different methods in building a $(m \times n)$-class open world classifier based on the rebuilding and cumulative learning strategies in Table VII and Table VIII. Table VII contains three sub-tables. From left to right, we show results of different methods on the Amazon dataset when the number of test classes $n = 50, 75, 100$. And Table VIII contains

	$n = 50$			$n = 75$			$n = 100$		
	m = 33%	66%	100%	33%	66%	100%	33%	66%	100%
1-vs-rest-SVM	0.498	0.501	0.568	0.442	0.490	0.541	0.460	0.444	0.418
cbsSVM	**0.580**	**0.632**	**0.639**	**0.546**	**0.581**	**0.619**	**0.579**	**0.565**	**0.569**
CL-cbsSVM	0.549	0.610	0.623	0.511	0.574	0.616	0.536	0.552	0.549
CL-1-vs-rest-SVM	0.352	0.511	0.472	0.488	0.440	0.424	0.352	0.373	0.394
1-vs-set-linear	0.437	0.496	0.334	0.379	0.499	0.534	0.379	0.463	0.290
wsvm-linear	0.506	0.537	0.335	0.454	0.535	0.547	0.465	0.499	0.309
wsvm-rbf	0.347	0.382	0.398	0.278	0.357	0.544	0.264	0.289	0.095
P_I-svm-linear	0.507	0.539	0.337	0.454	0.536	0.550	0.465	0.499	0.303
P_I-svm-rbf	0.407	0.595	0.409	0.388	0.576	0.603	0.389	0.562	0.310
ExploratoryEM	0.419	0.523	0.618	0.366	0.514	0.576	0.377	0.480	0.538

TABLE VII

Open classification results on the Amazon dataset.

	$n = 20$		
	m = 33%	66%	100%
1-vs-rest-SVM	0.652	0.714	0.808
cbsSVM	**0.662**	**0.728**	0.852
CL-cbsSVM	0.644	0.716	0.820
CL-1-vs-rest-SVM	0.417	0.632	0.713
1-vs-set-linear	0.620	0.529	0.577
wsvm-linear	0.597	0.606	0.677
wsvm-rbf	0.417	0.643	0.773
P_I-svm-linear	0.598	0.608	0.678
P_I-svm-rbf	0.435	0.715	0.767
ExploratoryEM	0.559	0.690	**0.864**

TABLE VIII

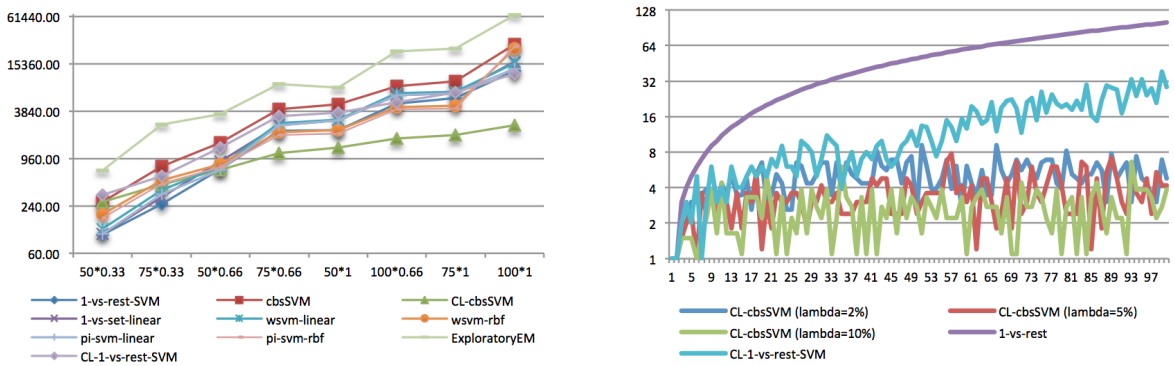
Open classification results on the 20newsgroup dataset.

results for 20newsgroup when $n = 20$. Within each set of results in Table VII and Table VIII, different columns list results of different methods when different proportions of test classes are used for training. From both tables, we can see that cbsSVM (Chapter 3) performs the best in almost all settings with only one exception. Even when 100% of the test classes are used for training, which is in the traditional closed world classification setting. Note that it does not use cumulative update of the system given new classes of documents. That is, it is based on the re-building strategy.

We also notice that the proposed CL-cbsSVM (which is the cumulative version of cbsSVM) always gives comparable results as cbsSVM. It is not surprising that CL-cbsSVM loses to cbsSVM, because cbsSVM uses all the classes when building the classifier while CL-cbsSVM only relies on a small subset of previous classes. This leads to dramatic efficiency increase over cbsSVM as will be shown in Section 4.4.5. In contrast to CL-cbsSVM, the baseline method CL-1-vs-rest-SVM, which also supports learning cumulatively, performs poorly in both open world and closed world classification tasks. The main reason is that its underlying learner SVM learns in the document space. Due to this reason, SVM is not suitable in handling unseen classes, identifying similar domains, or learning effectively given similar classes during the cumulative learning process.

4.4.5 Running Time Analysis

Apart from the classification results, we are also interested in evaluating the efficiency of each method. For comparison, we measure the time it takes for different systems to rebuild or cumulatively update when the last new class is presented in each setting. For methods based



(a) Running time summary

(b) The number of classifiers built

Figure 3. Cumulative learning efficiency analysis.

on the rebuilding strategy, this is equivalent to the total training time because we only build the system once using all the training classes.

In this work, all experiments are conducted using a single thread on a single 64-bit Windows server with Intel Xeon X5650 2.67GHz CPU. At the same time, we are also aware that some of the experimented methods can be easily parallelized. For example, a single classifier can be trained on a different machine for 1-vs-rest-SVM, 1-vs-set-linear and cbsSVM. And for methods based on the proposed cumulative learning process CL-cbsSVM and CL-1-vs-rest-SVM, both detecting similar classes and building/updating existing classifiers can be sent to different machines to speed up the overall running time. However, we plan to leave this part of work to the future.

For all the methods except Exploratory-EM, we implemented data preprocessing and transformation in Java, and called LIBSVM (Chang and Lin, 2011) or its extension packages for

training. Summary of the running time of different methods is shown in Figure 3(a). The x-axis indicates experimental settings. For example, 50×0.33 indicates the setting in which 50×0.33 ($= 16$) classes are used to train, and 50 classes for testing. The reason we show it this way is because Exploratory-EM is a semi-supervised method, the number of test domains also affects its running time. The y-axis indicates running time (in seconds) to rebuild/update the system when the last domain ($50 \times 0.33 = 16$th) arrives. Due to the wide range of running time of different algorithms, the y-axis (seconds) is in log2 scale. However, note that two reasons made the running time of Exploratory-EM algorithm not very comparable. Firstly, the Exploratory Learning package was written in Matlab. Secondly, its a semi-supervised method, which uses test data during learning.

From Figure 3(a), we are able to make several interesting observations. Firstly, apart from Exploratory-EM, cbsSVM takes longer time to run compared to other 1-vs-rest methods, which is due to the extra data preprocessing and transformation overhead and the use of RBF kernel. Secondly, our proposed CL-cbsSVM takes the shortest time to update especially when the number of classes is big. Due to the data transformation overhead, it is slightly slower than some other baselines when the number of classes is small. Thirdly, although also based on the updating strategy, running speed of CL-1-vs-rest-SVM is dramatically slower than that of CL-cbsSVM. This is because CL-1-vs-rest-SVM uses SVM as the underlying learner and is not good at identifying new classes. Thus many existing classifiers misclassify many instances from the new class and get retrained. As the number of classes in the system increases over time, this effect is amplified further. Lastly, most of other algorithms based on 1-vs-rest SVM (1-vs-

rest-SVM, 1-vs-set-linear, wsvm-linear, wsvm-rbf, P_I-svm-linear and P_I-svm-rbf) have similar running time and trend with the increase on the number of training classes/domains.

Although the running time of different methods is affected by many factors, another way of comparing the efficiency of different approaches is to look at the number of classifiers they build/update each time a new class arrives starting from two classes to k classes. In Figure 3(b), we show results for CL-1-vs-rest-SVM, CL-cbsSVM as well as all methods based on the rebuilding strategy. The y-axis indicates the number of classifiers that were built/updated (in log2 scale) and x-axis is the arrival of the ith class. As all the methods based on the rebuilding strategy always build t classifiers when the tth new class is presented to the system, it is thus the upper bound for CL-1-vs-rest-SVM and CL-cbsSVM. To illustrate the effect of λ_{sim} parameter, we vary the parameter $\lambda_{sim} = 10\%, 5\%, 2\%$ for both systems. But for the clarity of the plot, we only show $\lambda_{sim} = 2\%$ for CL-1-vs-rest-SVM, as both methods show similar trends by varying λ_{sim}. Since the sequence of arrival of new classes affects the results of our method, we average the numbers across 5 runs.

From Figure 3(b), we can see that CL-1-vs-rest-SVM clearly builds more classifiers than CL-cbsSVM does when λ_{sim} is set at 2% for both systems. By comparing different λ_{sim} values for CL-cbsSVM, we see that on average smaller λ_{sim} leads to more classifiers being updated. This is intuitive because smaller λ_{sim} indicates being stricter on if an old classifier should be updated, while at the same time, smaller λ_{sim} gives slightly better accuracy in classification, which is also intuitive.

Diaper	Clothing	Headphone	Lamp	Laptop
CL-cbsSVM				
Baby	Baby	CDPlayer	Kindle	HardDrive
Clothing	ArtsCrafts Sewing	CarStereo	Books	Lamp
	Care	Amplifier		Battery
	Beauty	Care		Computer
				Clothing
				GPS
CL-1-vs-rest-SVM				
Baby	Baby	DVDPlayer	Home Improvement	HardDrive
Care	Automotive	CDPlayer	Kindle	Computer
DVDPlayer	Care	CarStereo	Automotive	Clothing
Clothing	Beauty	Amplifier	Industrial Scientific	Automotive
ArtsCrafts Sewing	ArtsCrafts Sewing	Clothing	DVDPlayer	Home Improvement
Automotive	Battery	Care	Clothing	Battery
Beauty	CarStereo	Gloves	Battery	Car Stereo
CDPlayer	CDPlayer	Golf	CDPlayer	DVDPlayer
BluRay Player	Camcorder (2 more)	GPS (1 more)	Care (9 more)	Kindle (11 more)

TABLE IX

Negative classes selected by CL-1-vs-rest-SVM and CL-cbsSVM.

4.4.6 Qualitative Analysis of Cumulative Learning

One way to gain more insight into our proposed cumulative learning system CL-cbsSVM is by looking at what existing classes are selected as the negative training data during its learning process in building/updating classifiers when new classes arrive. Intuitively, an effective and efficient learning algorithm should be able to select the minimal number of closely relevant

domains instead of lots of irrelevant ones. For comparison, we also show the results from CL-1-vs-rest-SVM with the same class arrival order. Table IX lists the negative classes picked by both methods for the 26th till the 30th new class when building the system cumulatively on the Amazon dataset. The first row shows the names of the new classes, and each column shows the chosen similar classes by two systems. The reason we pick these five positions is because there are enough existing classes for a system to make mistakes, and not too many so that we can still list them. In fact, CL-1-vs-rest-SVM still picked too many classes beyond what we can enumerate. We also tried to manually identify those picked classes that we feel are unrelated, and they are marked in red.

From the results shown in Table IX, we can easily tell that not only the number of negative training classes selected by CL-cbsSVM is much smaller, they are also more relevant than those picked by CL-1-vs-rest-SVM.

CHAPTER 5

DETECTING CHANGED-HANDS ONLINE REVIEW ACCOUNTS

(This chapter includes and expands the paper currently under review, "Detecting Changed-Hands Online Review Accounts" in *The 55th Annual Meeting of the Association for Computational Linguistics (ACL'17)* (Fei et al., 2017).)

5.1 Introduction

Reputable social media accounts can be of great value to spammers with hidden agendas. A common type of spam in e-commerce (i.e., review sites such as Amazon, Yelp, etc.) or review sites is the *opinion spam*, because individuals and organizations rely heavily on online reviews/opinions to make their purchase decisions. Due to huge profits brought by positive reviews, opinion spammers game the system by posting fake reviews to promote or discredit some target products or services. Since the early work by Jindal and Liu (2008), detecting fake reviews and reviewers have drawn wide attention from both the research community and the industry.

A large number of methods have been proposed in recent years to fight against opinion spam and the problem has been investigated through different approaches, including linguistic (Ott et al., 2011; Li et al., 2014b), behavioral (Feng et al., 2012b; Ye and Akoglu, 2015), temporal (Xie et al., 2012; Li et al., 2015; Ye et al., 2016) and relational analysis (Wang et al., 2011; Jiang

et al., 2014; Rayana and Akoglu, 2015; Wang et al., 2016). (Please see Section 5.2 for additional related work.)

As a result of the advances in spam filtering techniques, spamming has become harder than before. For example, giving all-extreme ratings (Lim et al., 2010), writing near-duplicate reviews (Jindal and Liu, 2008), or posting many reviews in a short period of time (Fei et al., 2013) can be easily caught by an existing filtering system. Driven by profits, opinion spammers are forced to resort to other strategies. One of the strategies is offering to buy reputable accounts[1] (those with a clean history) and use them to post spam reviews. Selling/buying accounts is also prevalent in other forms of social media. Karma farmers[2] are such an example in the community website Reddit[3], who try to gain high karma (upvotes and reputation) quickly with new accounts so that their posts can show up in the front page, and then sell these seemingly benign and reputable accounts to spammers. In rare cases, accounts may also change hands when they are compromised by spammers, while it is harder to break into frequently-used reputable accounts.

In all of the above situations, accounts change hands at a certain time point and start to exhibit linguistic and/or behavioral differences in the midst of their life span. To the best of our knowledge, such changes, as a new form of spam, have not been studied before. Moreover,

---

[1] https://www.yelp.com/topic/boston-someone-offered-to-buy-my-yelp-account

[2] https://www.reddit.com/r/AgainstKarmaWhores/comments/3gwmae/types_of_karma_whores/

[3] https://www.reddit.com/r/AgainstKarmaWhores/comments/383qsp/why_are_we_doing_this/

as most existing spam filtering methods study the overall language usage and behaviors of reviewers, they are less likely to detect such accounts since they look normal in their early stage.

In this chapter, we propose to fill the gap to investigate the problem of detecting CH accounts from a linguistic perspective. An algorithm, called CHAD (*CH Accounts Detection*), is proposed to identify whether an account has changed hands and to estimate the time point of change if so. In case of a change, we assume there is only one change in an account's life time. This is reasonable as it is sufficient for spam detection purposes. Formally, our problem definition is as follows:

Problem Definition: Given an account $\boldsymbol{A} = \{r_1, r_2, \ldots, r_n\}$ with reviews r_j sorted by their posting dates, CHAD determines if a significant linguistic change has occurred starting from a particular review r_i $(1 < i < n)$. The algorithm returns i if yes, and returns *none* otherwise.

The proposed algorithm CHAD is based on the idea that reviews/posts written by one user are similar among themselves but different from those written by another user. It works in five main steps: (1) It computes a sequence of similarity scores by comparing reviews in a small window (called *pivot window*) with a moving window on the rest of the reviews of the same account based on a feature in a set of m features. Thus each pivot window generates m similarity sequences. (2) It eliminates noisy features and their sequences from the set of sequences for each pivot window. (3) It aggregates the remaining sequences for each pivot window. (4) It runs a statistical algorithm for *change-point* detection on each aggregated sequence. (5) It performs

two rounds of voting on the change-point detection results of the aggregated sequences of all pivot windows for each account to identify whether the account has changed hands and where the change has occurred.

Our problem presents two unique challenges:

1. **Inter-user differences**: Different CH accounts exhibit different changes, because not every pair of users has the same differences in their writings. For example, in some CH accounts, the two users can be distinguished by the average length of the words they use, e.g., one of the users likes to use short words, while the other uses long words. In some other CH accounts, the two users may be distinguished by the average sentence length but not by the average word length, because one user likes to use long sentences while the other likes to use short ones, but both of them mainly use short words.
2. **Intra-user variance**: Every review is unique in some way, which results in a certain amount of difference and variance when compared with other reviews of even the same user using some features in step (1). However, such differences may not indicate a real changing of hands between two users.

Given the above two challenges, an effective detection method needs to perform the above five steps on individual accounts (account level). In fact, step (2) of the proposed method CHAD uses a novel feature selection technique, which is very different from traditional feature selection approaches, on the similarity sequences constructed ‘for individual pivot windows (pivot level). It performs *pivot-level feature selection* which aims to address both challenges at the same time. We describe the details in Section 5.3.

The contributions of this work are as follows:

1. We study the outstanding problem of detecting CH accounts, which have become prevalent in many social media websites. The proposed approach is complementary to existing literature in spam detection and especially opinion spam detection on review websites.
2. We propose a novel algorithm, called CHAD, which leverages linguistic evidences to identify whether an account has changed hands during its life time and estimates the change point.
3. We evaluate our proposed algorithm CHAD on two datasets constructed from Amazon and Yelp reviews, and demonstrate that the approach is highly effective and superior to a list of baselines.

5.2 Related Work

Our work is related to opinion spam detection, tracking linguistic evolution and change point detection. We discuss these related work in turn here.

5.2.1 Opinion Spam Detection

The spamming activities that have received the most attention from the research community are web spam and email spam (Castillo et al., 2007; Spirin and Han, 2012; Chirita et al., 2005). However, opinion spam differs considerably in its ultimate goals and execution by the spammers (Jindal and Liu, 2008).

Since the first work by Jindal and Liu (2008), opinion spam has been widely studied. A wide range of techniques have been proposed for detecting individual spam reviews (Jindal and Liu, 2008; Li et al., 2011; Wang et al., 2011; Ott et al., 2012; Feng et al., 2012b; Feng et al., 2012a; Xie et al., 2012; Li et al., 2014a; Li et al., 2014b; Li et al., 2015; KC and Mukherjee,

2016; Hai et al., 2016), individual spammers (Jindal et al., 2010; Lim et al., 2010; Akoglu et al., 2013; Fei et al., 2013; Mukherjee et al., 2013a) and spammer groups (Mukherjee et al., 2012; Xu et al., 2013; Xu and Zhang, 2015; Ye and Akoglu, 2015). However, the use of CH accounts as a new instrument for spamming has not been studied thus far and no techniques are available for their detection.

Among existing techniques, detecting individual spammers is the most related task to our problem. In particular, Jindal et al. (2010) and Lim et al. (2010) studied rating behaviors of users; Akoglu et al. (2013) studied the relational collusion between reviewers and their target products; Mukherjee et al. (2013a) proposed a Bayesian approach to modeling the behavioral patterns of spammers and non-spammers; Fei et al. (2013) exploited review bursts for spammer detection. However, none of these approaches can effectively detect CH accounts because they examine the overall linguistic and/or behavioral patterns of each user account, while the spamming activities of CH accounts may go undetected by existing systems given a clean early history.

Our work is also related to using linguistics-based approaches to detecting individual spam reviews (Ott et al., 2011; Li et al., 2014b; Ren et al., 2014; Hai et al., 2016). In particular, Ott et al. (2011) hired Amazon turkers to write fake reviews and created a gold standard fake review dataset which can be used for developing supervised spam review detection models. Li et al. (2014) created a more comprehensive benchmark on three domains of reviews (Hotel, Restaurant, and Doctor), and explored generalized approaches for identifying online spam reviews. Our work is also loosely related to psycholinguistic deception detection (Feng et al.,

2012a; Newman et al., 2003; Hancock et al., 2007; Pérez-Rosas et al., 2015; Pérez-Rosas and Mihalcea, 2015), as we also use a linguistic-based approach in this work.

5.2.2 Tracking Linguistic Evolution

Our work is related to tracking linguistic evolution across time. Juola (2003) quantified the rate of change in language across two time periods. Säily et al. (2011) studied variations in noun/pronoun frequencies and Lijffijt et al. (2012) studied lexical stability in a historical corpus. Our work is different because the above works study linguistic changes by comparing language from two chosen time periods, while our goal is to estimate the change point from a sequence of documents. Tracking shifts in the meaning of individual words has been studied in (Wijaya and Yeniterzi, 2011; Gulordava and Baroni, 2011; Mitra et al., 2014; Kim et al., 2014; Kulkarni et al., 2015). Our work does not study shifts of meaning of words but "shifts in authorship". Authorship attribution methods (Stamatatos, 2009), while related, cannot be directly applied to a sequence of documents.

5.2.3 Change Point Detection

Change point detection is a core time series analysis problem (Basseville et al., 1993; Chen and Gupta, 1999; Taylor, 2000; Adams and MacKay, 2007). For extended surveys on change point analysis theory and applications, please refer to (Basseville et al., 1993) and (Chen and Gupta, 2012). In our work, we adopt the *single change point* detection technique of (Chen and Gupta, 1999), as it aligns with our goal of detecting *CH accounts*.

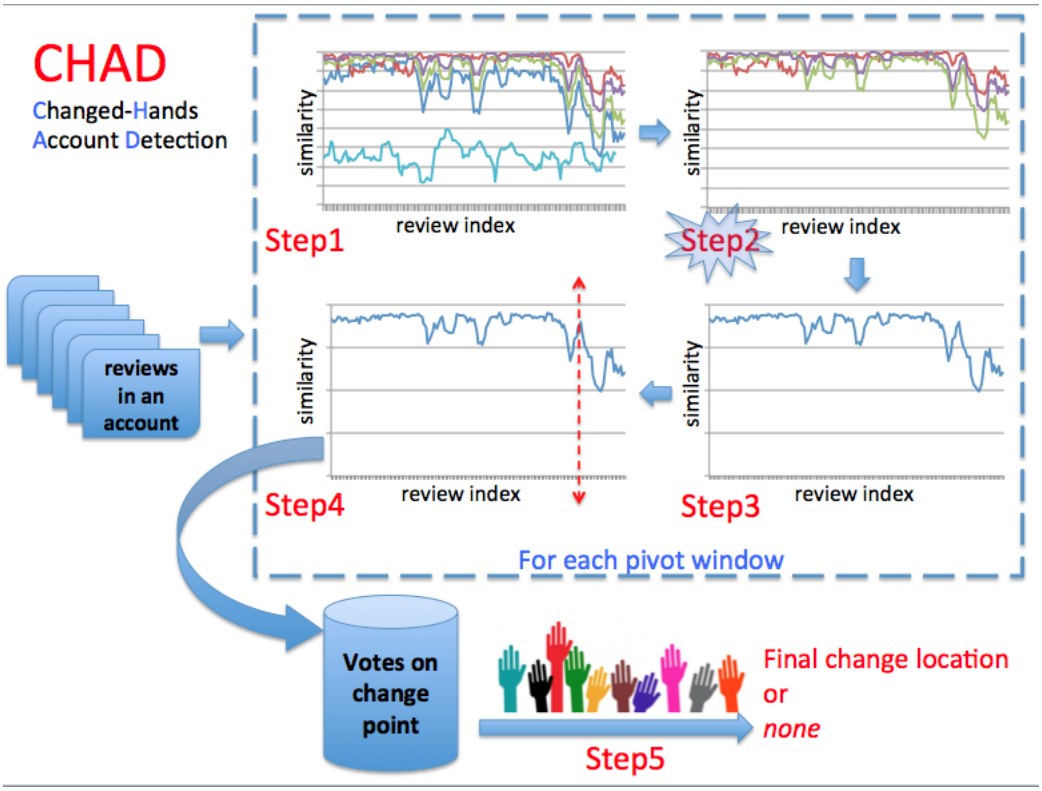


Figure 4. Five main steps of CHAD.

5.3 Proposed Method

This section presents our CHAD algorithm for detecting CH accounts. We start with describing the overall algorithm, and then go into the details.

5.3.1 The Overall Algorithm

As mentioned earlier, the main idea of CHAD is that the reviews written by one user are similar among themselves but different from those written by a different user. CHAD is outlined in Algorithm 2, which works on one account at a time. Note that it needs a pre-selected

Algorithm 2 CHAD

Input: Account := $\boldsymbol{A} = \{r_1, r_2, \ldots, r_n\}$,
Window size := K, Smoothing factor := λ_S
Features := $\boldsymbol{F}(\subseteq \boldsymbol{F}^{all}) = \{f_1, f_2, \ldots, f_m\}$
Output: Result := index i $(1 < i < n)$, or *none*

1: $\boldsymbol{C} := \emptyset$ // $\boldsymbol{C}$ is a multiset for voting.
2: **for each** $r_i \in \boldsymbol{A}(1 \leq i \leq n - K + 1)$ **do**
3: $\boldsymbol{S}_i := \emptyset$ // $\boldsymbol{S}_i$ is a set of similarity sequences for a pivot window.
4: *pivot-window* := $\{r_i, \ldots, r_{i+K-1}\}$
5: $\overline{\boldsymbol{A}} := \boldsymbol{A} \setminus$ *pivot-window*
6: **for each** $f_j \in \boldsymbol{F}$ **do**
7: ss_{ij} := compute-sim-seq$(r_i, K, \overline{\boldsymbol{A}}, f_j)$
8: $\boldsymbol{S}_i := \boldsymbol{S}_i \cup \{ss_{ij}\}$
9: **end for**
10: $\boldsymbol{S}_i$:= pivot-level-feature-select$(\boldsymbol{S}_i)$
11: s_i := aggregate$(\boldsymbol{S}_i)$
12: c_i := change-point-detect(s_i) // c_i is either a review's temporal index or *none*.
13: $\boldsymbol{C} := \boldsymbol{C} \cup \{c_i\}$
14: **end for**
15: *result* := is-change-vote$(\boldsymbol{C})$
16: **if** (*result* $\neq$ *none*) **then**
17: $\boldsymbol{C}_{-none}$:= remove *none* elements from $\boldsymbol{C}$
18: $\boldsymbol{C}^S_{-none}$:= smooth$(\boldsymbol{C}_{-none}, \lambda_S)$
19: *result* := change-point-vote$(\boldsymbol{C}^S_{-none})$
20: **end if**
21: **return** *result*

feature set $\boldsymbol{F}$ as input, which is a subset of all features $\boldsymbol{F}^{all}$. We will explain this shortly. We first introduce its five main steps (which is also illustrated in Figure 4):

1. *Generate similarity sequences* (lines 2-9): This step builds a set of similarity sequences $\boldsymbol{S}_i$ using features in $\boldsymbol{F}$ (Section 5.3.2) for a *pivot window* (size K) starting from review r_i. Each sequence $ss_{ij} \in \boldsymbol{S}_i$ is computed by comparing the similarity of reviews in the pivot window

and reviews within a moving window of also size K in the remaining reviews $\overline{\boldsymbol{A}}$ using one (f_j) of the features in $\boldsymbol{F}$ (line 7). For a CH account, we expect the similarities in a sequence to be high when comparing reviews written by the same user, but low across two different users. We expect to detect such changes using a change-point detection algorithm (Section 5.3.4).

2. *Eliminate noisy features* (line 10) (Section 5.3.3): This step performs pivot-level feature selection and removes the corresponding noisy sequences from $\boldsymbol{S}_i$. This step is the key to addressing the main challenges of our problem.
3. *Aggregate sequences* (line 11): We aggregate the remaining sequences for each pivot window by averaging the sequences in the resulting $\boldsymbol{S}_i$.
4. *Change-point detection* (line 12) (Section 5.3.4): We run a statistical algorithm for *change-point* detection on each aggregated sequence s_i.
5. *Two-round Voting* (line 15-20) (Section 5.3.5): We perform two rounds of voting on the change-point detection results on the aggregated sequences of all pivot windows for an account to determine if CH occurs and also identify the final change point.

Global feature pre-selection: As mentioned before, CHAD requires a pre-selected feature set $\boldsymbol{F}$ as input. $\boldsymbol{F}$ is selected globally by running Algorithm 2 without line 10 (pivot-level feature selection) on all accounts of a development set for multiple iterations starting with all features $\boldsymbol{F}^{all}$ as input. Each iteration removes one feature from $\boldsymbol{F}^{all}$ that gives the biggest performance gain in F1 score under the change-point evaluation ($eval_{cp}$) (Section 5.4.1). It globally removes those features in $\boldsymbol{F}^{all}$ that do not help our task. We will give more insights for this in Section 5.3.3.

5.3.2 Features and Similarity Metrics

We first introduce the set of all features $\boldsymbol{F}^{all}$ and similarity metrics used in the *compute-sim-seq* function in Algorithm 2 (line 7). Features with * produce a single value for reviews in a given window and the rest use the Bag-of-Words (BOW) model.

- **Average sentence length***. A single value feature that computes the average number of words in a sentence in a review window.
- **Average token length***. A single value feature that computes the average number of characters in a word in a review window.
- **Word unigrams**. All word unigrams in a review window represented by a BOW model.
- **Word bigrams**. All word bigrams in a review window represented by a BOW model.
- **Part-of-Speech unigrams**. All Part-of-Speech tags of unigrams in a review window represented by a BOW model.
- **Part-of-Speech bigrams**. All Part-of-Speech tags of bigrams in a review window represented by a BOW model.
- **Adjectives & adverbs**. All adjectives and adverbs in a review window represented by a BOW model.
- **Nouns**. All nouns in a review window represented by a BOW model.
- **Function words**. All function words in a review window represented by a BOW model.
- **Punctuations**. All punctuations in a review window represented by a BOW model.

To measure the similarity between two review windows, we use cosine similarity for BOW features. We tried some other measures but they did not perform well. For single value features, we compute the similarity *sim* using their absolute difference *diff* and normalizing it to [0,1]:

$$sim = 1/(1 + log(1 + diff)) \tag{5.1}$$

5.3.3 Pivot-Level Feature Selection

Now we describe the *pivot-level-feature-select* function in line 10 of Algorithm 2. As we pointed out earlier, one key challenge of our problem is that the writing differences between the pair of users in one CH account may be different from those between other pairs in other CH accounts. Furthermore, each review is unique in some way which can result in a certain amount of difference when computing similarity with other reviews using some features. Such differences however may not indicate real writing differences between two users. To solve the above two issues, we propose to perform pivot-level feature selection (Algorithm 3). The corresponding similarity sequence of each removed feature is deleted from $\boldsymbol{S}_i$.

The *pivot-level-feature-select* function selects a subset of sequences in $\boldsymbol{S}_i$ through correlation analysis, which is the same as selecting their corresponding features. It first averages all sequences in $\boldsymbol{S}_i$ to construct a *target* sequence (line 1). It then computes Pearson's correlation (Pc) of each sequence in $\boldsymbol{S}_i$ with the *target* and sorts the sequences based on correlation strength in descending order (line 3). Line 4 adds the sequence with the highest correlation to the result set $\boldsymbol{E}$. It then goes through the sorted sequence set $\boldsymbol{S}_i^{'}$ and tests if adding another

Algorithm 3 Pivot-level-Feature-Select

Input: $\boldsymbol{S}_i := \{ss_{i1}, ss_{i2}, \ldots, ss_{iT}\}$
Output: $\boldsymbol{E}$:= the set of selected sequences

1: $target := \text{avg}(\boldsymbol{S}_i)$
2: $\boldsymbol{E} = \emptyset$
3: $\boldsymbol{S}_i^{'} = \text{sort}(\{\text{Pc}(ss_{ij} \in \boldsymbol{S}_i, target)\})$ // computes Pearson's correlation (Pc) of each $ss_{it} \in \boldsymbol{S}_i$ to *target* and sort in descending order.
4: $\boldsymbol{E} := \boldsymbol{E} \cup \{\boldsymbol{S}_i^{'}[1]\}$ // adds the sequence with highest correlation to $\boldsymbol{E}$.
5: **for** $j \in \{2 : T\}$ **do**
6: $p := \text{avg}(\boldsymbol{E})$
7: $l := \boldsymbol{S}_i^{'}[j]$
8: **if** $\text{Pc}(\text{avg}(\boldsymbol{E} \cup \{l\}), target) > \text{Pc}(p, target)$ **then**
9: $\boldsymbol{E} := \boldsymbol{E} \cup \{l\}$
10: **else**
11: **break**
12: **end if**
13: **end for**
14: **return** $\boldsymbol{E}$

sequence to $\boldsymbol{E}$ would increase $\boldsymbol{E}$'s average's correlation to *target* (lines 5-13). If the correlation improves by the addition, we update $\boldsymbol{E}$; otherwise exit and return $\boldsymbol{E}$.

The intuition is that *target* is a representative sequence assuming only a few noisy sequences exist. Through correlation analysis, we identify sequences (or their features) that align with the *target* and discard those outlier ones that otherwise hinder the change point detection performance. Given the intuition above, it becomes clearer why we perform global feature selection for CHAD. Because the *pivot-level-feature-select* function considers *target* as a reference and assumes only a few noisy sequences exist, globally removing noisy features is able to enhance its effectiveness, as we will see in Section 5.4.4.

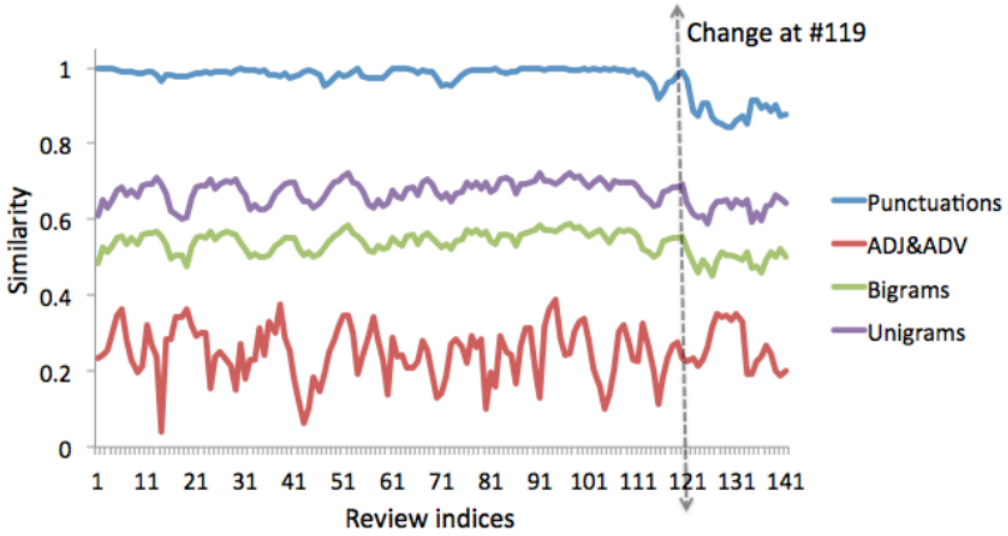


Figure 5. Sample similarity sequences in an $\boldsymbol{S}_i$.

Figure 5 gives an example of several similarity sequences computed for a pivot window on a CH account in the Amazon dataset (Section 5.4.1), where the actual change happens at review #119. For clarity, we only plot a subset of sequences generated using 4 features. We can see that changes in similarity can be captured by features such as punctuations, unigrams and bigrams and are reflected on their corresponding similarity sequences, while the sequence constructed using Adj&Adv is quite noisy and no change can be spotted. Through correlation analysis, our Algorithm 3 is able to effectively eliminate the noisy sequence generated from the feature Adj&Adv.

5.3.4 Change Point Detection

As mentioned in the introduction, we assume there is at most one change point in each account. As such, we use the single point change detection algorithm by (Chen and Gupta, 1999), which uses the Schwarz Information Criterion (SIC) (Schwarz and others, 1978) to search for the change point. We briefly review the technique in (Chen and Gupta, 1997; Chen and Gupta, 1999) in this subsection. Suppose $X_1, X_2, \ldots, X_n$ is a sequence of independent Gaussian random variables with means $\mu_1, \mu_2, \ldots, \mu_n$ and variances $\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2$, respectively. The method tests the hypothesis of whether there exists a single change point at an unknown position k, $2 \leq k \leq n-1$, where the mean and variance change:

$$H_0 : \mu_1 = \mu_2 = \cdots = \mu_n = \mu \text{ and}$$

$$\sigma_1^2 = \sigma_2^2 = \cdots = \sigma_n^2 = \sigma^2,$$

versus the alternative hypothesis

$$H_1 : \mu_1 = \cdots = \mu_k \neq \mu_{k+1} = \cdots = \mu_n \text{ and}$$

$$\sigma_1^2 = \cdots = \sigma_k^2 \neq \sigma_{k+1}^2 = \cdots = \sigma_n^2,$$

where μ and σ^2 are unknown common parameters when there are no changes. We thus have two models corresponding to the H_0 and H_1. The decision to reject H_0 is made based on the principle of minimum Schwarz information criterion (Schwarz and others, 1978), SIC. In particular, H_0 is not rejected if $\text{SIC}(n) \leq \min_k \text{SIC}(k)$, and rejected otherwise. $SIC(n')$ is defined as

$-2\log L(\hat{\Theta}) + p\log n'$, where $L(\hat{\Theta})$ and p are the respective maximum likelihood function and degree of freedom for each model, and n' being the sample size.

More specifically, the MLE's for μ and σ^2 under H_0 can be found to be: $\hat{\mu} = \bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$, and $\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2$. Then the SIC under H_0 (Chen and Gupta, 1997), SIC(n), is obtained as:

$$\begin{aligned} SIC(n) &= -2logL_0(\hat{\mu}, \hat{\sigma}^2) + 2logn \\ &= nlog2\pi + nlog\sum_{i=1}^{n}(X_i - \bar{X})^2 + n + (2-n)logn, \end{aligned} \tag{5.2}$$

in which the function L_0 is the maximum likelihood function under H_0.

While under H_1, the MLE's for μ_1, μ_n, ${\sigma_1}^2$, ${\sigma_n}^2$ can be found to be: $\hat{\mu}_1 = \bar{X}_k = \frac{1}{k}\sum_{i=1}^{k} X_i$, $\hat{\sigma}_1^2 = \frac{1}{k}\sum_{i=1}^{k}(X_i - \bar{X}_k)^2$, $\hat{\mu}_n = \bar{X}_{n-k} = \frac{1}{n-k}\sum_{i=k+1}^{n} X_i$, $\hat{\sigma}_n^2 = \frac{1}{n-k}\sum_{i=k+1}^{n}(X_i - \bar{X}_{n-k})^2$. And the SIC under H_1, SIC(k) for a particular k ($2 \le k \le n-2$), is thus defined as:

$$\begin{aligned} SIC(k) &= -2logL_1(\hat{\mu}_1, \hat{\mu}_n, \hat{\sigma}_1^2, \hat{\sigma}_n^2) + 4logn \\ &= nlog2\pi + klog\hat{\sigma}_1^2 + (n-k)log\hat{\sigma}_n^2 + n + 4logn, \end{aligned} \tag{5.3}$$

where $L_1(\hat{\mu}_1, \hat{\mu}_n, \hat{\sigma}_1^2, \hat{\sigma}_n^2)$ is the maximum likelihood function under H_1.

H_0 cannot be rejected if $SIC(n) \le \min_k SIC(k)$, and can otherwise be rejected if $SIC(n) > \min_k SIC(k)$, and under such circumstance the change point k can be estimated by $\hat{k}$ such that $SIC(\hat{k}) = \min_k SIC(k)$. Here, $SIC(k)$ is the test statistic for the model selection.

5.3.5 Two-Round Voting

In Algorithm 2, CHAD uses a two-round voting scheme (step 5) to determine if an account has changed hands and also to pinpoint the location of change (lines 15-20). In the first round (line 15), *is-change-vote* function determines if a change has occurred. Note that each element $c_i \in \boldsymbol{C}$ returned by the change point detection algorithm is either a change point (i.e., a review) or *none* (indicating no change). This function simply counts the number of votes for each change point and *none*. If *none* has the highest number of votes, it returns *none*; otherwise it registers that change has occurred and removes all the *none* elements from $\boldsymbol{C}$ (line 17). It then moves on to the second round of voting to pinpoint the actual change location. Instead of directly voting based on elements in $\boldsymbol{C}_{-none}$, we perform *smoothing* on $\boldsymbol{C}_{-none}$ first (line 18). Let us look at an example. Given a set of votes in $\boldsymbol{C}_{-none}$ in the format of change-point:#-of-votes 10:8, 50:5, 51:7, 52:4, the point that gets the highest votes is 10. However, the actual change point is more likely to be around 51. In order to overcome this possible noise factor, we smooth the votes by adding some extra counts to near-by locations of every change point in $\boldsymbol{C}_{-none}$ to construct $\boldsymbol{C}^S_{-none}$. Specifically, we pick a smoothing factor $\lambda_S \in \mathbb{Z}_{>0}$ and for a change point i with v votes, we add v/λ_S^{dist} extra votes to locations $i + dist$ and $i - dist$, where $dist = 1, 2, \ldots$. Finally, we perform the second round of voting on $\boldsymbol{C}^S_{-none}$ to determine the final change location. Assuming $\lambda_S = 2$, the final votes in the above example becomes 7:1, 8:2, 9:4, 10:8, 11:4, 12:2, 13:1, 48:1, 49:3, 50:9, 51:10, 52:8, 53:3, 54:1, and the 51th review is selected as the final change point.

5.4 Experiments

In this section, we evaluate our proposed changed hands detection algorithm, CHAD, and compare it with a list of baselines. We start by introducing two datasets, followed by the evaluation schemes and metrics.

5.4.1 Datasets and Evaluation Metrics

Datasets: Identifying opinion spam manually has been shown to be very challenging and unreliable by previous works (Ott et al., 2011). As such, no publicly available labeled data exists for our experiments. We thus constructed synthetic datasets for this work. Similar construction has been done previously, e.g., in identifying the same author under multiple userids (Qian and Liu, 2013). We use two publicly-available review datasets to construct CH accounts, one from Amazon (Jindal and Liu, 2008), which contains reviews for multiple product categories such as books, electronics and etc., and the other from Yelp (Mukherjee et al., 2013b), which contains only hotel reviews. The construction of CH accounts from each dataset is done as follows: We first randomly select two different accounts $\boldsymbol{A}_1 = \{r_{11}, r_{12}, \ldots, r_{1n}\}$ and $\boldsymbol{A}_2 = \{r_{21}, r_{22}, \ldots, r_{2n'}\}$, both sorted by their review posting dates. We then concatenate one account to the other, giving us $\boldsymbol{A}_{syn} = \{r_{11}, r_{12}, \ldots, r_{1n}, r_{21}, r_{22}, \ldots, r_{2n'}\}$. For each dataset, we created 700 accounts, 350 CH accounts and 350 normal (i.e., original) accounts. We then randomly sample 200 accounts from each set, 100 in each class, as the development set and consider the rest 500 accounts as the test set.

We believe that this construction method is reasonable because this is what usually happens with CH accounts. Also since we randomly select accounts from a large dataset, which may already contain some CH accounts, the chance of selecting existing CH accounts is very small.

We next explain why we use the Amazon and Yelp datasets in our experiments. The Amazon dataset has reviews of all kinds of products. We use it to create the scenario where a spammer buys an account and uses it to review products that are potentially quite different from those reviewed by the original user (although we do not enforce this when constructing the dataset). Moreover, products reviewed by a single user can also be quite diverse. In contrast, the Yelp dataset has only hotel reviews, which allows us to show whether our approach is able to detect CH accounts when the two users wrote reviews for the same type of entities. As we show below, CHAD is able to perform well on both scenarios.

Evaluation Schemes and Metrics: We deploy two evaluation schemes, *CH-accounts detection* evaluation ($eval_{cha}$) and *change-point detection* evaluation ($eval_{cp}$), and report corresponding precision, recall, F1, and accuracy on both tasks.

For $eval_{cha}$, we only identify CH accounts but not the actual change locations. For $eval_{cp}$, we go one step further to also evaluate the change point locations. Since it is quite hard to identify the exact change point (the review) at which a change-of-hands has occurred, we define a window $x \pm y$ around the actual change point x with window size y, and consider the predicted change point as accurate as long as it resides within the window. When a change point is detected for a non-CH account, it is considered an error. We study performance by varying y at the end of section 5.4.4.

5.4.2 Baselines

Since there is no previous work on detecting changed-hands accounts, we propose the following baselines:

- **One Sequence (OS)**. One of the simplest approaches to detecting CH accounts is to construct a single sequence of feature values directly from a moving window of reviews of size K (one value per review window) of an account and use it to run a change point detection algorithm. The set of features we tried includes: average sentence length, average token length, ratio of nouns, ratio of adjectives and adverbs, ratio of function words, and ratio of punctuations and special characters. This baseline thus produces 6 results named with OS- as the prefix.
- **One Feature (*OF*)**. This baseline is a variant of CHAD. The difference is that it only uses a single feature from $\boldsymbol{F}^{all}$ as input. Thus, lines 10-11 in Algorithm 2 do not have any effect. Since $\boldsymbol{F}^{all}$ contains 10 features, this baseline produces 10 different results, which are named with OF- as the prefix.
- **CHAD w/out Pivot-level Feature Selection (*CHAD-PFS*)**. This baseline is another variant of CHAD. The difference is that it does not perform pivot-level feature selection in Algorithm 2. It deploys the same procedure to pre-select a feature set and thus shares the same $\boldsymbol{F}$ with CHAD.
- **CHAD w/out Pre-selecting F (*CHAD-F*)**. This is a variant of CHAD that directly uses $\boldsymbol{F}^{all}$ without pre-selecting feature set $\boldsymbol{F}$ as input. It still performs pivot-level feature selection.

- **CHAD with Individual Voting (*IndV*)**. This baseline is also a variant of CHAD. The main difference is in lines 10-13 of Algorithm 2. Specifically, it does not perform pivot-level feature selection (line 10) or sequence aggregation on $\boldsymbol{S}_i$ (line 11). Instead, it performs change-point detection on every individual sequence in $\boldsymbol{S}_i$ and adds every result to $\boldsymbol{C}$ for final voting. It does use pre-selected feature set $\boldsymbol{F}$, however, by running IndV itself on the development set for multiple iterations. Each iteration removes one feature until the performance does not increase.

5.4.3 Parameter Settings

We use the respective development set of each dataset to set parameters. For both datasets, $K = 5$ was chosen for the window size in constructing similarity sequences (Section 5.3.1), and $\lambda_S = 2$ was chosen for vote smoothing (Section 5.3.5). For change-point detection, we use the implementation in R *changepoint* package (Killick and Eckley, 2014), which outputs an estimated change point along with its confidence level. We set a confidence level threshold of $\theta_{conf} = 0.99$ based on the development set, and consider any detected change point with confidence level lower than θ_{conf} as no change ($none$). In parameter selection and in the main results reporting, we use $y = 5$ for $eval_{cp}$, and later show the results by varying y.

We make the following remarks about these parameters. First, using a very small K (e.g., 1 or 2) leads to bad performance due to the high variance in similarities between reviews. On the other hand, while using a large K (e.g., 7 or 8) improves results for CH accounts detection ($eval_{cha}$), the performance of change-point detection ($eval_{cp}$) drops due to loss of granularity. Second, it is important to set the confidence level threshold θ_{conf} high to consider only the most

	$eval_{cha}$				$eval_{cp}$			
Feature Name	Prec.	Recall	F1	Accu.	Prec.	Recall	F1	Accu.
One Sequence Baselines								
OS-AvgSentLen	0.681	0.76	0.718	0.702	0.365	0.629	0.462	0.526
OS-AvgTokenLen	0.736	0.692	0.713	0.722	0.353	0.518	0.420	0.542
OS-Punctuations_Ratio	0.624	0.812	0.706	0.662	0.298	0.673	0.413	0.45
OS-Adj&Adv_Ratio	0.518	0.344	0.413	0.512	0.132	0.118	0.125	0.384
OS-Noun_Ratio	0.449	0.268	0.335	0.47	0.067	0.051	0.058	0.356
OS-Function_Word_Ratio	0.663	0.632	0.647	0.656	0.239	0.382	0.294	0.454
One Feature Baselines								
OF-Unigrams	0.832	0.776	0.803	0.81	0.639	0.726	0.680	0.72
OF-Bigrams	0.824	0.768	0.795	0.802	0.639	0.719	0.677	0.716
OF-Punctuations	0.778	0.8	0.788	0.785	0.548	0.738	0.629	0.667
OF-Function_Words	0.832	0.576	0.680	0.73	0.606	0.497	0.546	0.652
OF-POS_Unigrams	0.865	0.672	0.756	0.784	0.644	0.603	0.623	0.698
OF-POS_Bigrams	0.872	0.684	0.766	0.792	0.632	0.610	0.621	0.698
OF-Nouns	0.733	0.804	0.767	0.756	0.536	0.75	0.625	0.648
OF-Adj&Adv	0.821	0.496	0.618	0.694	0.602	0.419	0.494	0.628
OF-AvgSentLen	0.703	0.616	0.656	0.678	0.328	0.428	0.372	0.514
OF-AvgTokenLen	0.810	0.496	0.615	0.69	0.398	0.326	0.358	0.564
CHAD Variants								
IndV	0.901	0.584	0.708	0.76	0.820	0.561	0.666	0.734
CHAD-PFS	0.864	0.792	0.826	0.834	0.751	0.767	0.759	0.782
CHAD-F	0.879	0.844	0.861	0.864	0.729	0.817	0.770	0.792
CHAD								
CHAD	0.876	0.852	**0.864**	**0.866**	0.757	0.832	**0.793**	**0.808**

TABLE X. Results of CH detection on the Amazon accounts.

confident detections—due to the fact that each review is unique in some way, the constructed similarity sequences unavoidably fluctuate to a large extent.

5.4.4 Results and Analysis

Now we show the experimental results using the parameters set in Section 5.4.3. We present the results on Amazon and Yelp datasets, respectively in Tables Table X and Table XI. For each

	$eval_{cha}$				$eval_{cp}$			
Feature Name	Prec.	Recall	F1	Accu.	Prec.	Recall	F1	Accu.
One Sequence Baselines								
OS-AvgSentLen	0.748	0.728	0.738	0.742	0.473	0.628	0.539	0.608
OS-AvgTokenLen	0.761	0.576	0.656	0.698	0.402	0.417	0.409	0.562
OS-Punctuations_Ratio	0.660	0.724	0.690	0.676	0.321	0.560	0.408	0.49
OS-Adj&Adv_Ratio	0.508	0.36	0.421	0.506	0.107	0.106	0.106	0.364
OS-Noun_Ratio	0.478	0.316	0.380	0.486	0.109	0.095	0.101	0.364
OS-Function_Word_Ratio	0.718	0.612	0.660	0.686	0.356	0.439	0.393	0.532
One Feature Baselines								
OF-Unigrams	0.950	0.688	0.798	0.826	0.839	0.661	0.739	0.786
OF-Bigrams	0.914	0.684	0.782	0.81	0.796	0.653	0.718	0.766
OF-Punctuations	0.875	0.76	0.813	0.826	0.658	0.704	0.680	0.732
OF-Function_Words	0.903	0.376	0.531	0.668	0.644	0.300	0.409	0.614
OF-POS_Unigrams	0.945	0.484	0.640	0.728	0.773	0.434	0.556	0.684
OF-POS_Bigrams	0.946	0.492	0.647	0.732	0.730	0.427	0.539	0.676
OF-Nouns	0.796	0.392	0.525	0.646	0.528	0.299	0.382	0.58
OF-Adj&Adv	0.869	0.32	0.467	0.636	0.684	0.270	0.387	0.602
OF-AvgSentLen	0.734	0.576	0.645	0.684	0.510	0.485	0.497	0.596
OF-AvgTokenLen	0.888	0.352	0.504	0.654	0.545	0.25	0.342	0.586
CHAD Variants								
IndV	0.989	0.36	0.527	0.678	0.945	0.349	0.510	0.67
CHAD-PFS	0.981	0.62	0.759	0.804	0.892	0.597	0.715	0.776
CHAD-F	Same as CHAD as no features are globally removed from $\boldsymbol{F}^{all}$.							
CHAD								
CHAD	0.954	0.752	**0.841**	**0.858**	0.847	0.729	**0.784**	**0.816**

TABLE XI. Results of CH detection on the Yelp accounts.

dataset we only list all 6 OS results, all 10 OF results, results of IndV, CHAD-PFS, CHAD-F and our proposed CHAD. Note that CHAD and CHAD-F produce the same results on the Yelp dataset as no features are removed from $\boldsymbol{F}^{all}$.

First, CHAD achieves the best results on both datasets. CHAD-F, which uses pivot-level feature selection, outperforms CHAD-PFS, which demonstrates the effectiveness of our pivot-

level feature selection over global feature pre-selection. Moreover, for the Amazon dataset (Table Table X), CHAD-F outperforms CHAD-PFS, and CHAD improves upon CHAD-F further, which show that filtering the input feature set $\boldsymbol{F}$ helps pivot-level feature selection. We also note that IndV performs poorly on both datasets. This is because it detects change points on individual similarity sequences for each pivot window, which are unreliable due to noise and fluctuations.

Second, the results of OS baselines are quite poor, for which there are two possible explanations. They rely on computing a single value as feature, which may not be sufficient in capturing the differences between users in CH accounts. Moreover, OS baselines construct only one sequence, which is potentially less reliable than multiple sequences in CHAD.

Third, the three best OF baselines are the same for both datasets, namely OF-unigrams, OF-bigrams and OF-punctuations. They are inferior to CHAD-PFS, CHAD-F and CHAD on the Amazon dataset, but are better than CHAD-PFS on Yelp. As noted in Section 5.4.1, Yelp is different from Amazon in that all its reviews are about hotels. It thus requires more fine-grained feature selection where global feature selection is insufficient.

Fourth, we found that two OF baselines, OF-nouns and OF-adjective&adverbs, give very different results on two datasets. Specifically, they yield significantly better results on Amazon than on Yelp. We believe the difference is caused by the nature of the two datasets. As the Amazon dataset contains multiple categories of products, nouns and adjectives&adverbs may be able to capture the differences between users in CH accounts better than those on the Yelp dataset where all the reviews are about hotels. Overall, however, nouns and adjec-

	$y=1$				$y=3$			
CHAD-PFS	0.117	0.341	0.175	0.492	0.475	0.677	0.558	0.656
CHAD-F	0.095	0.370	0.152	0.488	0.487	0.75	0.590	0.676
CHAD	0.127	0.455	0.199	0.502	0.485	0.761	0.592	0.676
	$y=5$				$y=7$			
CHAD-PFS	0.751	0.767	0.759	0.782	0.764	0.770	0.767	0.788
CHAD-F	0.729	0.817	0.770	0.792	0.775	0.826	0.8	0.814
CHAD	0.757	0.832	0.793	0.808	0.781	0.837	0.808	0.82

TABLE XII

Effect of varying y under $eval_{cp}$ on the Amazon accounts.

	$y=1$				$y=3$			
CHAD-PFS	0.139	0.188	0.16	0.538	0.689	0.534	0.602	0.712
CHAD-F	Same as CHAD							
CHAD	0.152	0.326	0.207	0.542	0.614	0.661	0.636	0.724
	$y=5$				$y=7$			
CHAD-PFS	0.892	0.597	0.715	0.776	0.917	0.604	0.728	0.784
CHAD-F	Same as CHAD							
CHAD	0.847	0.729	0.784	0.816	0.878	0.736	0.801	0.828

TABLE XIII

Effect of varying y under $eval_{cp}$ on the Yelp accounts.

tives&adverbs do not perform as well as the best-3 OF-baselines (OF-unigrams, OF-bigrams and OF-punctuations). This might be due to the fact that an average Amazon reviewer is also likely to post reviews on different categories of products, which creates variance in similarities.

Lastly, we notice that two OF baselines, OF-AvgSentLen and OF-AvgTokenLen, perform worse than their counterparts OS-AvgSentLen and OS-AvgTokenLen on both datasets. We

believe this is caused by the fact that OF features are normalized to the range of [0,1] as we described in Section 5.3.2, while OS features are not. Normalized features may have less distinguishable power.

In the end, we aim to investigate the effect of window size y (Section 5.4.1) on change-point evaluation ($eval_{cp}$) by varying $y = 1, 3, 5, 7$. We report the results for CHAD-PFS, CHAD-F and CHAD on both datasets in Table Table XII and Table Table XIII. Only results for $eval_{cp}$ are listed as those for $eval_{cha}$ are not affected by the value of y. Both both tables, we can see that as expected, the results improve as the value of y increases. CHAD performs the best compared to the other two regardless of the value of y. And CHAD-F, which uses pivot-level feature selection, outperforms CHAD-PFS with a single exception at $y = 1$ on Amazon.

CHAPTER 6

CONCLUSIONS

In this thesis, we have studied three research problems related to open classification and the problem of detecting changed-hands online review accounts. We started with a special case of open classification—text classification under negative covariate shift. Then we studied the general problem of open classification. We further proposed cumulative learning in the context of text classification, which allows unseen classes of data not only be detected, but also cumulatively added to the existing system without rebuilding the system from scratch. In the end, we proposed to detect changed-hands online review accounts from a linguistic perspective in the unsupervised setting. In all of the above problems, transforming text data to a similarity space or similarity sequences enabled us to detect different types of changes in the data. The contributions of this thesis are summarized as below:

- First, we studied the problem of text classification under negative covariate shift, which can be seen as a special case of open classification. The problem is prevalent when one aims to collect relevant posts/documents of a particular topic of interest from social media. We attempted to solve this problem by proposing a new learning technique, called center-based similarity space learning (CBS-L), which transforms document representation from the traditional n-gram feature space to a similarity space and learns in the similarity space. Our experimental results show that the proposed method CBS-L outperformed strong baselines by large margins.

- We then studied the general problem of open classification. In particular, we approached the problem from an open space risk management perspective by proposing an open space formulation and provided a solution by generalizing the proposed CBS learning technique. Our open space formulation greatly reduced the positively labeled area compared to previous works, which markedly reduced the open space risk. With extensive experiments across two public datasets, we demonstrated that the proposed solution is highly promising. One of the future directions could be to design a more robust solution that works better when the number of known classes is small.
- We further proposed cumulative learning, as a special form of supervised lifelong machine learning. Two unique challenges in cumulative learning were identified and presented, namely, the ability to perform open classification in order to detect data from unseen classes in the test set and the ability to selectively update the existing classification model without requiring rebuilding the whole system from scratch. We also proposed a cumulative learning strategy, which we believe is similar to human concept learning. It only requires updating part of the existing classification model whenever a new class of data arrives and needs to be covered by the classification model. As time goes by, the system keeps learning more and more in an efficient manner and becomes more and more knowledgeable. Through extensive experiments by building classifiers for up to 100 consecutive classes, and comparing with strong baselines, we demonstrated the effectiveness and efficiency of the proposed approach.

- Lastly, we explored the use of similarity-based approaches in detecting a type of change in social media accounts in an unsupervised fashion. In particular, we aimed to detect changed-hands online review accounts. The problem has presented two unique challenges due to the differences in intra-user and inter-user writing styles. We introduced a detection algorithm, CHAD, which determines if an account has changed hands and the possible change point by transforming a sequence of reviews in an account into sequences of similarity scores. In order to address our challenges, CHAD employs a pivot-level feature selection algorithm, which removes noisy features and their corresponding similarity sequences for each pivot window. Experiments on two synthetic datasets constructed using Amazon and Yelp review datasets demonstrated that CHAD is highly effective compared to a list of baselines.

APPENDICES

Appendix A : Copyrights

Association for Computational Linguistics

Copyright Transfer Agreement

Title of Work:

Social Media Text Classification under Negative Covariate Shift

Author(s):

Geli Fei, Bing Liu

Copyright to the above work (including, without limitation, the right to publish the work in whole or in part in any and all forms and media, now or hereafter known) is hereby transferred to the Association for Computational Linguistics (ACL), effective as of the date of this agreement, on the understanding that the work has been accepted for presentation at a meeting sponsored by the ACL and for publication in the proceedings of that meeting. However, each of the authors and the employers for whom the work was performed reserve all other rights, specifically including the following: (1) All proprietary rights other than copyright and publication rights transferred to ACL; (2) The right to publish in a journal or collection or to be used in future works of the author's own (such as articles or books) all or part of this work, provided that acknowledgment is given to the ACL and a full citation to its publication in the particular proceedings is included; (3) The right to make oral presentation of the material in any forum; (4) The right to make copies of the work for internal distribution within the author's organization and for external distribution as a preprint, reprint, technical report, or related class of document. In the case of a work prepared under a government contract, if the contract so requires, that government may reproduce all or portions of the article and may authorize others to do so, for official government purposes only.

By signing below, I confirm that all authors of the work have agreed to the above and that I am authorized to execute this transfer on their behalf.

08/14/2015

Signature(s) Date

Geli Fei

Name(s) (please print)

Your job title (if not one of the authors)

Name and address of your organization

Association for Computational Linguistics

Copyright Transfer Agreement

Title of Work:

Breaking the Closed World Assumption in Text Classification

Author(s):

Geli Fei; Bing Liu

By signing below, I confirm that all authors of the work have agreed to the above and that I am authorized to execute this transfer on their behalf.

Signature(s) Date

04/01/2016

Geli Fei

Name(s) (please print)

Your job title (if not one of the authors)

Name and address of your organization

ACM Author Rights

"Authors can reuse any portion of their own work in a new work of their own (and no fee is expected) as long as a citation and DOI pointer to the Version of Record in the ACM Digital Library are included.

Contributing complete papers to any edited collection of reprints for which the author is not the editor, requires permission and usually a republication fee.

Authors can include partial or complete papers of their own (and no fee is expected) in a dissertation as long as citations and DOI pointers to the Versions of Record in the ACM Digital Library are included. Authors can use any portion of their own work in presentations and in the classroom (and no fee is expected).

Commercially produced course-packs that are sold to students require permission and possibly a fee." [1]

---

[1] http://authors.acm.org/main.html

CITED LITERATURE

[Adams and MacKay, 2007] Ryan Prescott Adams and David JC MacKay. 2007. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*.

[Akoglu et al., 2013] Leman Akoglu, Rishi Chandy, and Christos Faloutsos. 2013. Opinion fraud detection in online reviews by network effects. *ICWSM*, 13:2–11.

[Alter et al., 2000] Orly Alter, Patrick O Brown, and David Botstein. 2000. Singular value decomposition for genome-wide expression data processing and modeling. *Proceedings of the National Academy of Sciences*, 97(18):10101–10106.

[Basseville et al., 1993] Michèle Basseville, Igor V Nikiforov, et al. 1993. *Detection of abrupt changes: theory and application*, volume 104. Prentice Hall Englewood Cliffs.

[Bickel and Scheffer, 2007] Steffen Bickel and Tobias Scheffer. 2007. Dirichlet-enhanced spam filtering based on biased samples. *Advances in neural information processing systems*, 19:161.

[Bickel et al., 2009] Steffen Bickel, Michael Brückner, and Tobias Scheffer. 2009. Discriminative learning under covariate shift. *Journal of Machine Learning Research*, 10(Sep):2137–2155.

[Blei et al., 2003] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

[Blum, 1998] Avrim Blum. 1998. On-line algorithms in machine learning. In *Online algorithms*, pages 306–325. Springer.

[Bravo et al., 2008] Cristian Bravo, Jose Luis Lobato, Richard Weber, and Gaston L'Huillier. 2008. A hybrid system for probability estimation in multiclass problems combining svms and neural networks. In *Hybrid Intelligent Systems, 2008. HIS'08. Eighth International Conference on*, pages 649–654. IEEE.

[Carlson et al., 2010] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, page 3.

[Castillo et al., 2007] Carlos Castillo, Debora Donato, Aristides Gionis, Vanessa Murdock, and Fabrizio Silvestri. 2007. Know your neighbors: Web spam detection using the web topology. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 423–430. ACM.

[Cauwenberghs and Poggio, 2000] Gert Cauwenberghs and Tomaso Poggio. 2000. Incremental and decremental support vector machine learning. In *NIPS*, volume 13.

[Cha, 2007] Sung-Hyuk Cha. 2007. Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(2):1.

[Chang and Lin, 2011] Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

[Chen and Gupta, 1997] Jie Chen and Arjun K Gupta. 1997. Testing and locating variance changepoints with application to stock prices. *Journal of the American Statistical association*, 92(438):739–747.

[Chen and Gupta, 1999] Jie Chen and AK Gupta. 1999. Change point analysis of a gaussian model. *Statistical Papers*, 40(3):323–333.

[Chen and Gupta, 2012] Jie Chen and Arjun K Gupta. 2012. *Parametric Statistical Change Point Analysis: With Applications to Genetics, Medicine, and Finance; 2nd ed.* Springer, Boston.

[Chen and Liu, 2014a] Zhiyuan Chen and Bing Liu. 2014a. Topic modeling using topics from many domains, lifelong learning and big data.

[Chen and Liu, 2014b] Zhiyuan Chen and Bing Liu. 2014b. Topic modeling using topics from many domains, lifelong learning and big data. In Tony Jebara and Eric P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 703–711. JMLR Workshop and Conference Proceedings.

[Chen et al., 2015] Zhiyuan Chen, Nianzu Ma, and Bing Liu. 2015. Lifelong learning for sentiment classification. *Volume 2: Short Papers*, page 750.

[Chirita et al., 2005] Paul-Alexandru Chirita, Jörg Diederich, and Wolfgang Nejdl. 2005. Mailrank: using ranking for spam detection. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 373–380. ACM.

[Colas and Brazdil, 2006] Fabrice Colas and Pavel Brazdil. 2006. Comparison of svm and some older classification algorithms in text classification tasks. In *IFIP International Conference on Artificial Intelligence in Theory and Practice*, pages 169–178. Springer.

[Crammer et al., 2006] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar):551–585.

[Dalvi et al., 2013] Bhavana Dalvi, William W Cohen, and Jamie Callan. 2013. Exploratory learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 128–143. Springer.

[Denis, 1998] François Denis. 1998. Pac learning from positive statistical queries. In *International Conference on Algorithmic Learning Theory*, pages 112–126. Springer.

[Duan and Keerthi, 2005] Kai-Bo Duan and S Sathiya Keerthi. 2005. Which is the best multiclass svm method? an empirical study. In *International Workshop on Multiple Classifier Systems*, pages 278–285. Springer.

[Dudík et al., 2005] Miroslav Dudík, Steven J Phillips, and Robert E Schapire. 2005. Correcting sample selection bias in maximum entropy density estimation. In *Advances in neural information processing systems*, pages 323–330.

[Elkan and Noto, 2008] Charles Elkan and Keith Noto. 2008. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–220. ACM.

[Fei and Liu, 2015] Geli Fei and Bing Liu. 2015. Social media text classification under negative covariate shift. In *EMNLP*, pages 2347–2356.

[Fei and Liu, 2016] Geli Fei and Bing Liu. 2016. Breaking the closed world assumption in text classification. In *Proceedings of NAACL-HLT*, pages 506–514.

[Fei et al., 2013] Geli Fei, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013. Exploiting burstiness in reviews for review spammer detection. *ICWSM*, 13:175–184.

[Fei et al., 2016] Geli Fei, Shuai Wang, and Bing Liu. 2016. Learning cumulatively to become more knowledgeable. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1565–1574. ACM.

[Fei et al., 2017] Geli Fei, Shuai Wang, Leman Akoglu, and Bing Liu. 2017. Detecting changed-hands online review accounts. In *ACL under review*. Association for Computational Linguistics.

[Feng et al., 2012a] Song Feng, Ritwik Banerjee, and Yejin Choi. 2012a. Syntactic stylometry for deception detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 171–175. Association for Computational Linguistics.

[Feng et al., 2012b] Song Feng, Longfei Xing, Anupam Gogar, and Yejin Choi. 2012b. Distributional footprints of deceptive product reviews. *ICWSM*, 12:98–105.

[Fink et al., 2006] Michael Fink, Shai Shalev-Shwartz, Yoram Singer, and Shimon Ullman. 2006. Online multiclass learning by interclass hypothesis sharing. In *Proceedings of the 23rd international conference on Machine learning*, pages 313–320. ACM.

[Fumera and Roli, 2002] Giorgio Fumera and Fabio Roli. 2002. Support vector machines with embedded reject option. In *Pattern Recognition with Support Vector Machines*, pages 68–82. Springer.

[Gulordava and Baroni, 2011] Kristina Gulordava and Marco Baroni. 2011. A distributional similarity approach to the detection of semantic change in the google books ngram corpus. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 67–71. Association for Computational Linguistics.

[Hai et al., 2016] Zhen Hai, Peilin Zhao, Peng Cheng, Peng Yang, Xiao-Li Li, Guangxia Li, and Ant Financial. 2016. Deceptive review spam detection via exploiting task relatedness and unlabeled data. In *EMNLP*.

[Hancock et al., 2007] Jeffrey T Hancock, Lauren E Curry, Saurabh Goorha, and Michael Woodworth. 2007. On lying and being lied to: A linguistic analysis of deception in computer-mediated communication. *Discourse Processes*, 45(1):1–23.

[He et al., 2004] Xiaofei He, Deng Cai, Haifeng Liu, and Wei-Ying Ma. 2004. Locality preserving indexing for document representation. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 96–103. ACM.

[Heckman, 1977] James J. Heckman. 1977. Sample selection bias as a specification error (with an application to the estimation of labor supply functions). Working Paper 172, National Bureau of Economic Research, March.

[Huang et al., 2006a] Jiayuan Huang, Arthur Gretton, Karsten M Borgwardt, Bernhard Schölkopf, and Alex J Smola. 2006a. Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems*, pages 601–608.

[Huang et al., 2006b] Tzu-Kuo Huang, Ruby C Weng, and Chih-Jen Lin. 2006b. Generalized bradley-terry models and multi-class probability estimates. *Journal of Machine Learning Research*, 7(Jan):85–115.

[Jain et al., 2014] Lalit P Jain, Walter J Scheirer, and Terrance E Boult. 2014. Multi-class open set recognition using probability of inclusion. In *European Conference on Computer Vision*, pages 393–409. Springer.

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

[Jindal and Liu, 2008] Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 219–230. ACM.

[Jindal et al., 2010] Nitin Jindal, Bing Liu, and Ee-Peng Lim. 2010. Finding unusual review patterns using unexpected rules. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1549–1552. ACM.

[Joachims, 1996] Thorsten Joachims. 1996. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report, DTIC Document.

[Joachims, 1998] Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

[Kakade et al., 2008] Sham M Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. 2008. Efficient bandit algorithms for online multiclass prediction. In *Proceedings of the 25th international conference on Machine learning*, pages 440–447. ACM.

[KC and Mukherjee, 2016] Santosh KC and Arjun Mukherjee. 2016. On the temporal dynamics of opinion spamming: Case studies on yelp. In *Proceedings of the 25th International Conference on World Wide Web*, pages 369–379. International World Wide Web Conferences Steering Committee.

[Khan and Madden, 2013] Shehroz S. Khan and Michael G. Madden. 2013. One-class classification: Taxonomy of study and review of techniques. *CoRR*, abs/1312.0049.

[Killick and Eckley, 2014] Rebecca Killick and Idris A. Eckley. 2014. changepoint: An R package for changepoint analysis. *Journal of Statistical Software*, 58(3):1–19.

[Kim et al., 2014] Yoon Kim, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. 2014. Temporal analysis of language through neural language models. *arXiv preprint arXiv:1405.3515*.

[Kotz and Nadarajah, 2000] Samuel Kotz and Saralees Nadarajah. 2000. *Extreme value distributions: theory and applications*. World Scientific.

[Kulkarni et al., 2015] Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2015. Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*, pages 625–635. ACM.

[Kwok, 1999] JT-Y Kwok. 1999. Moderating the outputs of support vector machine classifiers. *IEEE Transactions on Neural Networks*, 10(5):1018–1031.

[Le and Mikolov, 2014] Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196.

[Lebanon, 2006] Guy Lebanon. 2006. Sequential document representations and simplicial curves. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'06, pages 273–280, Arlington, Virginia, United States. AUAI Press.

[Lee and Liu, 2003] Wee Sun Lee and Bing Liu. 2003. Learning with positive and unlabeled examples using weighted logistic regression. In *ICML*, volume 3, pages 448–455.

[Li et al., 2010] Xiao-Li Li, Bing Liu, and See-Kiong Ng. 2010. Negative training data can be harmful to text classification. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 218–228. Association for Computational Linguistics.

[Li et al., 2011] Fangtao Li, Minlie Huang, Yi Yang, and Xiaoyan Zhu. 2011. Learning to identify review spam. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*.

[Li et al., 2014a] Huayi Li, Zhiyuan Chen, Bing Liu, Xiaokai Wei, and Jidong Shao. 2014a. Spotting fake reviews via collective positive-unlabeled learning. In *2014 IEEE International Conference on Data Mining*, pages 899–904. IEEE.

[Li et al., 2014b] Jiwei Li, Myle Ott, Claire Cardie, and Eduard H Hovy. 2014b. Towards a general rule for identifying deceptive opinion spam. In *ACL (1)*, pages 1566–1576. Citeseer.

[Li et al., 2015] Huayi Li, Zhiyuan Chen, Arjun Mukherjee, Bing Liu, and Jidong Shao. 2015. Analyzing and detecting opinion spam on a large-scale dataset via temporal and spatial patterns. In *Proceedings of The 9th International AAAI Conference on Web and Social Media (ICWSM-15), Oxford, UK*, pages 26–29.

[Lim et al., 2010] Ee-Peng Lim, Viet-An Nguyen, Nitin Jindal, Bing Liu, and Hady Wirawan Lauw. 2010. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 939–948. ACM.

[Liu et al., 2003] Bing Liu, Yang Dai, Xiaoli Li, Wee Sun Lee, and Philip S Yu. 2003. Building text classifiers using positive and unlabeled examples. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 179–186. IEEE.

[Manevitz and Yousef, 2001] Larry M Manevitz and Malik Yousef. 2001. One-class svms for document classification. *Journal of Machine Learning Research*, 2(Dec):139–154.

[Manning et al., 2008] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. 2008. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge.

[Mitra et al., 2014] Sunny Mitra, Ritwik Mitra, Martin Riedl, Chris Biemann, Animesh Mukherjee, and Pawan Goyal. 2014. That's sick dude!: Automatic identification of word sense change across different timescales. *arXiv preprint arXiv:1405.4392*.

[Mukherjee et al., 2012] Arjun Mukherjee, Bing Liu, and Natalie Glance. 2012. Spotting fake reviewer groups in consumer reviews. In *Proceedings of the 21st international conference on World Wide Web*, pages 191–200. ACM.

[Mukherjee et al., 2013a] Arjun Mukherjee, Abhinav Kumar, Bing Liu, Junhui Wang, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013a. Spotting opinion spammers using behavioral footprints. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 632–640. ACM.

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

[Newman et al., 2003] Matthew L Newman, James W Pennebaker, Diane S Berry, and Jane M Richards. 2003. Lying words: Predicting deception from linguistic styles. *Personality and social psychology bulletin*, 29(5):665–675.

[Ott et al., 2011] Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 309–319. Association for Computational Linguistics.

[Ott et al., 2012] Myle Ott, Claire Cardie, and Jeff Hancock. 2012. Estimating the prevalence of deception in online review communities. In *Proceedings of the 21st international conference on World Wide Web*, pages 201–210. ACM.

[Pan and Yang, 2010] Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

[Pentina and Lampert, 2014] Anastasia Pentina and Christoph H Lampert. 2014. A pac-bayesian bound for lifelong learning. In *ICML*, pages 991–999.

[Pérez-Rosas and Mihalcea, 2015] Verónica Pérez-Rosas and Rada Mihalcea. 2015. Experiments in open domain deception detection. In *EMNLP*, pages 1120–1125.

[Pérez-Rosas et al., 2015] Verónica Pérez-Rosas, Mohamed Abouelenien, Rada Mihalcea, Yao Xiao, CJ Linton, and Mihai Burzo. 2015. Verbal and nonverbal clues for real-life deception detection. In *EMNLP*, pages 2336–2346.

[Platt, 1999] John C. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74.

[Qian and Liu, 2013] Tieyun Qian and Bing Liu. 2013. Identifying multiple userids of the same author. In *EMNLP*, pages 1124–1135.

[Radev et al., 2000] Dragomir R Radev, Hongyan Jing, and Malgorzata Budzikowska. 2000. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *Proceedings of the 2000 NAACL-ANLP Workshop on Automatic summarization*, pages 21–30. Association for Computational Linguistics.

[Ranzato and Szummer, 2008] Marc'Aurelio Ranzato and Martin Szummer. 2008. Semi-supervised learning of compact document representations with deep networks. In *Proceedings of the 25th international conference on Machine learning*, pages 792–799. ACM.

[Rayana and Akoglu, 2015] Shebuti Rayana and Leman Akoglu. 2015. Collective opinion spam detection: Bridging review networks and metadata. In *KDD*, pages 985–994. ACM.

[Ren et al., 2014] Yafeng Ren, Donghong Ji, and Hongbin Zhang. 2014. Positive unlabeled learning for deceptive reviews detection. In *EMNLP*, pages 488–498.

[Rocchio, 1971] Joseph John Rocchio. 1971. Relevance feedback in information retrieval.

[Scheirer et al., 2011] Walter J Scheirer, Anderson Rocha, Ross J Micheals, and Terrance E Boult. 2011. Meta-recognition: The theory and practice of recognition score analysis. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1689–1695.

[Scheirer et al., 2013] Walter Scheirer, Anderson Rocha, Archana Sapkota, and Terrance Boult. 2013. Toward open set recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(7):1757–1772, July.

[Scheirer et al., 2014] Walter J Scheirer, Lalit P Jain, and Terrance E Boult. 2014. Probability models for open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 36(11):2317–2324.

[Schölkopf et al., 2000] B. Schölkopf, RC. Williamson, AJ. Smola, J. Shawe-Taylor, and JC. Platt. 2000. Support vector method for novelty detection. In *Advances in Neural Information Processing Systems 12*, pages 582–588, Cambridge, MA, USA, June. Max-Planck-Gesellschaft, MIT Press.

[Schwarz and others, 1978] Gideon Schwarz et al. 1978. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464.

[Shackel, 2007] Nicholas Shackel. 2007. Bertrands paradox and the principle of indifference. *Philosophy of Science*, 74(2):150–175.

[Shimodaira, 2000] Hidetoshi Shimodaira. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244.

[Silver et al., 2013] Daniel L Silver, Qiang Yang, and Lianghao Li. 2013. Lifelong machine learning systems: Beyond learning algorithms. In *AAAI Spring Symposium: Lifelong Machine Learning*, pages 49–55. Citeseer.

[Spirin and Han, 2012] Nikita Spirin and Jiawei Han. 2012. Survey on web spam detection: principles and algorithms. *ACM SIGKDD Explorations Newsletter*, 13(2):50–64.

[Stamatatos, 2009] Efstathios Stamatatos. 2009. A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology*, 60(3):538–556.

[Sugiyama et al., 2008] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul V Buenau, and Motoaki Kawanabe. 2008. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in neural information processing systems*, pages 1433–1440.

[Tax and Duin, 1999a] David M. J. Tax and Robert P. W. Duin. 1999a. Data domain description using support vectors. In *Proceedings of the European Symposium on Artificial Neural Networks*, pages 251–256.

[Tax and Duin, 1999b] David M.J Tax and Robert P.W Duin. 1999b. Support vector domain description. *Pattern Recognition Letters*, 20(1113):1191 – 1199.

[Taylor, 2000] Wayne A Taylor. 2000. Change-point analysis: a powerful new tool for detecting changes. *preprint, available as http://www. variation. com/cpa/tech/changepoint. html*.

[Thrun and Mitchell, 1995] Sebastian Thrun and Tom M Mitchell. 1995. Lifelong robot learning. In *The biology and technology of intelligent autonomous agents*, pages 165–196. Springer.

[Thrun, 1996] Sebastian Thrun. 1996. Is learning the n-th thing any easier than learning the first? *Advances in neural information processing systems*, pages 640–646.

[Tsuboi et al., 2009] Yuta Tsuboi, Hisashi Kashima, Shohei Hido, Steffen Bickel, and Masashi Sugiyama. 2009. Direct density ratio estimation for large-scale covariate shift adaptation. *Information and Media Technologies*, 4(2):529–546.

[Wang and Domeniconi, 2008] Pu Wang and Carlotta Domeniconi. 2008. Building semantic kernels for text classification using wikipedia. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 713–721. ACM.

[Wang et al., 2011] Guan Wang, Sihong Xie, Bing Liu, and S Yu Philip. 2011. Review graph based online store review spammer detection. In *2011 IEEE 11th International Conference on Data Mining*, pages 1242–1247. IEEE.

[Wang et al., 2016] Xuepeng Wang, Kang Liu, Shizhu He, and Jun Zhao. 2016. Learning to represent review with tensor decomposition for spam detection. In *EMNLP*.

[Wijaya and Yeniterzi, 2011] Derry Tanti Wijaya and Reyyan Yeniterzi. 2011. Understanding semantic change of words over centuries. In *Proceedings of the 2011 international workshop on DETecting and Exploiting Cultural diversiTy on the social web*, pages 35–40. ACM.

[Xiao et al., 2014] Tianjun Xiao, Jiaxing Zhang, Kuiyuan Yang, Yuxin Peng, and Zheng Zhang. 2014. Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 177–186. ACM.

[Xie et al., 2012] Sihong Xie, Guan Wang, Shuyang Lin, and Philip S Yu. 2012. Review spam detection via temporal pattern discovery. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 823–831. ACM.

[Xu and Zhang, 2015] Chang Xu and Jie Zhang. 2015. Combating product review spam campaigns via multiple heterogeneous pairwise features. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 172–180. SIAM.

[Xu et al., 2013] Chang Xu, Jie Zhang, Kuiyu Chang, and Chong Long. 2013. Uncovering collusive spammers in chinese review websites. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 979–988. ACM.

[Yang and Pedersen, 1997] Yiming Yang and Jan O Pedersen. 1997. A comparative study on feature selection in text categorization. In *ICML*, volume 97, pages 412–420.

[Ye and Akoglu, 2015] Junting Ye and Leman Akoglu. 2015. Discovering opinion spammer groups by network footprints. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 267–282. Springer.

[Ye et al., 2016] Junting Ye, Santhosh Kumar, and Leman Akoglu. 2016. Temporal opinion spam detection by multivariate indicative signals. In *ICWSM*, pages 743–746.

[Yu et al., 2002] Hwanjo Yu, Jiawei Han, and Kevin Chen-Chuan Chang. 2002. Pebl: positive example based learning for web page classification using svm. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 239–248. ACM.

[Zadrozny and Elkan, 2002] Bianca Zadrozny and Charles Elkan. 2002. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 694–699. ACM.

[Zadrozny, 2004] Bianca Zadrozny. 2004. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 114–, New York, NY, USA. ACM.

[Zhao et al., 2011] Peilin Zhao, Steven CH Hoi, and Rong Jin. 2011. Double updating online learning. *Journal of Machine Learning Research*, 12(May):1587–1615.

VITA

Geli Fei

Education

Ph.D., Computer Science, University of Illinois at Chicago, Chicago, Illinois, 2017.

B.S., Software Engineering, Harbin Institute of Technology, China, 2011.

Working Experience

Software Engineering Intern at Google, Mountain View, CA, USA. May 2016 - August 2016.

Software Engineering Intern at Google, Mountain View, CA, USA. May 2015 - August 2015.

Research Intern at IBM Research-Almaden, Almaden, CA, USA. May 2014 - August 2014.

Summer Intern at @WalmartLabs, Mountain View, CA. May 2013 - August 2013.

Selected Publications

- Huayi Li, Geli Fei, Shuai Wang, Bing Liu, Weixiang Shao, Arjun Mukherjee, Jidong Shao. "Bimodal Distribution and Co-Bursting in Review Spam Detection". *Long Paper*. To Appear In Proceedings of **WWW 2017**.
- Geli Fei, Shuai Wang, Bing Liu "Learning Cumulatively to Become More Knowledgeable". *Full Paper*. In Proceedings of **KDD 2016**.
- Shuai Wang, Zhiyuan Chen, Geli Fei, Bing Liu, Sherry Emery. "Targeted Topic Modeling for Focused Analysis". *Full Paper*. In Proceedings of **KDD 2016**.

- Geli Fei, Bing Liu. "Open Text Classification by Reducing Open Space Risk". *Full Paper*. In Proceedings of **NAACL 2016**.
- Geli Fei, Zhiyuan Chen, Arjun Mukherjee, Bing Liu. "Discovering Correspondence of Sentiment Words and Aspects". *Full Paper*. In Proceedings of **CICLING 2016**.
- Jalal Mahmud, Geli Fei, Anbang Xu, Aditya Pal, Michelle Zhou. "Predicting Attitude and Actions of Twitter Users". *Short Paper*. In Proceedings of **IUI 2016**.
- Geli Fei, Huayi Li, Bing Liu. "Opinion spam detection in Social Networks". In Handbook of Sentiment Analysis in Social Network, **Elsevier 2016**.
- Geli Fei, Bing Liu. "Social Media Text Classification under Negative Covariate Shift". *Long Paper*. In Proceedings of **EMNLP 2015**.
- Geli Fei, Zhiyuan Chen, Bing Liu. "Review Topic Discovery with Phrases using the Pólya Urn Model". *Oral Paper*. In Proceedings of **COLING 2014**.
- Geli Fei, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, Riddhiman Ghosh. "Exploiting Burstiness in Reviews for Review Spammer Detection". *Long Paper*. In Proceedings of **ICWSM 2013**.
- Geli Fei, Bing Liu. "A Dictionary-Based Approach to Identifying Aspects Implied by Adjectives for Opinion Mining". *Poster Paper*. In Proceedings of **COLING 2012**.