A Multi-label Text Classification Framework: Using Supervised and Unsupervised Feature Selection Strategy

Long Ma

Follow this and additional works at: http://scholarworks.gsu.edu/cs_diss

Recommended Citation
Dissertation, Georgia State University, 2017.
http://scholarworks.gsu.edu/cs_diss/123
A MULTI-LABEL TEXT CLASSIFICATION FRAMEWORK: USING SUPERVISED AND UNSUPERVISED FEATURE SELECTION STRATEGY

by

LONG MA

Under the Direction of Yanqing Zhang, PhD

ABSTRACT

Text classification, the task of metadata to documents, requires significant time and effort when performed by humans. Moreover, with online-generated content explosively growing, it becomes a challenge for manually annotating with large scale and unstructured data. Currently, lots of state-or-art text mining methods have been applied to classification process, many of them based on the key word extraction. However, when using these key words as features in classification task, it is common that feature dimension is huge. In addition, how to select key words from tons of documents as features in classification task is also a challenge. Especially when using tradition machine learning algorithm in the large data set, the computation cost would be high. In addition, almost 80% of real data is unstructured and non-labeled. The advanced
supervised feature selection methods cannot be used directly in selecting entities from massive of data. Usually, extracting features from unlabeled data for classification tasks, statistical strategies have been utilized to discover key features. However, we propose a nova method to extract important features effectively before feeding them into the classification assignment. There is another challenge in the text classification is the multi-label problem, the assignment of multiple non-exclusive labels to the documents. This problem makes text classification more complicated when compared with single label classification. Considering above issues, we develop a framework for extracting and eliminating data dimensionality, solving the multi-label problem on labeled and unlabeled data set. To reduce data dimension, we provide 1) a hybrid feature selection method that extracts meaningful features according to the importance of each feature. 2) we apply the Word2Vec to represent each document with a lower feature dimension when doing the document categorization for the big data set. 3) An unsupervised approach to extract features from real online-generated data for text classification and prediction. On the other hand, to solve the multi-label classification task, we design a new Multi-Instance Multi-Label (MIML) algorithm in the proposed framework.

INDEX WORDS: Multi-label Text Classification, Feature Selection, Word2Vec, Natural Language Processing, Depression Symptoms, Social Medias
A MULTI-LABEL TEXT CLASSIFICATION FRAMEWORK: USING SUPERVISED AND UNSUPERVISED FEATURE SELECTION STRATEGY

by

LONG MA

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2017
A MULTI-LABEL TEXT CLASSIFICATION FRAMEWORK: USING SUPERVISED AND UNSUPERVISED FEATURE SELECTION STRATEGY

by

LONG MA

Committee Chair: Yanqing Zhang

Committee: Raj Sunderraman

Zhipeng Cai

Xin Qi

Electronic Version Approved:

Office of Graduate Studies
College of Arts and Sciences
Georgia State University
August 2017
DEDICATION

To my beloved wife, Ruohan and my parents for their love, support, and encouragement over the years.
ACKNOWLEDGEMENTS

I first would like to express my special appreciation and thanks to my advisor Professor Dr. Yanqing Zhang, you have been a tremendous mentor for me. I would like to thank you for encouraging my research and for allowing me to grow as a research scientist. Your advice on both research as well as on my career have been invaluable.

I would also like to thank my committee members, Professor Dr. Raj Sunderraman, Professor Dr. Zhipeng Cai, Professor Dr. Xin Qi for serving as my committee members even at hardship. I also want to thank you for letting my defense be an enjoyable moment, and for your brilliant comments and suggestions, thanks to you.

Last, I would like to acknowledge the continued financial support from the Computer Science Department, the Brain and Behavior (B&B) fellowship at Georgia State University.
# TABLE OF CONTENTS

ACKNOWLEDGEMENTS ........................................................................................................... v

LIST OF TABLES ....................................................................................................................... x

LIST OF FIGURES ..................................................................................................................... xi

1 INTRODUCTION ..................................................................................................................... 1

1.1 Background.......................................................................................................................... 1

1.2 Challenges for Multi-label Text Classification Framework ................................................. 5

1.3 Problem Statement............................................................................................................. 7

1.4 Organization....................................................................................................................... 9

2 RELATED WORK ..................................................................................................................... 9

2.1 MIML Algorithm .................................................................................................................. 9

2.2 Reduce Feature Dimension ............................................................................................... 10

2.2.1 Feature Reduction .......................................................................................................... 11

2.2.2 Feature Selection ............................................................................................................ 11

2.3 Artificial Neural Network .................................................................................................. 13

2.4 Depression Diagnosis System .......................................................................................... 15

3 MULTI-LABEL TEXT CLASSIFICATION FRAMEWORK .................................................. 17

3.1 Introduction ....................................................................................................................... 17

3.2 Overview of Framework ................................................................................................... 19
MULTI-LABEL TEXT CLASSIFICATION

4.1 Overview of Framework

4.2 Bagging Schemes Overview

4.3 Multi-Label Classification and LinearSVC

4.4 Bagging Schemes

4.4.2 Fixed Bag Size or Bag Number Schemes

4.4.3 Resampling Schemes

4.5 Feature Reduction

4.6 Results

4.6.2 Count Based Feature Selection

4.6.3 Feature Reduction with Word Stemming

4.6.4 Principal Component Analysis

4.6.5 Performance on All Label Dimensions

4.7 Conclusion

HYBRID FEATURE SELECTION

5.1 Introduction

5.2 Single Feature Selection Methods

5.3 Common Features Selected by Multiple Methods

5.4 Overall Performance
### 5.5 Artificial Data Evaluation

5.5 Artificial Data Evaluation

6 WORD2VEC PROCESS BIG DATA

6.1 Overview of Framework

6.2 Word2Vec Structure

6.3 Word Similarity

6.4 Word Clustering

6.5 Bag of Concept (BoC)

6.6 Multi-Class Classification

6.7 Conclusion

7 AI-BASED DEPRESSION DIAGNOSIS SYSTEM

7.1 Data Integration

7.1.2 *Data Preprocessing*

7.2 Entity Extraction

7.2.1 *Entity Co-occurrence*

7.2.2 *Entity Similarity*

7.2.3 *Rapid Automatic Keyword Extraction Algorithm*

7.3 Automatic Extract Keyword for Specific Terms

7.4 Depression Semantic Graph

7.5 Conclusion
LIST OF TABLES

Table 4.1 Summary of Data Based and Fixed Bagging Schemes .................................................. 27
Table 4.2 Performance of Bagging Schemes on the Original Data ............................................... 31
Table 4.3 Performance of Count Based Feature Selection .......................................................... 31
Table 4.4 Performance of Stemmed Data ..................................................................................... 33
Table 4.5 Performance on PCA Features ...................................................................................... 34
Table 4.6 Performance on Entire Label Dimension ...................................................................... 36
Table 4.7 PCA Performance on Eight Label Dimension .................................................................. 36
Table 5.1 The Number of Features Selected by Combination of RFE, SKB .................................. 42
Table 5.2 The Number of Features Selected by Combination of RFE, SKB and RF ...................... 43
Table 5.3 Feature Selection Performance on Eight Label Dimension ........................................... 45
Table 5.4 F1-Micro Scores with Wining Methods .......................................................................... 47
Table 5.5 Multi-label Classification on Genbase Dataset .............................................................. 48
Table 6.1 Similar Words ................................................................................................................ 54
Table 6.2 Cluster Contents ............................................................................................................. 56
Table 6.3 Multi-label Text Classification Performance ................................................................. 57
Table 7.1 Data Preprocessing ....................................................................................................... 61
Table 7.2 Word Counts .................................................................................................................. 62
Table 7.3 RAKE scores for single entity ....................................................................................... 67
# LIST OF FIGURES

Figure 1.1 MIML is in Learning Single-Label Examples Involving Complicated High-level Concepts [10] ........................................................................................................................................... 3

Figure 2.1 [10] Four Different Learning Framework ............................................................................................................. 10

Figure 2.2 Structure of NNLM .............................................................................................................................................. 15

Figure 3.1 Overview of Proposed Framework ......................................................................................................................... 19

Figure 4.1 Classifier Performance Decreases as the Number of Principal Components Increases ........................................................................................................................................... 35

Figure 5.1 Feature Selection Methods on Eight Label Dimensions .......................................................................................... 40

Figure 5.2 Feature Selection Methods Using RF ......................................................................................................................... 41

Figure 5.3 Co-occurrence Features Among Eight Label Dimensions .......................................................................................... 43

Figure 6.1 Working Process ......................................................................................................................................................... 51

Figure 6.2 Structures of CBOW and Skip-gram ........................................................................................................................ 52

Figure 7.1 Entity Co-occurrence counts that associate with the term “depression” ........................................................................ 64

Figure 7.2 Similar words related to "depression" ......................................................................................................................... 65

Figure 7.3 Hierarchical Structure of Depression and its Symptoms ............................................................................................. 66

Figure 7.4 Semantic graph with the center vertex “depression” and the weights represented as the importance score between two vertices ........................................................................................................................................... 70
1 INTRODUCTION

1.1 Background

Text classification and annotation is a task of assigning classes or labels to free text document [1,2]. It provides kind of view of document collections and has been used in many real applications. For example, each news can be categorized into different categories, such as political issues, sports, business and so on. The text classification or annotation gives business company insights for making their business decision more reliable. On the other hand, through viewing the text categories, it is convenient and helpful for people to choose their interests. Another interesting application of text classification is spam detection, which classified our emails into two categories of spam and normal emails. Text mining encompasses many different methods for organizing, searching, and annotating text. These ranges from methods of keyword searching, such as that used by large corporate search engines such as Google and Bing; textual information extraction, without machine learning (such as used by the ContentMine project; contentmine.org); to methods making heavy use of machine learning, such as the enterprise level solutions offered by the SAS Institute, Inc., to its clients. Standard machine learning algorithms used for text mining applications include: naïve Bayes classification, support vector machines, $K$-nearest neighbors [80], decision trees, artificial neural networks, linear discriminant analysis and many others, as well as algorithms developed specifically for text mining applications [81]. Overviews of text mining can be found in [2-4] among many other sources. In our framework, the research study will focus on the text classification, it is one type of the data mining task.

Many text mining procedures focus on extracting keywords from a text corpus and using them for proceeding mining tasks. In fact, there is not a general vocabulary that can guide us which words in the corpus are important; it is common that specific words that are very important to one
classification problem while useless for another. Often only stop words and punctuations are removed in preprocessing, possibly with or without some additional word stemming, and all the resulting text is used as features for the classification task. As a result, the number of features can be extremely big. This issue may affect the classification performance if the number of training instances is relatively small, as it is in one of our cases.

In text annotation, single label classification is a common and not too hard topic which the goal is to learn from a set of examples, each associated with a class label or category. Single label task is similar to the multi-class classification, which assigning only one category to each instance or data sample. However, nowadays, multi-label classification approaches are increasingly required by real applications, such as gene classification and article annotation. People explore many algorithms to improve multi-label classification, a common method is to convert a multi-label problem into single label task. A multi-instance multi-label (MIML) algorithm is used for supervised classification. The method takes “bags” of training instances, where each bag is associated with a collection of labels. The labels refer to the parts of the training instances [6]. For example, in the image classification domain, the bags are collections of images (training instances), the features used for classification are the pixels or collections of pixels appearing in the images, and certain collections of pixels make up the objects present in the images. The classifier learns to associate collections of pixels (representing objects) with labels which are the names of the objects. Note that the objects themselves enter into the classification explicitly only as labels and implicitly as collections of pixels. In Figure 1.1, an “Africa” image can be viewed of sub-subjects of African concepts, e.g. elephant and grassland. Thus, we can train “Africa” image with assigning the “African” labels to the bag of pixels. MIML algorithm has performed well in classifying complex objects such as images, texts, or the combination of them. In this case, text
documents are instances (images), and the words in the documents are the features (objects), and labels are names for collections of features; that is the concepts that are referred to by the words in the document [7-10]. Our theoretical approach is to exploit this analogy between images and texts; pixels and words; and objects and concepts.

(a) *Africa* is a complicated high-level concept

(b) The concept *Africa* may become easier to learn through exploiting some sub-concepts

*Figure 1.1 MIML is in Learning Single-Label Examples Involving Complicated High-level Concepts [10]*

A lot of researches have been proposed and developed on the algorithm perspective, yet more and more companies require employees to collect and process data in past decades. The good quality of data is more necessary than the powerfulness of algorithm is accepted by most of researchers and companies. In this big data era, data is everywhere. For instance, textual data, semi-structured data, graph data and streaming data. Many of these kinds of data are public, but others are not. The source of public data, e.g. social medias, which produces billions of data every
day, even every minute. They are public, free and innumerous, yet mining data on social media is difficult due to its complexity. The private data source, e.g. clinical data, is more valuable and meaningful. While it is more expensive than the public data and quantity is far less than the online data. An important research topic in the data science field is the data properties. To utilize all kinds of data in our research purpose, e.g. propose a new text classification algorithm, we usually work on the labeled data for evaluating algorithm performance. However, due to our framework could be built for a variety of data types, our research is not restricted to the labeled data. There are a few unsupervised or semi-supervised algorithms have been used in the specific tasks, such as $K$-means and its dependencies are good for entities clustering. The other difficulty in unlabeled data mining is the feature selection issue. It is a challenge to define which feature is a key candidate feature among countless of data that depends on its statistical distribution alone. We cannot select features rely upon supervised learning performance either, this task requires us develop an ensemble method to define key entities.

Text mining and machine learning require big enough data for information retrieval or classifier training. Recently, the big data attracts more and more attention from many fields, such science and economics. The big data even has been changing the way people working and thinking [45], it constructs the world in which business and computer science put their efforts to realize the value of data. In business, a company can find customers’ actual interests through integrating and analyzing the big data to improve the sale skills. In computer science, we have an advantage of working with the big data to training the machine learning model.

Big data is a huge data set with a complex structure that has been generated in various fields such as biology, medicine, social networks, and humanities. Training large-scale data set is a time-consuming task, it usually takes couple hours, days even weeks to build a model. After the
model is built, the classifier also needs to be re-training again in a specific period. It is difficult to handle such a big data set using traditional data processing tools. Many data intelligence applications have highly competitive performance when training model from a large amount of data. While the big data also makes learning model more expensive due to its data size and complexity. In order to reduce the data dimension, we use the machine learning and statistical learning strategies to achieve this goal. Some supervised learning algorithm, like Random Forest (RF) can be used individually to select important features. Additionally, we are looking for a powerful approach to lower the data dimension before feeding them into the text mining and machine learning tasks.

There is another method to decrease the computing cost of big data is to apply the parallel computing tools into that. Currently, Hadoop and Sparks [54,55] are open source big data computing frameworks that have been adopted by many enterprises, such as Yahoo, Baidu. Spark provides a framework that is used for cluster computing for many kinds of data type and file format. In addition, it supports HDFS which is a Hadoop stored file type. By parallel computing, the driver node assign tasks into cluster manager that user defined, the cluster master then divide the whole tasks into several partitions and spread them out to worker machines. In this way, the computing cost will be decreased but keeping the computing performance stable.

1.2 Challenges for Multi-label Text Classification Framework

In order to properly classify the published documents, it is essential to have automated systems that can clearly label technical aspects of published papers to allow researchers to ask relevant questions when searching this vast knowledge. Currently, such labeling or categorization require the time and attention of professional experts who are neither interested in carrying out the labeling task nor are they available to do it. As an example, we work with experts who label
published scientific papers with metadata that identifies key aspects of the experimental designs, subject populations, methods used, and so on. As part of their larger workload, this task can take months to label just a few hundred papers. This is a task that clearly needs to be automated given the rate of publication of research papers.

In our previous project, we extracted 3,606 features from scientific articles for training, but the number of training instances to be used in the text classification task is only 247. This is an example of an $n \ll p$ (n much less than p) problem: any problem where the collection of training instances ($n$) is much less than the number of features ($p$, for parameters) [5]. Many types of high-quality data suffer from this $n \ll p$ problem while most methods in data mining have been developed for situations where there is an abundance of (possibly) low-quality data, rather than a limited supply of high-quality data. We adapt a traditional data-mining method to this limited data problem. However, if we use our proposed approach in the large text data set, the performance and computing cost must be considered. Some state-of-art machine learning algorithms and statistical learning strategies have been applied to big data mining; many of them can build good learning model by training a large-scale data set. Unfortunately, the corresponding computing cost is much higher than dealing with smaller data set. To make learning algorithm more efficient and eliminate the risk of over-fitting, we are seeking for a way to decrease the large-scale data dimensionality.

Our research is not limited to build a supervised learning classifier by the golden-standard labeled data, the proposed framework can be used to process the real and unlabeled data. To collect data for specific text mining tasks, cleaning and formatting data are challenges in the preprocessing step. Besides, extracting key features and discovering the hidden information is another interesting topic.

For the multi-label problem, each label represents a different classification task, but the
tasks are somehow related. Therefore, it includes cases that allow for a variable number of labels to be assigned to each instance. For instance, an article in a newspaper or wire service may be assigned to the categories different categories. Currently, the main approach to improve the multi-label classification performance is to transfer the multi-label classification task into multiple single label problem. However, there are two existing issues need to be solved if using this problem transformation. The first issue is feature space is high, the other one is that how to keep the relationship among labels.

1.3 Problem Statement

For text classification, the problem is two-fold: (1) there is no pre-existing, agreed-upon, and fixed vocabulary in use in the sciences that guarantees that important ideas are always expressed in the same terms. Additionally, (2) deeper ideas in research are rarely easily expressed in simple “keyword” terms; they are concepts that require multi-word explanations that contain multiple underlying concepts. Our difficulty is to mine the scientific texts for the words that indicate these underlying concepts and then assign labels that make these concepts explicit. Simple searching for a fixed list of keywords will not suffice for such a complex text structure. In this situation, it is tempting to remove only the usual stop words (short function words which express grammatical structure; e.g., propositions, pronouns, articles, and particles) for the language in question. However, this leaves a collection of features that are too large to use effectively as a starting point for applying standard text mining algorithms; many algorithms work better with smaller feature spaces. This problem is particularly acute when the number of training examples is very small, as often happens in the case of expertly labeled documents used for supervised learning classification problems. In particular, many types of high-quality data suffer from this $n \ll p$ phenomenon, while most methods in data mining have been proposed for situations where
there is an abundance of relatively crude data. Thus, it is a good way to apply feature reduction or feature selection technique in training data when considering the large data set. However, due to training data is extremely huge, before doing the feature reduction or selection, we can do some preprocess the raw data to make dimension reduction more efficiently.

Considering the situation when working on a big data set, we might have to decrease the data dimension to reduce the computation cost, even improve the classification performance. Because our previous project had a good ability to solve the text classification problem, we will use the similar feature selection and reduction techniques to process the training data. In general, feature selection methods are mainly based on the statistical learning or model performance. However, there are countless data without any annotations or categorization. Moreover, those data are unstructured and complex. To discover the important or key entities from unlimited data would be a big challenge. Naïve statistical distribution analysis is a general method that is used to identify the key entities from all kinds of textual data. Because lack of labels or any categories, traditional supervised learning approach is limited to select features.

Our framework is used to solve the “multi-label” problem [2]. This is a type of problem where each instance can be labeled with, in principle, any combination of the labels used in the task. In other work [3], we approached this problem by using a multi-instance multi-label (MIML) algorithm [4]. There we used principal component analysis (PCA) and stemming to pre-process the data before classification [5-7]. It was shown that the PCA components, used as features, allowed a dramatic reduction in feature space dimension. However, as reported there, that may have been an artifact of the data. To address that issue, we use a broader collection of data here. In that work, PCA was shown to be a powerful technique for reducing the size of the feature space.
1.4 Organization

The rest of paper is organized as follows. Chapter 2 provided a literature review for related work. Chapter 3 presents an overview of the framework. Chapter 4 discusses the proposed MIMLfast algorithm with new bagging schemes for solving the multi-label classification problem. Chapter 5 and chapter 6 introduce two approaches to reduce the feature dimension. Chapter 5 studies the hybrid of feature selection methods, chapter 6 presents how to use the Word2vec to lower the feature dimension for huge training data set. After that, Chapter 7 illustrates a method to extract key features from massive unstructured data. At last, Chapter 8 points out the future researches.

2 RELATED WORK

2.1 MIML Algorithm

Comparing with the single instance single label classification, the Multi-Instance Multi-Label (MIML) algorithm has been shown to work well with data that contains complex and rich collections of features and it can be used to predict labels for unseen instances. Many MIML algorithm variations exist [8-10], and work well when the data set is moderate in size. However, they are not effective and efficient in dealing with large data sets in higher dimensions [11,12]. The Figure 2.1 [10] introduces the existing instance-label classification structure. Multi-instance learning and Multi-label learning are the N-1 and 1-N model; they are also being used in text/image classification. The key in the MIML is to discover and keep the relationship between each label and corresponding instances. Zhou et al. [10] propose the MIML algorithm that successful have been used in the image classification. In their framework, the MIML problem is transformed into two tasks. Solution one is using multi-instance as the bridge, the goal is to learn a function
In this sign function, $f_{MIL}(X_i, y) = 1$ if $y \in Y_i$, and $f_{MIL}(X_i, y) = -1$ otherwise. Different to the solution one, solution two is utilizing multi-label learning as the bridge.

To learn a function $f_{MLL}: Z \rightarrow 2^Y$. For any $Z_i \in Z, f_{MLL}(Z_i) = f_{MIML}(X_i)$ if $Z_i = \emptyset(X_i)$, $\emptyset: 2^X \rightarrow Z$ [8-10]. Based on the idea of these two solutions, Zhou et al. [10] propose the MIMLBoost and MIMLSVM algorithms for supervised learning. Their work performed well both in multi-label training and single label (multi-class) training.

![Four Different Learning Framework](image)

**Figure 2.1 [10] Four Different Learning Framework**

### 2.2 Reduce Feature Dimension

In machine learning, both feature reduction and feature selection are the processes of data dimension reduction. The goal of them is to transform the original feature set into a new and lower feature dimension, it is usually used when data size is extremely large. Thus, the major step in data
dimension reduction is to construct a new feature dimension. Feature reduction usually can be viewed as the feature extraction strategy, which transformation can be linear or non-linearity. The advantage of reducing data dimension includes: 1) reduce the computing time and storage space; 2) it is easier to implement the data visualization when data dimension is reduced to 2D or 3D.

2.2.1 Feature Reduction

There are many supervised and unsupervised approaches to transform and reduce the original feature dimension to a new dimension, such as the Principle Component Analysis (PCA) and Singular Value Decomposition (SVD) [59]. Through projecting the initial feature dimension into a lower dimension, the new features would be a good representative of the original feature set. Data clustering is also a good way to help us reduce the data dimension when we group the similar data and use the most important one to represent the whole group. We use both of two strategies in our data and they achieved the good computing performance.

2.2.2 Feature Selection

The straightforward way in feature selection method is selecting the most important variable, but the challenge is how to define the important feature is. In many text mining applications, we simply count the word frequency and choose the most frequent words as features in the future task. However, the frequent words usually are not the important words according to their contribution to text classification task. Thus, we explore the existing advanced feature selection approaches which have been proved very useful and reliable in the mathematics and statistic fields.

Recursive Feature Elimination (RFE; also called RFE-SVM), recursively prunes features according to each feature’s importance [30,44]. Feature usefulness is defined here as the features’ weight in the SVM classifier. Feature importance is determined by sequentially re-training an
SVM classifier and, at each step, removing less useful features. RFE proceeds until the target number of features is left after throwing out all the least useful.

Select K Best (SKB) is a procedure that constructs the $\chi^2$ (chi-square) statistic between each element of the feature space and the labels to determine which features are correlated with which labels [31]. Compute chi-squared stats between each non-negative feature and class, this score can be used to select the n features with the highest values for the test chi-squared statistic from $\chi$. More specifically in feature selection we use it to test whether the occurrence of a special term and the occurrence of a specific class are independent. Thus, we estimate the following quantity for each term and we rank them by their score. High scores on $\chi^2$ indicate that the null hypothesis ($H_0$) of independence should be rejected and thus that the occurrence of the term and class are dependent. If they are dependent, then we select the feature for the text classification. Thus, it rejects features with the smallest $\chi^2$ statistics [32, 33].

Some ensemble learning methods for classification or regression can be used as a feature ranking methods if a relevant importance score can be defined. Random Forests (RF) is a classifier that includes two methods [34-36]: bagging and random subspace. Suppose we have $N$ number of trees in the forest, then we create $N$ datasets created from random resampling of data in original data set with replacement. In order to build a tree, we randomly select subset of features to create. There are a lot of methods to calculate the feature importance, for example, we can compute the entropy for each node. The averages of these scores for each feature order them by importance, and this allows a ranking of features and elimination of less important ones.

Some combinations of methods of single feature selection techniques have been proposed. Li et al. [37] presented a method of combining multiple feature selection approaches by using the combinational fusion analysis (CFA). In the Li et al. paper [37], the authors have shown that a
combination method is able to outperform an individual feature selection method if each one has a scoring function. In another paper, Neumayer et al. [38] have shown the results of a combination of feature selection methods. The individual methods they have used include Document Frequency, Information Gain, GSS-Coefficient, among others. Based on this work, we propose to explore whether combinations of our feature selection methods can improve performance over the use of single methods.

Our primary interest is in the RFE and SKB methods, and we tried using them singly and in sequence, in both orders. The first method in a sequence selects a subset of the original 2,317 words, and then the second selects a smaller subset from this. We used RF as a classifier to select feature subsets with the procedure described above. Because RF generates random subspaces, it was not used as the first method for feature selection [34]. This is due primarily to the randomization (with replacement) missing features from the original feature space. However, RF can efficiently be used after either of the other methods.

Above feature selection methods are based on the supervised learning strategy, but we usually need to select data from millions of unstructured data. This task is a big challenge if people do not have domain knowledge or previous working experience. Few NLP techniques have been used to identify the important entities, but they are limited to specific uses. Kim et al. [82] proposed an evolutionary local selection algorithm (ELSA) for large-scale feature selection with unsupervised learning. Under the K-means guidance, Kim et al. use ELSA to search all possible combinations of features and numbers of clusters [82].

2.3 Artificial Neural Network

Artificial Neural Network (ANN) is a popular machine learning algorithm that has been used in many fields. ANN processes information in a similar way the human brain does. The
network is composed of several or many of highly interconnected neurons working in parallel to solve a specific problem. The neural network allows complex nonlinear relationships between the response variable and its predictors. Normally, ANN consists of two or three hidden layers, but the simple deep neural network contains multiple hidden layers and each layer consists of defined number of units. The type of neural network can be views as deep multi-layers if the number of layers larger than five.

Bengio et. al proposed the terminology “word embedding” and trained data in an Artificial Neural Network (ANN) model [69]. The feedforward neural network took the words from the corpus as inputs and converted them into a lower dimension space. Through back propagation with a fine tune, the word embedding was generated. The neural network language model (NNLM) can learn a probability distribution over words in the corpus. The NNLM is trained to produce the vector representations of a word to capture the semantic information. Through iterating over the whole document, every word is learned and could be represented by a vector with fixed length. Figure 2.2 illustrates the structure of the Bengio’s NNLM, we can find NNLM is similar to the regular ANN but an addition projection layer is being included.

Word2Vec [46, 47], is used to learn vector representations of words, which is known as word embedding. The most straightforward but powerful way to represent words in articles is to transform words into corresponding word vectors, then they will be used in training the statistical model to calculate the relationship among each word from a mathematic point of view. Especially in the neural network, the input data are the word vectors. There are two common learning models in neural network training, the Continues Bag of Words (CBOW) and Skip-gram [47]. The traditional neural network concatenate input word vectors in the projection layer, so the word vector size in the projection layer would be $N \times M$, $N$ is the number of words that surrounding by
the word \( w \); the \( M \) means vector size for every input word. The CBOW in the Word2Vec only sums and averages the input vectors into its projection layer. For the Skip-gram, it is different from the \( N \)-gram \[83\] model in Natural Language Process, the Skip-gram predicts word \( w \) by the surrounding words; while, the \( N \)-gram predicts word \( w \) only calculate previous \( N \) words.

![Diagram of NNLM](image)

\[
\text{ith word} = P(W_i = t \mid \text{context}(i))
\]

**Figure 2.2 Structure of NNLM**

### 2.4 Depression Diagnosis System

In our research, we build an AI-based depression diagnosis system by developing an algorithm to extract and summarize the uncommon but potentially helpful factors that depressive symptom performed from the social media data. We explore Twitter and Web Blogs to collect
depression-related data. Because the social media data is complex and unstructured, data processing and data analysis would be difficult. To clean the data we have collected, we integrate and apply the NLP tools to filter the noisy data. For data analysis, categorizing data could help us discover the hidden relationship of depression symptoms. Thus, the extracted depression symptoms from our system would be used as references when recognizing the clinical depression. Earlier work for driving the depression symptoms on literature help people learn the knowledge of depression detection. Wang et.al [70] applied the Latent Dirichlet Allocation (LDA) [71] as the topic categorizing tool to many of texts on adolescent substance use. Through separating the collections of articles into distinct themes by LDA, the known depressive facts were captured. Their work demonstrated that the topic modeling could be a useful approach to learn knowledge of depression prediction on the structured documents. In contrast to Wang’s work, our challenge is to collect and process the unstructured and complicated social media data. Mitigating the negative effect of the noisy data is our first task. To extract the depression symptoms from the social media data, we use a hybrid method like the one that Ma et. al [7] has proposed in their work. In Ma’s work, they employed the Word2Vec [46,47] to convert each word in the corpus to the corresponding vector. Moreover, their framework has an ability to group words that share semantic similarities by using a clustering method, K-means. We would apply a similar approach to extract the facts of depression in our research. The relationship of these depression symptoms can also be found from the data.

Nowadays, machine learning algorithms can help a doctor do depression diagnosis. Doctors confirm patients’ depression via inputting the symptoms that have been showing in patients, the machine can tell a doctor if a patient is in the risk of depression or classify the patient into depression category. A classification method was used in social media mining [72]. It used
the wide variety of features, such as bag-of-words. An approach was developed for depression diagnosis by analyzing the records of user’s activities in Twitter [73]. The features were extracted from the activity histories of users, such as tweet frequency and tweet frequency. The data were collected from Twitter users who report that they had been diagnosed with clinical Major Depressive Disorder (MDD)\(^1\) [74]. The classifier was built to predict if a person was vulnerable to depression. This paper focuses on selecting the reliable depression symptoms for building an intelligent depression diagnosis system for medical doctors and a convenient depression self-screening system for ordinary people.

### 3 MULTI-LABEL TEXT CLASSIFICATION FRAMEWORK

#### 3.1 Introduction

The great motivation for building a multi-label text classification framework is that text mining is becoming popular and useful in our real lives. The object of text mining could be a short phrase or even a corpus. Originally, most of the documents had been classified manually by the experts with knowledge of linguists and otologists. However, this work has been limited by the large scale of textual data and shortness of human knowledge. Considering the time consuming and expensive labor, related machine learning algorithms and natural language processing tools would be applied to text classification tasks. Through training the data set under a supervisor, a classifier has been built based on the machine learning and statistics learning. Therefore, given a new and unknown document, the classifier assigns each article into the right category or discover the hidden knowledge from the data set. Currently, many machine learning algorithms have been

\(^1\) https://about.twitter.com/company
designed and applied in computer science and statistics field, such as support vector machine, artificial neural network, decision tree and so on. They have been proved they performed well on data mining and machine learning, includes textual data, image data and digital data as well. Especially, the neural network and its dependencies attract more and more attention because they have been successfully built for business application and research purpose. They are not only used in the supervised machine learning, but also applied to the natural language processing. Additionally, deep learning has become more popular and powerful due to its performance on image and text classification tasks. In a word, machine learning algorithm pushes the current world into an artificial intelligence era.

Although machine learning algorithm is a key part of text mining procedure, training data quality is even more important for text classification and annotation. The “data quality” itself is a vague term, it is not easy to define what data set is good or not good. In the text classification task, we build a classifier that divides all documents into variant categories. Otherwise, in the document clustering task, the target is to group very similar documents. Another challenge for text classification is the document quantity. For real situation, we had good quality and gold-standard training data. While the number of data instances is far less than the features that have been extracted from the original data set. To avoid the overfitting risk and to improve the classification performance, we should design a method to reduce the feature dimension. Data dimensionality reduction could be fulfilled by feature selection or feature selection strategies. We develop a new method for feature reduction when it is used in big text data set; also, we build a system to select features using supervised and unsupervised approaches and they would be used in text classification task in the future.
3.2 Overview of Framework

In this paper, we introduce a multi-label text classification framework that would be used for the large data set. Our multi-label classification framework includes two major components. One is used to deal with labeled data with feature reduction, the other one discusses a method to extract features from unlabeled online-generated data. To implement the text classification task, we use some strategies to make our framework powerful when the data size is large enough though. The first section discusses the MIMLfast algorithm to be used for solving the multi-label classification problem, the second section to sixth section present new techniques to reduce the feature dimension in training data. To evaluate the proposed framework, we will use the online public data has big size, then we apply the 10-fold cross validation and F1-micro score to test framework’s performance on the large-scale textual data set. The seventh section introduces a method to extract key features from social media and then use these features into text classification or prediction tasks. Figure 3.1 shows the components of proposed framework.

![Figure 3.1 Overview of Proposed Framework](image-url)
The main purpose of the framework is to classify or predict the documents, such as an article, a tweet or a web blog. The left part that included in the framework is to reduce the training data dimension by employing the entity extraction and hybrid of feature selection under unsupervised and supervised learning guidance. Another highly competitive way to reduce the feature dimension is that we combine a popular neural network called Word2Vec [45] with one of popular clustering algorithms, $K$-means, to achieve this goal. This semi-supervised learning approach uses a nova way to project each document in a new lower dimension.

To deal with the multi-label classification task, different from the traditional binary label classification task, we will keep the correlations among the multiple labels but still lower the feature space. We build new bagging schemes before applying the MIMLfast algorithm into training data. The proposed bagging schemes are based on the multiple different label combinations. Besides, we apply the feature reduction PCA to reduce its feature dimension. This work was done in our previous project, we obtained the valuable results and prove this approach is feasible.

4 MULTI-LABEL TEXT CLASSIFICATION

4.1 Overview of Framework

Although the multi-label text classification project performed well on the small data set, it could illustrate our method would be used in big data set. However, we will show our proposed method is to solve the multi-label text classification problem. We trained all labels together according to our new instance bagging schemes rather than training it separately, this could retain the relationships among the labels. Another technique was used in this project is we applied the feature reduction combined with the MIMLfast algorithm to reduce the feature dimension before
learning. The raw of training data is the text of the published abstracts of 247 human neuroimaging journal articles and their corresponding metadata labels. The text of the abstracts was obtained from the PubMed publication database system (www.ncbi.nlm.nih.gov/pubmed). The full-text articles were marked up as part of the development of the BrainMap database (www.brainmap.org) and were annotated using a standard set of labels (the cognitive paradigm ontology, or CogPO; see www.cogpo.org for details), identifying aspects of the experimental designs reported in each of the papers. See [1] for a summary of this data source and see [19] for full details of the CogPO labels used for classification. These labels were originally added to the BrainMap database by trained expert annotators, based on their reading of the entire text of each journal article. Note that the classifiers developed here only use the partial information provided by the article abstracts, not the full text of the original scientific articles, which is consistent with previous work on automatic classification [1].

The stop words and punctuations were removed from the abstracts using the Natural Language Tool Kit (www.nltk.org) stop word list [20]. The words that remain are the features used for classification. The result was a vocabulary of 3,606 individual words in a fixed order. Many of the words in the vocabulary will have unknown relationships with other words, and we cannot tell, a priori, which features are useful for text classification. Each abstract is then represented by a bag-of-words vector: for each article, we find the locations of the words from the article in the vocabulary and then simply count the occurrences of each word. So each abstract becomes a length 3,606 vector of word counts. These vectors are sparse, as each abstract has, in general, about 200 words, and therefore each vector will have less than 10% non-zero elements in the vector representation. This results in a final data matrix with 3,606 rows (one per word) and 247 columns (one per abstract).
The labels used here are the *response type* labels described in detail in [1, 19]. There are 9 possible labels present in this set of 247 abstracts. Note that this is a true multi-label problem in that articles may be assigned one or more labels. In this particular data, there are 20 unique combinations of the 9 labels present. This expert assignment is used as a gold standard or ground truth for training the classifier. The goal of this research is the development of automatic classifiers that can perform at levels competitive with the human experts and that can be used to annotate articles online in support of web-based search of the scientific literature. This is essential as the classification task is both dependent on significant expertise while being repetitive and boring; this latter aspect making it prone to error. It is also too slow for online use and generally too expensive when done by trained human experts.

Traditional MIML algorithm has great classification performance on moderate data set for image detection and annotation, thus we would use an advanced MIML algorithm in our labeled data set. MIMLfast algorithm, which is proposed by Zhou, has achieved great performance by exploiting label relations with lower shared space and discovering sub-concepts for complicated labels. Since our training data labels have high dimension and the relationship among labels have to be taken into account. Experiments illustrate that the performance of MIMLfast is as good as existing MIML approaches, computing time even is less. In order to find the correlations among multiple labels, MIMLfast initially learn a shared space for all the labels from the original features, and then train label linear models from that shared space. The classification phase, it is similar to the traditional MIML algorithm, the key instance is identified to represent a bag for a specific label, we next train the classifier on the instances, and finally choose the instance with maximum prediction. The way to make learning fast is applying stochastic gradient descent (SGD) to optimize an approximated ranking loss. At each iteration of SGD, MIMLfast randomly samples a
combination which consists of a bag, a relevant label of the bag and an irrelevant label.

Zhou et al. [10] denote \{(X_1, Y_1, \hat{Y}_1), \ldots, (X_n, Y_n, \hat{Y}_n)\} as the training data that consist of \(n\) samples where: each bag \(X_i\) has \(Z_i\) instances \(\{X_{i,1}, \ldots, X_{i,Z_i}\}\); each \(Y_i\) holds the relevant labels present in the corresponding bag \(X_i\); and each \(\hat{Y}_i\) represents the collection of the irrelevant (not present) labels. Both \(Y\)'s are subsets of the set of all labels \(\{y_1, \ldots, y_L\}\).

For multi-label tasks, traditional approaches, such as [14–16], degenerate the multi-label problem into multiple single label problems; however, this strategy may eliminate the relationships that exist among the labels. To overcome this limitation [7] formulates the MIMLfast model as a combination of two parts: the first part is the linear mapping from original feature space to a lower dimensional space (the shared space); the second part then learns the label model from the shared space [7, 17]. The two components work together to fit the training examples with all the labels. All examples contribute to optimize the shared space, and this allows labels find their relationships with each other. Consider a training instance \(x\) and define the classification model on label \(l\) as:

\[
f_l(x) = w_l^\top W_0 x
\]

where \(W_0\) is a \(m \times d\) matrix which (linearly) maps the original features into the shared space; \(m\) is the dimension of the shared space and \(d\) is the dimension of the original feature space; and \(w_l\) is the \(m\)-dimensional linear weight vector for label \(l\). The two models correspond to the optimization of \(w_l\), the weights of the linear model, and \(W_0\), the linear map from features to the shared subspace. See [7] for the specific details of both the algorithm and the implementation in MATLAB [18]. We used this implementation in the present work.
4.2 Bagging Schemes Overview

Before introducing the bagging schemes, we arranged the data for performance analysis. For the MIMLfast experiments in the original feature space, we performed a 10-fold cross validation, a model validation technique for analyzing the performance of the algorithm. In our experiment, we have only 247 original articles. Thus, we separate them into 10 chunks (7 with 25 articles each, and three with 24 articles each). We select nine of the chunks and perform the specific bagging scheme on them; then build the classifier. Finally, we test the classifier on the holdout chunk. We repeat this procedure until each of the chunks has been held out, one at a time, as testing data. No attempt was made to make the chunks used for training representative, so some of the cross-validation runs will attempt to fit labels which are not well trained due to random chance. This is consistent with prior work [1].

The MIMLfast algorithm shows it has good performance on an image or a large text classification, and each bag contains several instances (paragraphs), corresponding labels and unrelated labels. Consider the type of our data, we design different bagging schemes before input our data into the MIMLfast algorithm. In new bagging schemes, each bag contains some of our data examples according to the combination of multiple labels. The other reason why we use different bagging schemes is that we have higher label dimensions, and labels in each dimension are related to the ontology domain.

4.3 Multi-Label Classification and LinearSVC

One approach to multi-label classification (MLC) [21,22] is to convert multi-label classification tasks into single-label (binary) classification tasks, and there are a variety of ways to do this [2]. One method, binary relevance (BR) decomposes multi-label problems with n labels into N independent binary classification tasks [2,23]. In BR, labels are predicted independently
and so label dependencies are not considered [24]. To use binary relevance method, we implement One-versus-the-Rest (OvR) multi-label strategy on our training data [14]. In this approach, the classifier predicts multiple labels by fitting training instances with labels against all instances without the label.

BR is a meta-method that requires an underlying single-label classifier. We used the LinearSVC (Linear Support Vector Classifier) implementation of the Support Vector Machine implemented in Scikit-Learn [32]. This classifier is based on LIBLINEAR [43].

4.4 Bagging Schemes

4.4.1 Data Based Schemes

The first schemes (1 through 4) are determined by the structure of the data and combination of labels. Neither the size nor the number of bags is fixed, but instead the bags are defined by the relationships in the labels.

**Bagging Scheme 1.** In the first bagging scheme, we reduce the problem to single-instance single-label problem. The label power set transformation is implemented in this bagging scheme [16]. We collect all the combinations of labels and each combination is represented as a new, single, composite label. There are 20 unique combinations of the original labels. Each article is placed alone into a single bag. Thus, the number of bags will be equal to the number of articles (247) and each bag has one of 20 new composite labels.

**Bagging Scheme 2.** In this scheme, we bag the articles based on presence of each of the original labels. We make one bag for each label, e.g., bag 1 contains all the articles that have label 1, whatever other labels they may have. This gives 9 bags, one for each original response type label (see above), as training data. Each bag has the original label that defines it.

**Bagging Scheme 3.** This scheme has all the bags in scheme 2 extended with bags defined
by the absence of a given label. Thus, this scheme has 18 bags total: 9 defined by label presence and an additional 9 defined by label absence. Each bag is given the original labels present in it.

**Bagging Scheme 4.** The instances are bagged based on the label combinations, similar to the bagging in scheme 1. However, the data are labeled with the original labels, not the composite labels used there. In our data, there are twenty combinations for all the articles, so we have twenty bags and each bag contains all the articles that have that unique label combination.

### 4.4.2 Fixed Bag Size or Bag Number Schemes

In the schemes above, the number of instances in each bag is not fixed which may negatively influence the performance of the MIMLfast algorithm. In the next two schemes, we fix either the number or the size of the bags. (If unable to divide the data equally, make the last bag flexible.) This is a more general idea that could be fully exploited when applying MIML methods to larger data sets. The following schemes were chosen after some experimentation with possible values.

**Bagging Scheme 5.** In this scheme, we fix the size of each bag. In the first results presented below, we fix 3 instances per bag, which yields 74 bags for the 222 articles used for training in that case. For labeling, collections of single labels are used in this bagging scheme (as in scheme 4)

**Bagging Scheme 6.** In this scheme, we fix the number of bags. In the first results below, this is set to 55, yielding 4 instances per bag for our data.

Table 4.1 displays some details about the different bagging scheme, data based schemes have indefinite the number of bags, whereas the bagging scheme 5 and 6 have fixed number of bags or number of instances in each bag.
### Table 4.1 Summary of Data Based and Fixed Bagging Schemes

<table>
<thead>
<tr>
<th>Scheme Features</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Instances per Bag</td>
<td>1</td>
<td>Indefinite</td>
<td>Indefinite</td>
<td>Indefinite</td>
<td>3 (fixed)</td>
<td>4</td>
</tr>
<tr>
<td>Number of Bags</td>
<td>222</td>
<td>9</td>
<td>18</td>
<td>20</td>
<td>74</td>
<td>55 (fixed)</td>
</tr>
<tr>
<td>Number of Labels</td>
<td>20 (composite labels)</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Label matrix size</td>
<td>222 × 20</td>
<td>9 × 9</td>
<td>18 × 9</td>
<td>20 × 9</td>
<td>74 × 9</td>
<td>55 × 9</td>
</tr>
</tbody>
</table>

#### 4.4.3 Resampling Schemes

While the above schemes are reasonable in cases with very large quantities of data available, they fail when faced with much smaller data sets. As the labels are assigned to bags of instances, resampling the instances may provide some benefit as different combinations of instances produce bags with different label combinations. That is, putting the same abstract text into different bags will not always produce the same labels, as this depends on which other abstracts are also present. We explore this in scheme 7 (not included in Table 4.1).

**Bagging Scheme 7.** Training data is resampled (sampled with replacement) into the bags. In this scheme, we define both the number and the size of bags. We randomly pick instances for each bag and put them back for next pick, repeating this process until all the bags have been made. The goal is to find an optimal ratio between the number and size of bags. This is first used in the feature selection results in table 4.3, where 55 bags of size 5 and 100 bags of size 3 are used.
4.5 Feature Reduction

A traditional approach to the problem of data with too many features is to utilize either a feature selection or a feature reduction technique. Such techniques use various methods to convert the original collection of features into either a subset of the original features (feature selection) or into a new smaller set of derived features. We use three methods here: (1) feature selection by word frequency or count, (2) word stemming, and (3) principal component analysis [39]. We discuss each of these in turn:

**Selection by Word Frequencies or Counts.** Words that occur very often or very rarely are often not useful for text classification [40]. We experiment with various feature reductions based on frequency of occurrence (total word counts) of the words in the texts in the corpus.

**Word Stemming.** Through word stemming, we reduce and simplify words present in the article abstracts by mapping morphological variants onto their stems (or roots). For instance, the words “active”, “activation”, and “activity” all have the same root “activ-”. Even though “activ-” is not a real word, we consider it the stem of all those words, and it represents the common underlying meaning in all the variants. These stems are what are put into the new feature pool. The software we have used for word stemming is the NLTK text mining module [38]. We used the Porter Stemmer [41] and the WordNet Lemmatizer, both of which are NLTK Stem package.

**Principal Component Analysis (PCA).** PCA, also called the Karhunen-Loève transform, is a statistical method that uses an orthogonal transformation to convert a set of observations (possibly correlated) into a set of combinations (sums) of linearly uncorrelated variables, called the principal components. The method assures that each principal component, in turn, has the maximum variance, given that it is uncorrelated with the previously determined components. The goal of PCA is to reduce the dimension of the feature space. We use the transformed features and
their values as our training and testing data to implement the algorithm and evaluate the performance. We used the MATLAB implementations of the PCA algorithm.

4.6 Results

We use several measures to summarize the performance of the algorithm under the various conditions. The algorithm assigns labels to training instances and each label will be in one of the following categories: true positive (TP; present and correctly assigned to the instance), false positive (FP; present but incorrectly assigned to an instance), false negative (FN; not present, but it should have been assigned to the instance), or a true negative (TN; correctly not present). For each label and data instance we can compute the values for these four quantities. In these terms, accuracy is determined by calculating the percentage of TP and TN among the total number of cases with the label. Precision is defined as the proportion of TP against all the positive results: TP + FP. Recall is defined as TP divided by the sum TP + FN. See [40] for details.

The F1-micro (Micro F1) score is an average of the precision and recall [42]. In the micro average, the overall TP, FP, FN, and TN are computed across all labels, and then the measures (precision, recall, etc.) are computed. As this is a multi-label classification problem, the F1-micro measure is the most relevant to the discussion of the results. The F1 measures balance precision and recall, which is essential when labels may be in any of the above categories. Specifically, F1-micro allows the best interpretation across different data. Comparisons with Previous Research.

The central focus of our work is the comparison of MIML performance under the different bagging schemes, so comparison across conditions is emphasized in the tables containing the results [27]. However, for comparison, we report the performance of three standard algorithms applied to this data, all from [1], specifically Table I. Using naïve Bayes with the binary relevance transformation [16] for multi-label classification, an F1-micro score of 0.715 was obtained. For a
support vector machine, the result was an F1-micro of 0.702. (Using the label power set transformation [16], a slightly better result of 0.704 was obtained, but not statistically significant.) Finally, for K-nearest neighbors, using the binary relevance transformation, a maximum F1-micro 0.368 was obtained. Most of the results obtained in this study are between 0.4 and 0.6 in terms of F1-micro performance. So MIML compares favorably with k-nearest neighbors, but poorly with the other two methods, though we note that the best MIML performance is generally comparable with the best methods in [1]. This suggests that further work with MIML may be warranted.

4.6.1 Original Data (No Feature Selection or Reduction)

Table 4.2 shows the performance of six data-based and fixed bagging schemes applied to the original data. This is the case with no feature selection, feature reduction, or resampling. The scores presented are the averages based on the 10-fold cross-validation procedure described above, and this is how all results will be presented throughout this subsection. In terms of F1-micro, schemes 5-6 are both reasonably close in performance and better than the other schemes, with F1-micro results of about 0.50; however, as noted above, previous research shows that simpler methods can achieve F1-micro scores on this same data in the range of 0.70 [1]. This condition serves as a baseline to compare with the feature selection and reduction strategies below. In the following conditions the data is processed before being classified by the MIMLfast algorithm; no adjustments to the algorithm are made.

4.6.2 Count Based Feature Selection

In this condition, we selected subsets of features based on both the upper and lower limits of the word frequencies. Two different resampling schemes are used as well: 55 bags of size 5 (labeled 5x55) and 100 bags of size 3 (3x100). Table 4.3 shows F1-micro results for different word frequency ranges.
Table 4.2 Performance of Bagging Schemes on the Original Data

<table>
<thead>
<tr>
<th>Bagging Scheme</th>
<th>Performance Measures</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
<td>Precision</td>
</tr>
<tr>
<td>1</td>
<td>91.12</td>
<td>33.88</td>
</tr>
<tr>
<td>2</td>
<td>53.10</td>
<td>19.48</td>
</tr>
<tr>
<td>3</td>
<td>52.23</td>
<td>19.64</td>
</tr>
<tr>
<td>4</td>
<td>50.21</td>
<td>18.34</td>
</tr>
<tr>
<td>5</td>
<td>77.72</td>
<td>38.68</td>
</tr>
<tr>
<td>6</td>
<td>76.12</td>
<td>36.58</td>
</tr>
</tbody>
</table>

Table 4.3 Performance of Count Based Feature Selection

<table>
<thead>
<tr>
<th>Word Count Thresholds</th>
<th>Performance (Micro F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Scheme 5x55</td>
</tr>
<tr>
<td>&gt;100 (55)</td>
<td>0.5171</td>
</tr>
<tr>
<td>&gt;80 (79)</td>
<td>0.5434</td>
</tr>
<tr>
<td>&gt;50 (135)</td>
<td>0.5362</td>
</tr>
<tr>
<td>&gt;30 (231)</td>
<td>0.5282</td>
</tr>
<tr>
<td>&gt;20 (349)</td>
<td>0.5353</td>
</tr>
<tr>
<td>30-80 (151)</td>
<td>0.4426</td>
</tr>
<tr>
<td>50-100 (79)</td>
<td>0.4419</td>
</tr>
<tr>
<td>80-300 (72)</td>
<td>0.5209</td>
</tr>
</tbody>
</table>
In the left column of Table 4.3, the word count thresholds are given; for instance, “>100” means words occurring more than 100 times, “80-300” means words appearing between 80 and 300 times, and so on. The number in parentheses gives the count of features preserved by the rule (i.e. how many features are in the feature space for that rule). The first five rows use a lower bound of the word count, while the last 3 use both upper and lower bounds.

It is worth noting a few salient points: (1) the “>80” and the “50-100” condition have the same number of features but very different performance, an F1-micro score difference of 0.08 for the 3x100 condition and a difference of approximately 0.10 for the 5x55 condition. Thus, it is not simply the number of features that matters. (2) With only two exceptions, the conditions appearing in this table all perform better than the cases considered in table II. (3) Note that the difference between the “>80” and “80-300” conditions is just a set of 7 words from the original vocabulary and these are words appearing more than 300 times total in the corpus. For the 5x55 conditions this change makes a difference in F1; for the 3x100 conditions the difference is muted. This suggests that resampling with large numbers of small bags may be less influenced by small changes of the feature space.

Finally note that all F1-micro scores for the 3x100 condition are greater than the corresponding scores for the 5x55 condition. This is consistent with the observation that using more, but smaller, bags are better than the opposite strategy. It is not obvious that this is a general principle.

### 4.6.2 Feature Reduction with Word Stemming

For word stemming condition, we applied the Porter stemmer to the raw data, reducing the number of features from the original 3,606 to 2,017 by this process. In this condition, all the bagging schemes are tried (Table 4.4). For the resampling (scheme 7), combinations 3x55, 6x25,
and 6x55 are used. Focusing on F1-micro, results for schemes 1 through 6 are roughly comparable with Table II (the full original feature space condition), although the results here are strictly greater than for the original feature space. The only dramatic change is for scheme 5. The improvements are likely due to the reduction of the redundancy contained in the original feature space by mapping morphological variants onto their common stems. Of scheme 7, the smaller bags (3x55) compare well with schemes 1-6, while the other resampled schemes perform better than schemes 1-4, but worse than 5-6.

Table 4.4 Performance of Stemmed Data

<table>
<thead>
<tr>
<th>Bagging Scheme</th>
<th>Performance Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
</tr>
<tr>
<td>1</td>
<td>69.75</td>
</tr>
<tr>
<td>2</td>
<td>48.73</td>
</tr>
<tr>
<td>3</td>
<td>45.71</td>
</tr>
<tr>
<td>4</td>
<td>40.09</td>
</tr>
<tr>
<td>5</td>
<td>82.08</td>
</tr>
<tr>
<td>6</td>
<td>77.94</td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>3x55</td>
<td>80.86</td>
</tr>
<tr>
<td>6x25</td>
<td>75.46</td>
</tr>
<tr>
<td>6x55</td>
<td>74.45</td>
</tr>
</tbody>
</table>
4.6.3 Principal Component Analysis

Finally, we reduced the dimensions of the feature space more dramatically using PCA. The principal components (PCs) are derived from the underlying variance structure of the original features, and each abstract is represented as the loadings or weights used in its representation in terms of the PCs. For this section, the estimates of the performance scores are based on either 5 times Monte-Carlo sampling (for Table 4.5) or 20 times Monte-Carlo sampling for Table 4.5. The sampling randomly held out a set of 25 abstracts for each run to be used as a test set.

We considered representations using just the first 5, 10, 50 or 100 PC features. In this condition, each bag contains one unique instance. Table 4.5 shows the performance results for these 4 conditions. Inspired by these results, we explored a variety of values for the number of PC features ranging from 1 to 550. The results are summarized in Figure 4.1. Figure 4.1 shows the F1-micro score for classification based on using varying numbers of PCs to represent the abstracts.

<table>
<thead>
<tr>
<th>Number of PCA Features</th>
<th>Performance Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
</tr>
<tr>
<td>5</td>
<td>88.80</td>
</tr>
<tr>
<td>10</td>
<td>88.62</td>
</tr>
<tr>
<td>50</td>
<td>83.02</td>
</tr>
<tr>
<td>100</td>
<td>81.1</td>
</tr>
</tbody>
</table>

As can be clearly seen in Figure 4.1, for these labels using just a single PC provides better performance than using more PCs to represent the data; in fact, it performs as well as the best result
previously established. We believe that this represents a characteristic of this specific data set; that it is intrinsically low dimensional, and this may be related to the low performance in other conditions when compared with previous research.

![MIML Classifier Performance by PCA Features](image)

*Figure 4.1 Classifier Performance Decreases as the Number of Principal Components Increases*

### 4.6.4 Performance on All Label Dimensions

In Table 4.6, we have applied all bagging schemes except the last resampling bagging scheme on our data set. The results tell us the fifth and sixth bagging schemes overall performed better than other bagging schemes. In addition, we display the feature reduction (PCA) performance on entire data sets in Table 4.7. The experiment results show using PCA would improve the multi-label classification performance. In our experiments, about 10 components are enough for text classification.
Table 4.6 Performance on Entire Label Dimension

<table>
<thead>
<tr>
<th>Label Dimensions</th>
<th>Bagging Schemes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>BDL</td>
<td>0.1649</td>
</tr>
<tr>
<td>DL</td>
<td>0.2692</td>
</tr>
<tr>
<td>PCL</td>
<td>0.083</td>
</tr>
<tr>
<td>ITL</td>
<td>0.1013</td>
</tr>
<tr>
<td>RML</td>
<td>0.3619</td>
</tr>
<tr>
<td>RTL</td>
<td>0.3979</td>
</tr>
<tr>
<td>SML</td>
<td>0.5335</td>
</tr>
<tr>
<td>STL</td>
<td>0.0659</td>
</tr>
</tbody>
</table>

Table 4.7 PCA Performance on Eight Label Dimension

<table>
<thead>
<tr>
<th>Label Dimensions</th>
<th>Principal Components</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td>BDL</td>
<td>0.3558</td>
</tr>
<tr>
<td>DL</td>
<td>0.8437</td>
</tr>
<tr>
<td>PCL</td>
<td>0.3168</td>
</tr>
<tr>
<td>ITL</td>
<td>0.4514</td>
</tr>
<tr>
<td>RML</td>
<td>0.7027</td>
</tr>
<tr>
<td>RTL</td>
<td>0.6378</td>
</tr>
<tr>
<td>SML</td>
<td>0.7278</td>
</tr>
<tr>
<td>STL</td>
<td>0.3835</td>
</tr>
</tbody>
</table>
4.7 Conclusion

This preliminary work on a novel approach to the \( n \ll p \) problem, the problem of classifying small quantities of high-quality data. We adapt an established method for mining larger quantities of data, the MIML algorithm in general and MIMLfast, and explore different ways of organizing the data for the classifier. The performance of the algorithm on the raw feature space is not very good, but with several simple data adjustments, the performance was improved substantially. This suggests that further work with the algorithm is warranted. Finally, we note that the specific data we used in this study may be of a lower dimensional structure than we originally suspected. We continue to explore this in ongoing work.

The MIML algorithm seems natural for this data type and problem. It has emerged from work in image classification where the images to be classified are made up of features related to objects in the pictures, and the goals of classification are to determine the labels or names of the objects pictured. Our expert annotated text is conceptually similar. While results with MIML have not exceeded the previously established benchmark, enough improvement has come from altering the bagging schemes and the feature space that further exploration of these algorithms is worth consideration.

Text annotation takes a great deal of time and effort by the limited number of humans available with the requisite knowledge to do the task. While the production of knowledge has continued to dramatically expand, the availability of tools to catalog, index, classify, and search—more generally to curate—are still lagging significantly behind.
5 HYBRID FEATURE SELECTION

Training data dimension includes column (features) and rows (samples), our previous work mainly focuses on reducing the feature dimension. We implemented two approaches to achieve this goal, one proposed a hybrid of feature selection method; the other one using the Word2Vec and K-means to lower the feature dimension. After feature selection and reduction, each document is projected into a lower dimensionality and is represented by the fixed length vector. Usually, feature selection and reduction are strategies to avoid the overfitting risk in machine learning.

5.1 Introduction

The reason that we use the feature selection into text classification project is that we can make learning more accurate and efficient via discarding the noisy and less useful variables when training. In this project, we also used the artificial data that created by Angela though, the approach was included in the project could give us some hints when we deal with the large data set. In this section, we implemented three feature selection methods on our training data, RFE, SKB and RF. We only evaluated the RFE and SKB individually, but combined two of them as sequential approach as well as using them as a preprocessing step for RF. In our feature selection framework, we not only used several individual feature selection methods, we also applied them into a sequential line to make them work together in order. The purpose for using the hybrid of feature selection methods is that we can hierarchal select the important features in sequence; besides, we built a kind of vote system that every single feature selection method would select features that all of them think those features are important by their sides.
5.2 Single Feature Selection Methods

- **RFE** – The RFE method is used as the only feature selection method. It was run five times, selecting either 50, 100, 150, 200, 250 features and these were used as input for the BR Linear SVC classifier.

- **SKB** – The SKB method is used as the only feature selection method. The details are as for the RFE method above.

- **RFE (300) → SKB** – First, the RFE method selects the 300 most important features; then from these 300 features, SKB selected the 50, 100, 150, 200, 250 most important of these. These five final feature sets are used for the classification as above.

- **SKB (300) → RFE** – The same as the previous procedure, but in the other order.

- **RFE (50-250) → RF** – In this condition, RFE first selects a fixed size subset of the original features; then RF is applied to this subset. Because of the random nature of RF, the final size of the feature subsets it selects are variable; the nominal sizes (those produced by the RFE step) are reported. Note that the actual final feature sets were all much smaller, ranging from 14 to 73 features after the RF step; varying by dimension.

- **SKB (50-250) → RF** – This is the same as the previous procedure, except with SKB as the first feature selector. Here the actual number of features produced by the RF step ranged from 7 to 68; these also varied by dimension.

Figure 5.1 shows classification performance on the eight label dimensions using the above six feature selection methods and combinations. The results presented are 10-fold cross-validated F1-Micro scores, see [10,3] for details. F1-Micro is a score that varies between 0 and 1, with scores closer to 1 being better. Micro averaging the F1 score, across instances, allows for better
comparisons across data sets; however, more complex data sets will intrinsically have lower F1 scores.

The results in Figure 5.1 show there is no profound difference among the feature selection methods when applied to our training data. RFE and SKB (300) → RFE do better overall than the other methods, but most methods show some improvement over the same classification done with the entire original feature space as input. (See Table 5.4 and the discussion of overall results in the section below.)

Figure 5.1 Feature Selection Methods on Eight Label Dimensions

The last two conditions, SKB (50-250) → RF and RFE (50-250) → RF are shown in Figure 5.2 in terms of nominal (input) numbers of features. For comparison, Figure 5.2 shows the same results plotted in terms of actual numbers of features. The main feature to note here is that the actual number of features in use in either of these conditions is always less than about 75; see the descriptions of the methods above for maximum and minimum values [27].
5.3 Common Features Selected by Multiple Methods

Because each feature selection method chooses features by different statistical tests, it is reasonable to expect that the selected features might differ. We considered two different combinations of this type:

- **Common Features (RFE&SKB)** – we first selected 50, 100, 150, 200, 250 features using RFE and corresponding numbers from SKB and then paired up the corresponding sets. We obtained the intersection of the sets, and the features falling into the intersection were used as features for training the LinearSVC classifier.
• **Common Features (RFE&SKB&RF)** – Same as the previous method, but also with the top ranked 50, 100, 150, 200, 250 features from RF and a three-way set intersection.

Table 5.1 shows the number of common features that have been selected from the combination of RFE and SKB. The first row in the Table 5.2 is the top $K$ selected features through RFE and SKB methods. For example, we apply the RFE and SKB individually select 50 features in BDL, there are 13 common features are in RFE results and SKB results as well. Table 5.1 lists the common features that have been selected from the combination of SKB, RFE and RF. Because of the random forest select features randomly, thus the number of common features in the combination of RFE, SKB and RF is less than the combination of RFE and SKB. In the experiment, we will use these features into multi-label text classification task.

*Table 5.1 The Number of Features Selected by Combination of RFE, SKB*

<table>
<thead>
<tr>
<th>RFE &amp; SKB</th>
<th>50</th>
<th>100</th>
<th>150</th>
<th>200</th>
<th>250</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDL</td>
<td>13</td>
<td>35</td>
<td>53</td>
<td>75</td>
<td>96</td>
</tr>
<tr>
<td>DL</td>
<td>14</td>
<td>36</td>
<td>53</td>
<td>82</td>
<td>115</td>
</tr>
<tr>
<td>ITL</td>
<td>12</td>
<td>31</td>
<td>51</td>
<td>75</td>
<td>101</td>
</tr>
<tr>
<td>PCL</td>
<td>10</td>
<td>31</td>
<td>56</td>
<td>75</td>
<td>99</td>
</tr>
<tr>
<td>RML</td>
<td>8</td>
<td>30</td>
<td>45</td>
<td>65</td>
<td>94</td>
</tr>
<tr>
<td>RTL</td>
<td>3</td>
<td>22</td>
<td>35</td>
<td>42</td>
<td>58</td>
</tr>
<tr>
<td>SML</td>
<td>12</td>
<td>39</td>
<td>65</td>
<td>85</td>
<td>117</td>
</tr>
<tr>
<td>STL</td>
<td>7</td>
<td>19</td>
<td>44</td>
<td>66</td>
<td>91</td>
</tr>
</tbody>
</table>
Table 5.2 The Number of Features Selected by Combination of RFE, SKB and RF

<table>
<thead>
<tr>
<th>RFE &amp; SKB &amp; RF</th>
<th>50</th>
<th>100</th>
<th>150</th>
<th>200</th>
<th>250</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDL</td>
<td>6</td>
<td>15</td>
<td>38</td>
<td>41</td>
<td>51</td>
</tr>
<tr>
<td>DL</td>
<td>6</td>
<td>13</td>
<td>18</td>
<td>25</td>
<td>34</td>
</tr>
<tr>
<td>ITL</td>
<td>9</td>
<td>17</td>
<td>28</td>
<td>34</td>
<td>47</td>
</tr>
<tr>
<td>PCL</td>
<td>7</td>
<td>20</td>
<td>26</td>
<td>38</td>
<td>49</td>
</tr>
<tr>
<td>RML</td>
<td>4</td>
<td>14</td>
<td>24</td>
<td>25</td>
<td>41</td>
</tr>
<tr>
<td>RTL</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>24</td>
<td>29</td>
</tr>
<tr>
<td>SML</td>
<td>10</td>
<td>18</td>
<td>24</td>
<td>25</td>
<td>34</td>
</tr>
<tr>
<td>STL</td>
<td>4</td>
<td>13</td>
<td>26</td>
<td>29</td>
<td>50</td>
</tr>
</tbody>
</table>

Figure 5.3 Co-occurrence Features Among Eight Label Dimensions
In Figure 5.3, the $X$-axis is the number of features (in the set intersection) and $Y$-axis is the F1-Micro score obtained by the classifier using the co-occurrence feature sets. The left-hand side shows the co-occurrence features in five feature sets (50-250) that are selected by RFE and SKB; while, the right-hand side shows the co-occurrence features from the three feature selection methods, RFE, SKB, and RF. In general, the performance improves as the number of features decreases. Figure 5.3 is plotted in terms of the actual number of features used for the classifier, not in the nominal 50-250 range. It is important to note the number of features in these conditions is much smaller than most other conditions presented here; one the left the maximum number of common features is 117, or the right it is 51. Performance at the largest number of features in Figure 5.3 is in the general range of performance in the previous figures, but there is no improvement, for the most part. On only one label dimension (PCL or Paradigm Class Labels) did this collection of methods achieve better performance than that seen in other conditions. See Table 5.1 and the next section. It is worth noting that PCL is a conceptually complex dimension compared to some of the others.

5.4 Overall Performance

Table 5.3 displays all the results that performed by feature selection methods on eight label dimensions. The number of selected features start from 50 through 250 as same as the previous experiments.
Table 5.3 Feature Selection Performance on Eight Label Dimension

<table>
<thead>
<tr>
<th>Methods</th>
<th>BDL</th>
<th>DL</th>
<th>ITL</th>
<th>PCL</th>
<th>RML</th>
<th>RTL</th>
<th>SML</th>
<th>STL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RFE</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.4904</td>
<td><strong>0.9292</strong></td>
<td>0.5440</td>
<td>0.4518</td>
<td>0.7924</td>
<td>0.7630</td>
<td>0.8788</td>
<td>0.5412</td>
</tr>
<tr>
<td></td>
<td>0.5108</td>
<td>0.9277</td>
<td>0.6060</td>
<td>0.4683</td>
<td>0.8917</td>
<td>0.8265</td>
<td><strong>0.9107</strong></td>
<td><strong>0.6234</strong></td>
</tr>
<tr>
<td></td>
<td>0.5003</td>
<td>0.9235</td>
<td><strong>0.6374</strong></td>
<td>0.4397</td>
<td>0.8881</td>
<td><strong>0.8395</strong></td>
<td>0.9044</td>
<td>0.6228</td>
</tr>
<tr>
<td></td>
<td>0.4989</td>
<td>0.9188</td>
<td>0.6371</td>
<td>0.4543</td>
<td><strong>0.8936</strong></td>
<td>0.8291</td>
<td>0.8907</td>
<td>0.5951</td>
</tr>
<tr>
<td></td>
<td>0.4878</td>
<td>0.9127</td>
<td>0.6043</td>
<td>0.4467</td>
<td>0.8639</td>
<td>0.8081</td>
<td>0.8678</td>
<td>0.5635</td>
</tr>
<tr>
<td><strong>SKB</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.4204</td>
<td>0.9156</td>
<td>0.4804</td>
<td>0.2251</td>
<td>0.7008</td>
<td>0.6155</td>
<td>0.7925</td>
<td>0.2377</td>
</tr>
<tr>
<td></td>
<td>0.5013</td>
<td>0.9036</td>
<td>0.5113</td>
<td>0.4657</td>
<td>0.7644</td>
<td>0.6852</td>
<td>0.7897</td>
<td>0.4637</td>
</tr>
<tr>
<td></td>
<td>0.4878</td>
<td>0.9025</td>
<td>0.5045</td>
<td>0.4596</td>
<td>0.7515</td>
<td>0.6660</td>
<td>0.7892</td>
<td>0.4314</td>
</tr>
<tr>
<td></td>
<td>0.4935</td>
<td>0.9000</td>
<td>0.5249</td>
<td>0.4585</td>
<td>0.7458</td>
<td>0.6717</td>
<td>0.7870</td>
<td>0.4563</td>
</tr>
<tr>
<td></td>
<td>0.5038</td>
<td>0.8952</td>
<td>0.5189</td>
<td>0.4715</td>
<td>0.7402</td>
<td>0.6840</td>
<td>0.7905</td>
<td>0.4658</td>
</tr>
<tr>
<td><strong>RFE (300) -&gt; SKB</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.4956</td>
<td>0.8750</td>
<td>0.5005</td>
<td>0.4815</td>
<td>0.8059</td>
<td>0.7072</td>
<td>0.8055</td>
<td>0.4492</td>
</tr>
<tr>
<td></td>
<td>0.4825</td>
<td>0.8917</td>
<td>0.5442</td>
<td>0.4764</td>
<td>0.7605</td>
<td>0.7318</td>
<td>0.7992</td>
<td>0.4695</td>
</tr>
<tr>
<td></td>
<td>0.5062</td>
<td>0.8960</td>
<td>0.5587</td>
<td>0.4732</td>
<td>0.7956</td>
<td>0.7107</td>
<td>0.8259</td>
<td>0.4616</td>
</tr>
<tr>
<td></td>
<td>0.4977</td>
<td>0.8975</td>
<td>0.5413</td>
<td>0.4468</td>
<td>0.7982</td>
<td>0.7151</td>
<td>0.8185</td>
<td>0.4955</td>
</tr>
<tr>
<td></td>
<td>0.4889</td>
<td>0.8992</td>
<td>0.5839</td>
<td>0.4261</td>
<td>0.8126</td>
<td>0.7386</td>
<td>0.8391</td>
<td>0.5118</td>
</tr>
<tr>
<td><strong>SKB (300) -&gt; RFE</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.5134</td>
<td>0.9349</td>
<td>0.5421</td>
<td>0.4838</td>
<td>0.8044</td>
<td>0.7218</td>
<td>0.8675</td>
<td>0.4950</td>
</tr>
<tr>
<td></td>
<td><strong>0.5340</strong></td>
<td>0.9177</td>
<td>0.5881</td>
<td>0.4974</td>
<td>0.8241</td>
<td>0.7621</td>
<td>0.8564</td>
<td>0.5251</td>
</tr>
<tr>
<td></td>
<td>0.5023</td>
<td>0.9133</td>
<td>0.5449</td>
<td>0.4744</td>
<td>0.7933</td>
<td>0.7183</td>
<td>0.8411</td>
<td>0.4944</td>
</tr>
<tr>
<td></td>
<td>0.4944</td>
<td>0.9035</td>
<td>0.5402</td>
<td>0.4753</td>
<td>0.7614</td>
<td>0.6987</td>
<td>0.8137</td>
<td>0.4358</td>
</tr>
<tr>
<td></td>
<td>0.4935</td>
<td>0.8999</td>
<td>0.5352</td>
<td>0.4753</td>
<td>0.7347</td>
<td>0.6896</td>
<td>0.7937</td>
<td>0.4322</td>
</tr>
<tr>
<td><strong>SKB (50-250) -&gt; RF</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.3584</td>
<td>0.8994</td>
<td>0.4260</td>
<td>0.1877</td>
<td>0.7129</td>
<td>0.6379</td>
<td>0.8004</td>
<td>0.2357</td>
</tr>
<tr>
<td></td>
<td>0.4785</td>
<td>0.9159</td>
<td>0.5215</td>
<td>0.4437</td>
<td>0.7825</td>
<td>0.7040</td>
<td>0.8247</td>
<td>0.4278</td>
</tr>
</tbody>
</table>
Table 5.4 summarizes the overall results of the study, including a major point that may not be clear from the presentation above: feature reduction does improve performance overall. In Table 5.4, the first row shows the F1-Micro score obtained by the classifier when applied to the entire feature space of 2,317 words (“all features”) without selection. The second row shows the best F1-Micro score obtained across all methods. In every case, at least one feature reduction strategy could do better than all the features used without selection, and very often it is a dramatic
improvement. The smallest improvement is just over 0.05 F1-Micro units, and the largest is almost 0.18. The last two rows give the winning method and the number of features (actual) needed to achieve the score. In general, RFE alone and RFE proceeded by SKB achieved the best results. No more than 200 features were used.

Table 5.4 F1-Micro Scores with Winning Methods

<table>
<thead>
<tr>
<th>Label Dimensions</th>
<th>BDL</th>
<th>DL</th>
<th>ITL</th>
<th>PCL</th>
<th>RML</th>
<th>RTL</th>
<th>SML</th>
<th>STL</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1-Micro Score (All Features)</td>
<td>0.4294</td>
<td>0.8830</td>
<td>0.4746</td>
<td>0.3672</td>
<td>0.7187</td>
<td>0.6808</td>
<td>0.7418</td>
<td>0.3983</td>
</tr>
<tr>
<td>Best F1-Micro Score</td>
<td>0.5340</td>
<td>0.9349</td>
<td>0.6374</td>
<td>0.5044</td>
<td>0.8936</td>
<td>0.8395</td>
<td>0.9107</td>
<td>0.6234</td>
</tr>
<tr>
<td>Method</td>
<td>SKB(300) → RFE</td>
<td>RFE</td>
<td>SKB(300) → RFE</td>
<td>Common (RFE &amp; SKB)</td>
<td>RFE</td>
<td>RFE</td>
<td>RFE</td>
<td>RFE</td>
</tr>
<tr>
<td>Number of Selected Features</td>
<td>100</td>
<td>50</td>
<td>150</td>
<td>56</td>
<td>200</td>
<td>150</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

We are currently developing these techniques using both more data (a larger sample of the BrainMap database), different types of labels with different underlying structure and complexity, and using the full-text of the scientific articles rather than just the text of the abstracts. It should be noted that this last goal, the using of full text, will make the $n \ll p$ problem worse: even with a larger corpus of expert assigned labels, the number of features will dramatically increase if we use the full text of the scientific paper. With this sort of data, this imbalance is here to stay.

The experiment results could tell that proposed ensemble approach performs better than individual feature selection when the feature dimension is much higher than the number of data
instances. In general, around 100 features have been selected by our method is good enough for text classification purpose.

5.5 Artificial Data Evaluation

We evaluated proposed hybrid of feature selection methods on our data, which is golden-standard labeled data with 247 instances * 2317 features dimension. In the next step, we will evaluate our feature selection method on an artificial data set, Genbase [85], we will prove proposed feature selection methods perform well on this data set. Genbase data [85] set contains 662 data instances and 27 features, it is opposed to our previous data set which feature dimension is far larger than the number of data samples. Table 5.5 shows the partial performance on Genbase for multi-label text classification. The second row is the F1-micro without using any feature selection methods.

Table 5.5 Multi-label Classification on Genbase Dataset

<table>
<thead>
<tr>
<th>Methods</th>
<th>F1-Mirco</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL (No feature selection)</td>
<td>0.9711055</td>
</tr>
<tr>
<td>RFE (100)</td>
<td>0.987105505</td>
</tr>
<tr>
<td>Random Forest</td>
<td>0.985112082</td>
</tr>
<tr>
<td>SKB (100)</td>
<td>0.987105505</td>
</tr>
<tr>
<td>RF-&gt; RFE (100)</td>
<td>0.988674132</td>
</tr>
<tr>
<td>RFE (100) -&gt;RF</td>
<td>0.970608959</td>
</tr>
<tr>
<td>SKB (100)--&gt; RF</td>
<td>0.970608959</td>
</tr>
<tr>
<td>SKB (200)-&gt;RFE (100)</td>
<td>0.987105505</td>
</tr>
<tr>
<td>RFE (200)-&gt;SKB (100)</td>
<td>0.987105505</td>
</tr>
</tbody>
</table>
Figure 5.4 displays all the feature selection performance on Genbase data set. Although the improvement is not big, yet the proposed hybrid of feature selection methods has been proved that it could be utilized in many kinds of data sets.

5.6 Conclusion

This chapter presents an approach to feature reduction in large complex textual feature spaces, but using standard methods both alone and in combination. The principal results are surprising, namely that chaining together multiple methods does not necessarily produce an improvement. However, the results obtained do suggest that more work should be done in this area.
We focus on the $n \ll p$ problem, the problem of classifying small quantities of high-quality data. In this case, we have expensive to produce data that can only come from expert work. This type of data will always have an imbalance between $n$ and $p$, that is, there will always be more features than training instances. The performance of the classification algorithm on the raw feature space is not very good, but with several combinations of feature selection methods, the performance was improved substantially.

Currently, text classification takes a lot of time and effort by the limited number of humans available with the requisite knowledge to do the task. The product of the scientific required to be developed very frequent and human have to learn more and more knowledge on machine learning and text mining fields. This is a substantial problem with connections both to deeper issues in machine learning and to interesting technical problems.

6 WORD2VEC PROCESS BIG DATA

Word2Vec has been used in many applications for various purpose, especially in the natural language understanding, such as machine translation and sentiment analysis. Most of the applications generate word vectors by the Word2Vec, we will use the Word2Vec to reduce the feature dimension.

6.1 Overview of Framework

Figure 6.1 shows a structure that using the Word2Vec to process a big text data set, this is another method we proposed to reduce the feature dimension. We used the whole raw documents as input examples and fed them into Word2Vec to generate word vectors. Then created word vectors would be merged into different groups by calculating their distances. At last, all documents
vectors were transferred into new lower feature dimensions, then they would be used in text classification task.

![Diagram of Word2Vec](image)

*Figure 6.1 Working Process*

### 6.2 Word2Vec Structure

Word2vec [45, 46] is not a single algorithm, but a three-layer neural network that process text data. In fact, Word2vec contains two different models (CBOV and skip-gram), each with or without negative sampling training model instead of using the traditional hierarchical Softmax model. Given a word, the skip-gram model predicts the neighboring words according to the given word. In contrast, the CBOV model predicts the current word if given the neighboring words in the surrounding window. Figure 6.2 illustrates the structure of Word2Vec, in the CBOV learning model for example, the inputs are initialized word vectors, then they are summed and averaged as Context(w) into the project layer. In the output layer, we originally build a Hoffman tree based on the word frequency. Each node in the tree has a unique path from the tree root to it. Because
the Hoffman tree is a binary tree, thus, there would be a binary classification through passing a
node to its children. To node \( w \), Word2Vec calculate \( w \)'s probability of \( P(w|\text{Context}(w)) \) according to the Maximum Log likelihood \( \mathcal{L} = \sum_{w \in C} \log p(w|\text{Context}(w)) \) and combine this with binary classification (like sigmoid). To increase the probability of \( P(w|\text{Context}(w)) \), the Word2Vec use the Scholastic Gradient Ascend (SGD) to update the weights. Next, each word vector is updated based on its contribution to construct the vector of \( \text{Context}(w) \). At last, each word in the article will be updated to generate the new vectors. Therefore, the purpose of Word2Vec is to get better representation for each word through either of two learning method, CBOW and Skip-gram. After training each word in an article, each word would have distinct vectors with the same size. We have some benefits for generated word vectors, the most useful one is that the word similarity would be calculated to discover the relation of each word.

Figure 6.2 Structures of CBOW and Skip-gram
6.3 Word Similarity

The training data we use is one popular data set, the 20 Newsgroups data set [50], which was originally collected by Ken Lang. The data is organized and split into 20 newsgroups, and each newsgroup corresponds to the different news domain, such as, sci.med and rec.autos. We only use its 11,314 training data to evaluate our model. Although the training data size is not too large, it should be a good example to introduce our model. Before inputting the training data to Word2Vec, we only remove punctuations for each training instance. We apply the gensim [51], which is a Python library to help us implement the Word2Vec. In this process, we first build a vocabulary from the entire training data. To generate the word vectors well, we employ the Skip-gram model because it has a better learning ability than CBOW if we are not taking computing speed into account [46]. After training, each word attaches a vector. Finally, we construct a high dimension matrix. Each row in the matrix represents every training example and the columns are the generated word vectors. Consequently, the word has multiple degrees of similarity, it can be computed via a linear calculation. For example, vector (“Beijing”) – vector (“China”) + vector (“America”) produces a vector that represents the word “Washington”. Besides, the vector arithmetic can be represented as the format: vector (“Beijing”) – vector (“China”) = vector (”America”) - vector(“Washington”). Thus, the Word2Vec could calculate the similarity between words by the following.

\[
\arg \max_{b^*}(\cos(b^*, b - a + a^*)) = \\
\arg \max_{b^*}(\cos(b^*, b) - \cos(b^*, a) + \cos(b^*, a^*))
\]

Word2Vec generates two numerical vectors X and Y for two different words, the cosine similarity [84] between the two words is defined as the normalized dot product of X and Y:

\[
similarity = \cos(\theta) = \frac{X \cdot Y}{||X||_2 ||Y||_2}
\]
To test the word similarity, we have few examples in Table 6.1, we only list the two most similar words and their cosine distances. For example, given a word “computer”, we obtain the two similar words “evaluation” and “algorithm” with their distances to “computer”. Although there is not a gold-standard vocabulary to evaluate the quality of word vector in natural language process, the results of our experiment show the Word2Vec is succeeded to find the semantically related words.

*Table 6.1 Similar Words*

<table>
<thead>
<tr>
<th>Words</th>
<th>1st Word</th>
<th>Cosine Distance</th>
<th>2nd Word</th>
<th>Cosine Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>computer</td>
<td>evaluation</td>
<td>0.9325</td>
<td>algorithm</td>
<td>0.9292</td>
</tr>
<tr>
<td>president</td>
<td>property</td>
<td>0.9306</td>
<td>population</td>
<td>0.9123</td>
</tr>
<tr>
<td>American</td>
<td>Churches</td>
<td>0.9438</td>
<td>Greece</td>
<td>0.9365</td>
</tr>
<tr>
<td>bank</td>
<td>finance</td>
<td>0.9502</td>
<td>interests</td>
<td>0.9473</td>
</tr>
<tr>
<td>football</td>
<td>team</td>
<td>0.931</td>
<td>Champion</td>
<td>0.9061</td>
</tr>
</tbody>
</table>

6.4 **Word Clustering**

In Natural Language Process (NLP), it might achieve better performance by grouping similar words together [52]. To achieve this, we need to look for the centers of each word cluster. We have learned that the Word2Vec could produce the good quality of word vectors through training the corpus, it would not be difficult to capture the semantic relationship between words. In mathematics, statistics and physics, a vector is a geometric object that has direction and length.
Because each word has multiple degrees of similarity and it can be computed via a linear calculation, our research would benefit from using the Word2Vec. In our work, we could find the similar depression facts by given a depression symptom or a depression related word.

Similar words tend to be close to each other, to group semantic similar words, we use the \( K \)-means [75] to partition the \( N \) objects into \( K \) (\( K \ll N \)) clusters depends on their geometric locations. Initially, \( K \)-means algorithm randomly generates \( K \) seed points as the center of \( K \) clusters, each object is calculated the Euclidean distance between its location and the \( K \) seed points. Then assigning the object to one cluster whose distance from the center of the cluster is the minimum of all the seed points. Next, recalculate the new cluster centers via the formula:

\[
v_i = \frac{1}{s_i} \sum_{j=1}^{s_i} x_j
\]

Where \( s_i \) is the number of objects in the \( i^{th} \) cluster [75]. After new cluster centers are found, compute the distance between each object and the new centers again. Above iteration is operating until there is no object is reassigned to the new clusters. We believe that the word clustering technique could help us understand the relationship among words in our research. Given the different \( K \) values, we will have \( K \) different number of clusters. Therefore, all words are grouped into \( K \) clusters and each word attaches its index of the cluster. Table 6.2 shows few examples that the contents of clusters when \( K \) equals to 500. We can find the similar words are almost correctly grouped into corresponding clusters.
Table 6.2 Cluster Contents

<table>
<thead>
<tr>
<th>Cluster Index</th>
<th>Cluster Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>my, has, is, in, the, had, for…</td>
</tr>
<tr>
<td>2</td>
<td>components, speed, trigger, applications, stage, developers, manufacturers …</td>
</tr>
<tr>
<td>3</td>
<td>Medical, cigarettes, usma, smoked, food, Laboratory, chewing…</td>
</tr>
<tr>
<td>4</td>
<td>population, federal, laws, treasure, crime, organizations, Pope…</td>
</tr>
<tr>
<td>5</td>
<td>bank, financial, interests, client, card, credit …</td>
</tr>
</tbody>
</table>

6.5 Bag of Concept (BoC)

Generally, each document in the text classification is represented by the fixed length vector as we discussed Bag of Words (BoW) approach in previous chapters. In fact, the number of instances (11,314) is much less than the number of original features (61,189). Thus, we will apply the feature reduction method to lower our data dimension by transforming the BoW to Bag of Concepts (BoC). This transformation is done by grouping original words into $K$ clusters, each cluster can be viewed as one topic. The contents of every cluster should be the semantic similar words. At last, the original feature dimension (61,189) is projected to a new feature dimension ($K$). This strategy works as a combination of PCA [21] and LDA, and the processed data dimension will be $11,314 \times K$ ($K << 61,189$) and each document is represented as several topics. Finally, we will use the dimension-reduced data in multi-class classification experiment and values of $K$ that we choose are 500, 1000, 1500, and 2000.
6.6 Multi-Class Classification

In order to evaluate our model’s performance, we fit our data into multi-class classification task as well. First, we convert the multiclass classification into multiple binary classifications. Here, we apply One-vs-Rest [11] technique that is training a single classifier for each class. The instance of one class is viewed as a positive class; the others are considered as negative. We then apply the LinearSVC (Linear Support Vector Classifier) as a classifier and it is based on LIBLINEAR [43].

To evaluate classification performance, we calculate the F1-micro score and record the running time. In addition, we implement the 10-fold cross validation on training data. Table 6.3 [62] shows the performance of multi-class classification on dimension-reduced data sets. The first row shows the F1-micro score and time consuming without applying Word2Vec and clustering algorithm. The rest of rows present the dimension-reduced data classification performance. We can find the performances are a little better than the original data set, but the time cost has a relatively big improvement.

<table>
<thead>
<tr>
<th>Data Dimension</th>
<th>Classification Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F1-Micro Score</td>
</tr>
<tr>
<td>11,314 * 61,189</td>
<td>0.7524</td>
</tr>
<tr>
<td>11,314 * 2000</td>
<td>0.774</td>
</tr>
<tr>
<td>11,314 * 1500</td>
<td>0.7619</td>
</tr>
<tr>
<td>11,314 * 1000</td>
<td>0.753</td>
</tr>
<tr>
<td>11,314 * 500</td>
<td>0.7506</td>
</tr>
</tbody>
</table>
6.7 Conclusion

In our work, we apply the Word2Vec technique into big data processing. Considering the huge data dimension issue when dealing with large-scale training data, Word2Vec provides a way to clustering the similar data. This strategy can be used to reduce the data dimension. To achieve this, we employ two methods.

On one hand, we feed our training data into Word2Vec to generate the word vectors. Through applying the linear calculation on each word vector, we can find semantically related words. On the other hand, we group similar words together using $K$-means clustering algorithm. By given values to $K$, we construct $K$ clusters. Thus, instead of creating word vectors via vocabulary, we make word vectors based on contents that in each cluster. This strategy decreases the data dimension and speeds up the multi-class classification. The new data dimension lowers the time cost in the machine learning task without affecting learning ability from the training data, or even improve learning ability. Therefore, our model will be helpful when dealing with big text data.

7 AI-BASED DEPRESSION DIAGNOSIS SYSTEM

Mental illness has been prevalent in our world, and depression is one of the most common psychological problems. Untreated depression increases the chance of dangerous behaviors. Studying the depression report\(^2\), one in four of those who suffered from the depression or receiving any kinds of treatment. Such a higher rate attracts more studies on detecting and curing depression. Accurate depression diagnosis is a very complex long-term research problem.

For clinic depression, doctors may evaluate the patient via the depression test, such as physical exam or a lab, which are taken by patients. Even a communication between doctors and

\(^2\) http://cep.lse.ac.uk/pubs/download/special/depressionreport.pdf
patients could be used to detect patients’ depression. On one hand, the current depression diagnosis methods are not accurate due to the limited number of symptoms, to discover more depression symptoms over the world is required. On the other hand, the significant challenge of detecting depression is the recognition that depressive symptoms may differ from patients’ behavior and personality [76] such as age, sex, environment, and countries. Unfortunately, there is not a general standard to help doctors confirm patients’ depression. Although we could believe the clinical depression symptoms are more general, yet these clinical records are private and expensive. It is extremely difficult to have these data for our research. To overcome such limitations of clinical depression data, we are looking for another method to collect data and design our own depression diagnosis system based on the data we have gathered.

Public data is being generated every day, countless data are free and huge. Especially, the social medias produce the large-scale of data. Moreover, people discuss all kinds of topics and knowledges on social medias. We believe it would be beneficial to extract useful information via text mining tools on social media [77], such as Twitter, Facebook, Weibo, and Instagram.

To build an AI-based depression diagnosis system, our research is focus on two major steps. In the first place, we propose an approach to select specific features under the unsupervised learning guidance. Next, we can apply selected features from the first step to depression diagnosis system construction. Therefore, the final step will be a text classification or prediction process, which fits to our framework.

7.1 Data Integration

7.1.1 Data Collection

Social media provides real and massive data; it could be an essential resource for research purpose. At past decades, more advanced techniques are required to be developed for social media
data mining purpose. In our research, to extract new depression symptoms that are substitute with common depression facts, we gather data from two social media platforms: Twitter and Web Blogs.

- **Tweets (TW)**

Twitter rapidly has become one of the most popular social media since it launched. Its short 140-character messages that are posted via a smart phone or a personal computer are known as “tweets”, which can be shared by the entire community. Twitter advises 313 million active users who produce 6,000 tweets on Twitter every second as June, 2016\(^3\). Because of this tremendous volume of data, our research would be beneficial from it. In favor of gathering the depression related data, our method is not complicated. We keep monitoring each streaming tweet that includes the word “depression” for almost two weeks in entire Twitter community. Totally, we roughly have gathered 54-million of tweets that discussed the depression relevant field. These data are original raw data that transmitted by users, and they are biased and noisy \([78]\).

- **Professional Twitter Accounts (PTA)**

Although we have collected the depression data from the general Twitter users, another extension of Twitter data collection is that we would gather each tweet that has been posted by professional mental health accounts. The purpose of collecting these specific tweets is that PTAs are more knowledgeable and professional on mental health field. Starting to web scraping the initial webpage\(^4\), thousands of professional mental health tweets have been accumulated at the end. We found many of these tweets discussed the depressive behaviors and their treatments. Therefore, mining the professional twitter account could be used to in our research.

\(^3\) [https://about.twitter.com/company](https://about.twitter.com/company)
\(^4\) [http://treat-depression.com/top-mental-health-accounts-to-follow-on-twitter](http://treat-depression.com/top-mental-health-accounts-to-follow-on-twitter)
• **Depression Blog (DB)**

Other than collecting data from the professional Twitter accounts and active Twitter users, another data resource comes from the depression web blogs. Similarly, web scraping begins at the specific webpage\(^5\) to gather data. These blogs and their deep links are almost referring to the depressive symptoms and relative treatments. This set of data excels both in quality and quantity. Investigating the blog data is another appropriate method to discover the hidden insights of the depression symptoms.

### 7.1.2 Data Preprocessing

Because the data we have collected from the tweets and web blogs are too biased and noisy, they need to be cleaned in the first place. Considering the complexity of data format, we combine several tools and techniques to make data as clean as possible. Table 7.1 shows the example of noisy data and preprocessing tools we have used.

**Table 7.1 Data Preprocessing**

<table>
<thead>
<tr>
<th>Noisy Data</th>
<th>Data Example</th>
<th>Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specific Characters</td>
<td>@RT, http://</td>
<td>Regular Expression</td>
</tr>
<tr>
<td>Punctuations</td>
<td>Comma, colon</td>
<td>Regular Expression</td>
</tr>
<tr>
<td>Stop words</td>
<td>is, the</td>
<td>NLTK Toolkit</td>
</tr>
<tr>
<td>Non-Words</td>
<td>lmao, hrt</td>
<td>NLTK Toolkit</td>
</tr>
<tr>
<td>non-Nouns</td>
<td>eat, wonderful</td>
<td>NLTK Toolkit</td>
</tr>
</tbody>
</table>

---

Generally, the special characters, such as retweet tag “@RT: xxx”, link address “http://www.”, contains less information, so they have been removed at the beginning. In the next step, stop words and punctuations are filtered by stop word list that has been aggregated by us. Although stop words exhibit high word frequency, they could not yield useful information. Non-words are very common in social media data due to any types of typo or acronyms, for instance, “hrt”, “lmao”. These words are removed through checking the English dictionary that is pre-built in the NLTK toolkit [20]. The last step of data preprocessing is to clean the non-Nouns, such as verbs and proposition words. Through employing the Part-of-Speech (PoS) tagging [20] tool on each word, nouns have been extracted. At last, we had the raw data cleaned. Table 7.2 shows the number of words has left after each step of data preprocessing procedure.

Table 7.2 Word Counts

<table>
<thead>
<tr>
<th>Steps</th>
<th>TW</th>
<th>PTA</th>
<th>DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw data</td>
<td>54M</td>
<td>18M</td>
<td>46M</td>
</tr>
<tr>
<td>Non-words removal</td>
<td>7M</td>
<td>2.6M</td>
<td>8M</td>
</tr>
<tr>
<td>Stop words removal</td>
<td>3M</td>
<td>1.2M</td>
<td>3.4M</td>
</tr>
<tr>
<td>Nouns</td>
<td>0.72M</td>
<td>0.2M</td>
<td>0.74M</td>
</tr>
</tbody>
</table>

7.2 Entity Extraction

The most difficult task in our research is to extract meaningful entities related to depression from unstructured data because the data itself contains tedious and useless information. Machine learning algorithms and statistical learning methods were used to select the important features according to the model performance or correlation between each entity and target class. Since there
are no any labels and classes for the collected depression oriented tweets, we use the unsupervised strategy to select entities. We create a new hybrid algorithm integrating the statistical analysis and the NLP strategy to select major depression symptoms. Our ensemble method could find out the common depression symptoms as being used in the clinical depression detection, we could also discover the unfamiliar but useful depression symptoms.

7.2.1 Entity Co-occurrence

The co-occurrence distribution displays the importance of a term in text data. If the probability distribution of co-occurrence between a specific term and the frequent terms is biased to a subset of the frequent terms, then this specific term is likely to be a keyword. Because we have collected all tweets that contain the term “depression”, counting the co-occurrence frequency of each entity that associates with the term “depression” is useful to find the distribution of them. This approach may find out that which entities are common with the term “depression”.

To discover the most frequent entities that co-occur with the term “depression”, a co-occurrence matrix is created. In Figure 7.1, the most frequent entities associated with the term “depression” are shown. For instance, one of the most common symptoms of depression is “anxiety”, it is prevalent in the collected tweets. Depression symptoms are selected by calculating the entities’ counts associated with the term “depression”.
7.2.2 **Entity Similarity**

To analyze the relations between words in the documents, words are initially transformed into vectors. Currently, there are two common strategies to generate word vectors. One is one-hot encoding, and the other one is the word embedding (e.g. Word2Vec). The essential idea of one-hot encoding [79] is to associate each word in the vocabulary with vector representation. Words are represented in a high and sparse dimension; each word corresponds to a point in the vector space. The produced word vectors have a higher feature dimension (vocabulary size $N$), so it is difficult to capture the “relationship” between words.

Mikolov *et al.* [46, 47] extended Bengio’s NNLM [69] and developed the Word2Vec to generate the word representations via different learning methods. The Word2Vec contains two
distinct learning models: Continues Bag of Words (CBOB) and Skip-gram [46, 47]. Figure 9 shows the structures of them. The CBOB model predicts the current word when the neighboring words are given in the surrounding window. In the opposite of the CBOB model, the Skip-gram model learns the context words by giving a word in the input layer and predicts its surrounding words in the output layer. The reason for applying the word similarity as another factor when choosing the key entities is that the Word2Vec performs well on generating the good quality of word representations. Words can be represented by fixed length vectors. Calculating the cosine similarity between words’ vectors can be used to find the similar words of a target term. For example, Figure 7.2 shows the similar words and corresponding cosine similarities given a target word “depression”. This method could discover the key depression related entities from the natural language knowledge perspective.

![Similar words of “depression”](chart)

*Figure 7.2 Similar words related to "depression"*
Another experiment we have done on entity similarity in Figure 7.3 illustrates the hierarchical structure and relationships among the depression facts that have learned from the whole data. In this tree, we extract the most four frequent depression symptoms from whole data and they are shown on the second level. In the third layer, we show a sample of depression facts that are related to one of four common depression symptoms individually. These depression facts have been calculated and accumulated by the Word2Vec given one depression symptom. We believe that extracted depression facts could be good references when recognizing the depressive behaviors.

**Figure 7.3 Hierarchical Structure of Depression and its Symptoms**

7.2.3 **Rapid Automatic Keyword Extraction Algorithm**

The **Rapid Automatic Keyword Extraction (RAKE) algorithm** is described [78]. Candidates are selected from the text by finding all possible strings that do not include stop words or phrase delimiters. The RAKE algorithm generates a list of candidate keywords and phrases. For each candidate, The RAKE algorithm calculates properties to identify the candidate’s importance.
Firstly, compute the frequency of each word. Then, find the degree of every word. The degree of a word is the number of how many times a word is used by other candidate keywords. An individual word’s score is calculated by a formula:

\[
\frac{T + D}{T}
\]

where \(T\) is the word frequency, and \(D\) is the degree of a word. A score is calculated for each phrase that is the summation of the single word’s score. Therefore, the RAKE score can be used in our new approach as a factor to extract the key entities.

Both long and short phrases reveal meaningful information for the information retrieval in natural language processing task, the RAKE algorithm calculates the scores for individual words. Table 7.3 shows the entity and its importance score calculated by the RAKE algorithm. Because the scores shown in Table 7.3 are close to each other, the AKEW algorithm assigns a relatively smaller weight to a RAKE score than other factors.

Table 7.3 RAKE scores for single entity

<table>
<thead>
<tr>
<th>Entities</th>
<th>RAKE Scores</th>
</tr>
</thead>
<tbody>
<tr>
<td>anxiety</td>
<td>1.0121</td>
</tr>
<tr>
<td>insomnia</td>
<td>1.0018</td>
</tr>
<tr>
<td>hate</td>
<td>1.0</td>
</tr>
<tr>
<td>hunger</td>
<td>1.0</td>
</tr>
<tr>
<td>brain</td>
<td>1.0009</td>
</tr>
<tr>
<td>stress</td>
<td>1.0011</td>
</tr>
<tr>
<td>loneliness</td>
<td>1.0004</td>
</tr>
</tbody>
</table>
### 7.3 Automatic Extract Keyword for Specific Terms

We propose an ensemble method to extract depression symptoms from social medias. The relationships between a key entity and its related entities are shown in a semantic graph. To extract important entities for the term “depression”, we develop an Automatic Extract Keyword for specific terms (AEKW) algorithm that combines the co-occurrence count between the term “depression” and other entities, Word2Vec and RAKE. Thus, the AEKW integrates the statistical analysis and word embedding technique to select the key entity relevant to the term “depression”. The AEKW algorithm is given below.

<table>
<thead>
<tr>
<th><strong>AEKW Algorithm</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> term</td>
</tr>
<tr>
<td><strong>Output:</strong> Semantic Graph $G (V, E)$</td>
</tr>
<tr>
<td>$X_i$ = entity in Tweets</td>
</tr>
<tr>
<td>$T_i$ = the co-occurrence counts between entity and the term</td>
</tr>
<tr>
<td>$R_i$ = the RAKE score for entity</td>
</tr>
<tr>
<td>$W_i$ = the similarity between term and entities</td>
</tr>
<tr>
<td>$S_i$ = the score for entities that relevant to term</td>
</tr>
<tr>
<td>$G (V, E)$: $V$ = entities, $E$ = importance scores</td>
</tr>
</tbody>
</table>

**begin**

For entity in Tweets:

- **Calculate the $T$ for $X_i$:** \{$(X_1, T_1)$, \{$(X_2, T_2)$, \ldots \{$(X_n, T_n)$\}$

- **Calculate the $R$ for $X_i$:** \{$(X_1, R_1)$, \{$(X_2, R_2)$, \ldots \{$(X_n, R_n)$\}$

- **Calculate the distance between term and $X_i$:** \{$(X_1, W_1)$, \{$(X_2, W_2)$, \ldots \{$(X_k, W_k)$\}$
The AKEW algorithm counts the co-occurrence of entities paired with the term “depression” and normalizes the co-occurrence counts into the range between 0 and 1. Next, it utilizes the Word2Vec to compute the cosine similarity between “depression” and each entity. The RAKE algorithm is used to calculate an individual entity and assign a score to it. Now, every entity has three scores that are calculated by the above methods. Based on the total score of the three scores for each entity, top \( K \) entities with the \( K \) highest scores are selected. Then, the AEKW algorithm can build a relevant semantic graph. In this semantic graph, the new depressive symptoms may be discovered. Moreover, the important entities related to depression will be used as features in the depression diagnosis and depression self-screening tasks.
7.4 Depression Semantic Graph

The AEKW algorithm calculates the importance score for each entity, then builds a semantic graph to display the relationships between a specific term and its correlated entities. This semantic graph uses the entity “depression” as the central word, the vertices are other relevant entities, and edges represent the importance scores between vertices. In the semantic graph, we may find important entities for depression specifically. Moreover, after expanding the semantic graph, we may discover more unfamiliar and useful depression symptoms. Figure 7.4 displays a partial semantic graph for the central term “depression”.

Figure 7.4 Semantic graph with the center vertex “depression” and the weights represented as the importance score between two vertices
7.5 Conclusion

Clinic depression is a serious mental illness since past decades which negatively affects human’s health. Unfortunately, in contrast to other mental disorder, the depression is persistent. Lots of researches contribute to depression detecting and treatment in diverse science fields. However, it is still a challenge to confirm human’s depression symptoms from their behaviors via clinic records or depression tests. Fortunately, we are living in the big data era. As social media has grown rapidly, this new type of people communication provides a way of sharing any kinds of feelings and knowledge in the community.

Because of clinical data privacy in hospitals, our new method extracts useful depression symptoms from public tweets generated by depression patients, medical doctors and other users. Considering Twitter and Web Blog are valuable data resources; we have gathered data large quantity of data from them. Although there are some studies on detecting the depression symptoms via the social networks or biomedical literature, our work uses the advanced strategies such as web scraping and word embedding, to process and analyze data. To extract more useful depression symptoms as good references, our research truly benefits from the social media data, a new algorithm AEKW is designed by integrating the statistical analysis and the NLP technology. Depression symptoms extracted from the AEKW algorithm are used to construct a depression semantic graph.

8 CONCLUSION

We propose and develop a multi-label text classification framework that can be used to for text mining and text annotation tasks. Because we have a variety of types of data, we investigate different approaches and strategies to analyze data and build a classifier. The main contribution on
proposed framework is to extract feature and reduce the data dimension when using it in the data mining and text annotation.

For the golden-standard and labeled data, we design the new MIML Bagging schemes before applying the MIMLfast algorithm into our work. Compare to the previous work [1], though our method cannot greatly improve the classification performance due to the limited quantity data we have, we find our MIML bagging schemes performed well in the label dimensions which contain much more mutual exclusive labels, such as BDL and PCL. Besides, we implement the feature reduction into the MIMLfast algorithm since we have \( n << p \) problem in our data samples, the overall performance was increased as our expectation. Thus, our proposed new MIMLfast algorithm can be used in text classification.

In the first project, we only apply the PCA to reduce the feature reduction. Next, we develop a hybrid of feature selection methods. Through reviewing the current feature selection techniques, we build the sequential and ensemble approaches to lower the feature dimension. The experiment results show our methods can increase the accuracy in all label dimensions. To evaluate our method on different data set, we collect and train the artificial data under our feature selection guidance, its multi-label text classification accuracy is also increased.

Benefits from the feature selection, we propose a nova algorithm to process the big textual data by using the neural network language model, Word2Vec. Through transforming the document representation from Bag of Words (BoW) into Bag of Concepts (BoC), the classification performance is guaranteed and the computing cost is decreased as well. In addition, our method to represent each document can be used for document summarization and clustering.

Above multi-label text classification work on the labeled data, that is supervised learning. In our framework, we have been working on building a depression diagnosis system. Considering
the clinical data is extremely expensive and limited, we collect and process data from the social medias. The social medias provide communities where users can share their feelings and knowledge with publics, such as Facebook, Twitter, Weibo and Web Blogs, our research would be beneficial from data mining on those social networks. The challenge is to take off the noisy data and discover the true important key words that related to the depression. Therefore, we build a AEKW algorithm to define the importance of an entity and build a semantic graph to display the most important key words that related to the term depression. In this method, we extract the depression symptoms from the social medias. Many of these depression symptoms are common symptoms that are known; while, our approach can find some useful but unfamiliar depression symptoms to people. They are good references for doctor to detect and confirm patients’ depression.

Our present research goal is to build an AI-based depression diagnosis system. In the future, we will use the semantic graph to generate major depression symptoms from Twitter, then apply the association rule mining [86] to find mutual relations among top symptoms without labeled data. These major strong association rules can be used for initial depression diagnosis.

In addition, we will design a grammar based approach to identifying the depression users on Twitter. Through combining the depression symptoms and depression users’ tweets, a classifier will be built for accurate depression diagnosis. An intelligent depression diagnosis system for medical doctors and a convenient depression self-screening software system for ordinary people will be developed.
REFERENCES


[50] K. Lang, “20 newsgroup data set”. Available at: qwone.com/~jason/20Newsgroups/


[65] Jesse Read, Bernhard Pfahringer, Geoffrey Holmes, and Eibe Frank. Classifier chains for


