OPEN DOMAIN TARGETED SENTIMENT POLARITY DETECTION FOR MICRO-BLOGS IN SOCIAL MEDIA

BY

SHADI IBRAHIM HAFIZ ABUDALFA

A Dissertation Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

In

COMPUTER SCIENCE AND ENGINEERING

May, 2018

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN- 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **SHADI IBRAHIM HAFIZ ABUDALFA** under the direction his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE AND ENGINEERING.**

[illegible]

Dr. Moataz Ahmed
(Advisor)

For: [illegible]

Dr. Adel F. Ahmed
Department Chairman

[illegible]

Dr. Mohammad Alshayeb
(Member)

[illegible]

Prof. Salam A. Zummo
Dean of Graduate Studies

عمادة الدراسات العليا
DEANSHIP OF GRADUATE STUDIES

[illegible]

Dr. Tarek El-Bassuny
(Member)

1/8/2018
Date

[illegible]

Prof. Aiman El-Maleh
(Member)

[illegible]

Prof. Sadiq M. Sait
(Member)

DEDICATION

I would like to dedicate this work with my deep love to my parents, brothers, and sisters for their support and prayers.

This work is also dedicated to my wife, daughter, and sons for being my inspiration.

ACKNOWLEDGMENTS

First, and foremost, all the praises and thanks are due to ALLAH Almighty for giving me the health and ability to carry out this research.

I would like to express my deepest gratitude to my advisor Dr. Moataz Ahmed for his encouragement, support and guidance throughout this work. My appreciation is extended to the committee members for their cooperation and valuable comments.

I would also like to thank my instructors in College of Computer Sciences and Engineering. Moreover, I acknowledge all the support given by King Fahd University of Petroleum & Minerals during my graduate studies and research work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

SVM	:	Support Vector Machine
NLP	:	Natural Language Processing
CRF	:	Conditional Random Field
SA	:	Sentiment Analysis
QN-S3VM	:	Quasi-Newton Semi-Supervised Support Vector Machines
NER	:	Name Entity Recognition
kNN	:	k Nearest Neighbors
RBF	:	Radial Basis Function
LDA	:	Linear Discriminant Analysis
PCA	:	Principle Component Analysis
HMM	:	Hidden Markov Model
CNN	:	Convolutional Neural Networks

ABSTRACT

Full Name	:	[Shadi Ibrahim Hafiz Abudalfa]
Thesis Title	:	[Open Domain Targeted Sentiment Polarity Detection For Micro-Blogs In Social Media]
Major Field	:	[Computer Science and Engineering]
Date of Degree	:	[May, 2018]

Tremendous amount of opinions, regarding almost every topic, is available on the internet these days through social media. Evidences show that such opinions play important role in our life and affect behavior of individuals, communities, industries, and governments. Availability of such wealth of opinions in the social media motivated researchers to develop automated systems for opinion mining, also known as sentiment analysis. A sentiment represents a polarity of opinion that is typically expressed as one of three classes: positive, negative, or neutral. Many tools are currently available for sentiment mining in short text, referred to as micro-blogs, for different languages but their efficacies are still limited. Such limitations include and not limited to dealing with specific domains and providing limited performance.

In this work, we developed an approach for topic identification and polarity classification of opinions offered in the form of micro-blogs. We propose a new context-based analysis system for detecting targets among a set of micro-blogs and detecting sentiment polarities towards categorized topics that describe the targets. Our literature review revealed that the research direction has been originally focusing on classifying sentiments polarities towards specific targets, i.e., topics, in the micro-blogs. A more recent direction currently addresses the problem of detecting the target and identifying the sentiment polarity toward it. While, the former direction is referred to as target-dependent sentiment classification, the latter one is referred to as open domain targeted sentiment classification.

Our literature review also revealed that majority of the state of the art approaches use supervised learning techniques for both target-dependent and open domain targeted sentiment classification. Such techniques need a huge amount of labeled data for increasing classification accuracy. However, preparing labeled data from social media needs a significant effort and may cause inaccurate results if some micro-blogs are annotated incorrectly. For that matter, we propose new techniques to employ semi-supervised learning methods for improving the performance of both target-dependent and open domain targeted sentiment classification by using partially labeled data.

Additionally, we propose new supervised techniques for improving the performance of both target-dependent and open domain targeted sentiment classification. Numerous experiments are conducted to show that our proposed techniques outperform prominent ones available in the literature. A comparison framework and statistical analysis are included in this work as well to validate experiment results.

ملخص الرسالة

الاسم الكامل: شادي إبراهيم حافظ أبوضلفة

عنوان الرسالة: كشف قطبية وجهة النظر للموضوعات المطروحة في المدونات الصغيرة على وسائل التواصل الاجتماعي خلال النطاق المفتوح

التخصص: علوم وهندسة الحاسب الآلي

تاريخ الدرجة العلمية: مايو، 2018

يتوفر هذه الأيام كم هائل من الموضوعات والآراء المتاحة على شبكة الإنترنت من خلال استخدام وسائل التواصل الاجتماعي، مما يدلل على أن هذه الآراء تلعب دورا هاما في حياتنا، وتؤثر على سلوك المجتمعات والحكومات. وفرة هذه البيانات على وسائل التواصل الاجتماعي يفتح الباب أمام العلماء لتطوير النظم الآلية المناسبة لتحليل المشاعر والآراء. تم تطوير العديد من الأدوات لكشف الآراء آليا عبر النصوص القصيرة ولكنها لا تزال محدودة الكفاءة.

في هذا البحث، قدمنا العديد من التقنيات الجديدة والمناسبة لتحليل المشاعر والآراء على وسائل التواصل الاجتماعي. واقترحنا نظام جديد لتحليل النصوص القصيرة وإستكشاف سياق الكلام بالكشف عن الموضوعات المطروحة واستخلاص قطبية الرأي تجاهها بخصوص التصنيفات الإيجابي والسلبي والمحايدة. يتركز بحثنا على الكشف عن أي موضوع مطروح في النص ومن ثم تحديد قطبية الآراء تجاهه ، والذي يشار إليه بالنطاق المفتوح لتحليل المشاعر للموضوع المستهدف.

قدمنا آليات جديدة لتحليل المشاعر في النطاق المفتوح للمواضيع المستهدفة وتصنيف الآراء استنادا إلى تقنيات تعليم الآلة بالإعتماد على عدد محدود من البيانات المؤشرة. ويشمل عملنا البحثي كذلك إقتراح طرق جديدة تزيد الدقة في إستكشاف قطبية الآراء. وأجرينا كذلك العديد من التجارب والتحليلات لتوضيح قوة الآليات المقترحة بالمقارنة مع الدراسات السابقة التي تم أنجازها في هذا المجال.

CHAPTER 1

INTRODUCTION

In the last decade, the social media has become a major part in our life through including its services to build social networks or social relations with other people who share similar personal or career interests, activities, backgrounds or real-life connections. For example, due to advances in information technology and communication, the number of users in social networks has increased significantly.

Social media plays the most effective role in enabling people to freely share their opinions with regard to almost everything. Popularity of social media sites has been increasing sharply in the world especially after spreading civil movement in many countries. This popularity assists in generating a massive data for different topics especially on some famous social media sites such as Facebook and Twitter.

The availability of tremendous public opinions opens the door to researchers and scholars to mine people's polarity of opinions with regard to almost every topic of interest in almost any domain. This introduces what is popularly known as sentiment analysis. Nowadays, sentiment analysis is employed in many services that are available on the Internet. For example, it is used for detecting polarity of opinions expressed in forums.

1.1 General Problem Statement

Sentiment analysis which is also known as opinion mining is one of the major tasks under umbrella of natural language processing (NLP) [1][2]. It is also one of the active research areas in text mining (TM) which has gained much attention nowadays. The main goal of sentiment analysis is identifying polarity of opinions [3]. Sentiment analysis includes numerous subtopics such as polarity classification [4], subjectivity detection [5], review summarization [6], and rumor detection [7][8]. Our research focuses specifically on polarity classification. In the rest of this dissertation, we use term *analysis* to describe mainly classification task. However, there are other tasks included such as detecting named entities.

Sentiment analysis has been included in many systems. For example, many websites of electronic commerce provide services to recommend products and analyze product reviews. Sentiment analysis is a main component in these websites to find buyer's opinion and increase purchasing power. Another important example is related to predicting directions of voters in the election process by analyzing their opinions on the social media through governmental institutions.

State-of-the-art systems for sentiment analysis deal mainly with three levels of annotation granularity towards the input: document, sentence, aspect, or phrase (word) [9]. Our research focuses on the sentence level and especially a short sentence namely micro-blog in social media. Different tools are available nowadays for opinion mining of micro-blogs. Typically, the input to such tools is a short sentence that is gathered from the social media by querying about a specified target (what the opinion is about). The output is the opinion polarity that is inferred from the input text and expressed in one of three options: positive, negative, or neutral.

Most available sentiment analysis tools are based on target-independent strategy. Thus, these applications may fail to assign correct sentiment polarity to a micro-blog that includes more than one target (topic). For Example, consider the micro-blog: “Windows is much better than iOS!”. A target-independent system would classify this micro-blog as positive since it contains only positive words (much better). However, a target-dependent system would classify this micro-blog as negative if “iOS” is a target of interest. Otherwise, it would be classified as positive if the target of interest is “Windows”.

A more challenging scenario deals with detecting the name entities (targets) in the micro-blog and identifying sentiment polarities toward them. Referring to the above example, the system detects firstly words “Windows” and “iOS” as targets and then identifies opinions toward them as discussed previously. Such scenario is referred to as *open domain targeted sentiment classification* which helps in detecting opinions in micro-blogs towards any named entity (such as person or organization).

According to the best of our knowledge, all previous work proposed only supervised learning techniques for improving the performance of both target-dependent and open domain targeted sentiment classification. Thus, training all previous supervised learning methods require a huge amount of labeled micro-blogs. While, preparing labeled micro-blogs is a time-consuming process and usually leads to inaccurate results.

Providing a huge number of labeled micro-blogs needs significant effort since we need to annotate them manually. Using manual methodology for annotating micro-blogs may lead to inaccurate results that are related to human errors. Moreover, employing additional workers for annotating micro-blogs more accurately will increase efforts and may cause biased decisions. On the other hand, using automated tools [10] for annotating micro-blogs may affect also on classification accuracy since their efficiency are still limited. As a result of this, some micro-blogs may be annotated incorrectly. It is worth mentioning also that supervised learning machines may converge to overfitting phenomenon [11].

In this research, new techniques have been proposed for improving the performance of both target-dependent and open domain targeted sentiment classification. We improve the performance by increasing classification accuracy or improving the results of other evaluation measures such as F1-score, precision and recall. We evaluated and validated the applicability of different learning techniques such as supervised, unsupervised techniques to our problem. We propose as well new semi-supervised learning techniques that decrease the need to use only labeled data and overcome the overfitting phenomenon for both target-dependent and open domain targeted sentiment classification. To the best of our knowledge, our work is the first research that employs semi-supervised learning techniques in both research directions.

Additionally, a new system is proposed that differs from status quo followed in developing systems for open domain targeted sentiment classification. Existing systems detect opinions in each micro-blog individually. While, our proposed approach helps in developing a context-based analysis system among a set of micro-blogs. The proposed system detects context patterns among a set of micro-blogs by detecting targets and identifying sentiment polarities towards categorized topics that describe the detected targets.

1.2 Contributions

This section summarizes the main contributions of our research work by describing briefly new techniques proposed for improving both target-dependent and open domain targeted sentiment classification.

1.2.1 Survey on Target-Dependent Sentiment Analysis

We carried out a comprehensive review on sentiment analysis in social media. A survey on target-dependent sentiment analysis is carried out also with summarized results. The survey revealed some gaps to be addressed in future research and illustrates that there are still many limitations in previous research works. Some discussions are included in this survey on target-dependent sentiment analysis as promising future research direction. Findings have been recently published [12].

1.2.2 Comparative Study on Target-Dependent Sentiment Analysis

An extension to our recent survey has been presented by compiling accuracy reported by researchers with respect to the application of different techniques applied to the same dataset. Our study presents comparisons between different techniques with regard to both the target-dependent and the open domain targeted sentiment classification. The study identifies some gaps to be addressed in future research. For instance, it shows that performance of both target-dependent and open domain targeted sentiment classification is still limited, and further future research could be promising. Findings have been recently published [13].

1.2.3 Target-dependent Sentiment Analysis

Performance of applying many supervised learning techniques has been evaluated and new solutions are proposed for improving the performance of target-dependent sentiment classification. Experiment results are provided to show efficacy of the proposed solutions. Additionally, we have addressed the difficulty of preparing labeled data from social media by employing semi-supervised learning techniques that have not been used before for detecting opinion polarity of micro-blogs based on target-dependent sentiment classification.

To the best of our knowledge, our work is the first research that employs semi-supervised learning techniques in this research direction. We also have proposed a new semi-supervised learning technique that uses partially labeled micro-blogs. Experiment results show that the proposed technique performs competitive performance.

Moreover, efficiency of using deep learning techniques has been addressed for improving the performance of target-dependent sentiment classification. We have compiled all previous works that employed deep learning techniques for both target-dependent and open domain targeted sentiment classification. We evaluated as well the efficiency of applying neural networks and deep conventional neural networks on target-dependent sentiment classification. Findings have been recently published [14].

1.2.4 Open Domain Targeted Sentiment Analysis

In this research direction, adequacy of developing new semi-supervised learning technique has been addressed to serve open domain targeted sentiment classification. Two new solutions have been proposed for improving the performance of open domain targeted sentiment classification. The first solution is a supervised learning technique while that other one is a semi-supervised learning technique.

To the best of our knowledge, our solution is a new semi-supervised learning technique that is proposed for open domain targeted sentiment classification. We conducted numerous experiments for showing that the proposed solutions outperform other pervious related works. Additionally, a new system has been developed for context-based target-dependent sentiment analysis. The proposed system detects context patterns among a set of micro-blogs by detecting targets and identifying sentiment polarities towards categorized topics that describe the detected targets.

1.3 Thesis Organization

The rest of this dissertation is organized as illustrated in Figure 1.1. Chapter 2 presents a background on some topics necessary to understand the rest of the dissertation. Chapter 3 includes a literature review to related works in the state of the art. Chapter 4 defines the research problem and describes the approach used for finding a solution. Chapter 5 describes the experiment design used for conducting our experiments. Chapter 6 describes in details our solutions proposed for improving the performance of target-dependent sentiment classification. Chapters 7, 8, and 9 show experiment results provided by using target-dependent sentiment classification. Chapter 10 describes in details our solutions proposed for improving the performance of open domain targeted sentiment classification. Chapters 11 shows experiment results provided by using open domain targeted sentiment classification. Finally, Chapter 12 concludes the dissertation and presents suggestions for a future work.

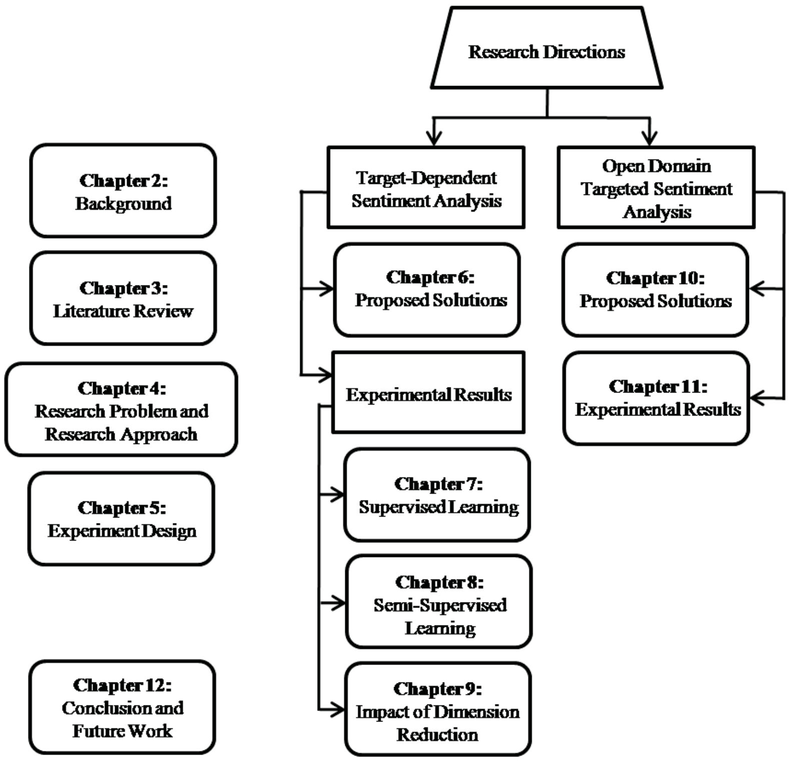

Research Directions
Target-Dependent Sentiment Analysis
Open Domain Targeted Sentiment Analysis
Chapter 2: Background
Chapter 3: Literature Review
Chapter 4: Research Problem and Research Approach
Chapter 5: Experiment Design
Chapter 12: Conclusion and Future Work
Chapter 6: Proposed Solutions
Experimental Results
Chapter 7: Supervised Learning
Chapter 8: Semi-Supervised Learning
Chapter 9: Impact of Dimension Reduction
Chapter 10: Proposed Solutions
Chapter 11: Experimental Results

Figure 1.1: Overview of the dissertation organization.

CHAPTER 2

BACKGROUND

This chapter presents an overview on the general framework of sentiment analysis. We also present topics necessary to understand the rest of the dissertation.

2.1 General Framework for Sentiment Analysis

Sentiment analysis of micro-blogs can be accomplished by passing through different stages [15] as shown in Figure 2.1. It starts by collecting data and building corpus of micro-blogs. After collecting data, the next step is preprocessing the data by removing unrelated contents and keeping the text only. Then, the filtering stage is applied for removing unnecessary words without affecting the meaning of input micro-blog. The next stage is extracting features from the text and selecting the best ones. This stage converts the text into a vector of feature attributes which is referred to as a data point. The final step includes applying classification methods to classify the data point into different classes such as positive, negative, and neutral. The following subsections describe some details for these stages which are related to our work.

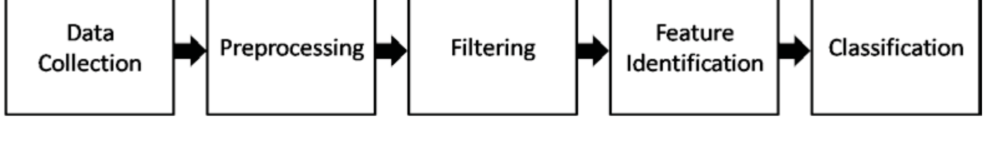


Figure 2.1: General framework for sentiment analysis.

2.1.1 Data Collection

In this step, various micro-blogs should be collected from different topics to train the proposed system and test the performance. The collected data can be used also to build a big corpus for evaluating the proposed techniques and providing experimental results.

2.1.2 Preprocessing

The preprocessing step is used to clean text from unsentimental contents, such as user-names, pictures, hash-tags, and URLs [16]. These contents may decrease accuracy of classification system. For example, URLs and user-names are not related to the topic of micro-blog. After the preprocessing stage the outcome will be only a pure text.

2.1.3 Filtering

Some proposed techniques use this stage to filter the micro-blog before extracting feature attributes from it. Filtering stage may include many steps to enhance the text for increasing the accuracy of the classification technique. One of the required steps in filtering stage is correcting misspelling since most of bloggers type micro-blogs quickly and some of them do not have enough capabilities to spell words correctly.

Moreover, some micro-blogs contain words with repeated letters and the filtering stage should tackle theses words by removing the repeated letters manually or automatically [17]. There also some stop words that may be included in tweets such as prepositions. The stop words will not affect the meaning of micro-blogs and they should be removed in this stage. Finally, the filtering stage will normalize the micro-blog by removing punctuation, non-letters, short vowels, etc.

2.1.4 Feature Identification

This stage is referred to also as feature engineering. In this stage, different features are extracted from the filtered text by using different methods proposed in the literature. The most suitable features should be also selected to reduce dimensionality of the accumulated data. The output of this stage is a vector of feature attributes that is referred to as a data point.

2.1.5 Classification

This is the final stage in any sentiment analysis system. Numerous methods could be used in this stage to classify the data points that are generated from the previous stage. The outcome of this step expresses the sentiment polarity toward each input micro-blog. The outcome, for example, may be represented as one of three options: positive, negative, or neutral. Most of classification methods are based on supervised or unsupervised machine learning techniques. Section 2.3 presents different machine learning methods.

2.2 Challenges

Sentiment analysis on sentence level is a very difficult task by itself and there are additional challenges for dealing with non-English languages [18]. This section describes some difficulties that are revealed when developing sentiment analysis systems.

2.2.1 Difficulties Related to Sentiment Analysis

Detecting opinion polarities expressed in text is a challenging task. The following difficulties affect on the general process of sentiment analysis regardless of which language is used:

1) People do not always express their opinions in the same way.

2) The word which is expressed to be positive in some sentences may be considered as negative in others.

3) Some web users use different opinions in the same text which is easy to classify by human and difficult to parse by machine.

4) Shuffle words in the same sentence will change semantic meaning which makes sentiment analysis more difficult.

5) Using negations in sentence is still an open research problem in sentiment analysis.

2.2.2 Difficulties Related to Social Media Text

There are many difficulties for applying sentiment analysis specifically in social media. The following points describe some challenges related to this issue:

1) The used language in social media is highly unstructured and contains misspellings, slang words, contractions and abbreviations.

2) The content of text messages such as tweets includes many peculiarities. For examples: string “RT” is an acronym for a “re-tweet”. The hash-tag “#” is used to organize tweets. Emoticon “:-)” indicates a smiley face. The tweets also may include external web links.

3) The produced data is continuous with a large and uncontrolled number of users.

4) The produced micro-blogs in social media tend to be very short, for example tweet length is limited to 140 characters. This limitation increases hardness of sentiment

analysis in comparison with using document that includes more information for detecting opinions.

5) Words and phrases that are used by web users for expressing their sentiments are subjective and tend to user cultures. For example, some users use a metaphor to describe their opinions which increases difficulty of sentiment analysis.

2.2.3 Difficulties Related to Non-English Language

Analyzing text of a non-English language is a difficult task in comparison with English language. The following points shed the light on some difficulties for identifying sentiments in Arabic micro-blogs [19]:

1) The Arabic language is a rich language and one lemma can have thousands of surface forms.

2) The unavailability of Arabic labeled corpora is one of the serious issues which are revealed when building systems for Arabic sentiment analysis.

3) Most of Arabic tweets contain informal phrases since web users use different dialects.

2.3 Machine Learning

This section presents a background to the machine learning methods that are used in our work. These methods are used to make comparisons with our proposed solutions and develop more efficient systems for target-dependent sentiment classification. These methods are categorized into supervised, unsupervised, and semi-supervised learning methods as presented in the next three subsections.

2.3.1 Supervised Learning

Learning-based method is the most used classifier in sentiment analysis. Supervised learning method uses only labeled data for training the model. We used various supervised learning methods for checking their efficiency in sentiment classification and making comparisons. The next subsections describe some methods used in our research work.

2.3.1.1 Decision Tree Classifier

Decision tree classifier [20] creates a model for detecting the label of a new data point by learning some decision rules that are formed from the data features. This classifier is simple to understand and can be visualized easily. It can be also used as a baseline classifier since it does not use any parameter.

Decision tree classifier suffers from some limitations because it is based on the structured decision tree. It is an uneasy task to describe all data using a decision tree and creating a complex decision tree may end up with the overfitting problem. Moreover, finding an optimal decision tree for describing the training data is an NP-complete problem.

2.3.1.2 Naive Bayes

Naive Bayes [21] is one of the most famous learning methods and used frequently by scholars for improving performance of sentiment analysis. This method is based on Bayes theorem for calculating confidence level of classifying classes. It uses also naive independence assumption when states relationship between each pair of feature attributes. There are many classifiers that are inspired by Naive Bayes approach such as Gaussian Naive Bayes classifier which assumes that the likelihood of features is Gaussian.

2.3.1.3 Discriminant Analysis

Discriminant analysis classifier [22] uses Bayes rule to generate a boundary that splits all dense regions in the training data. There are two main types of Discriminant analysis. The first one is a linear discriminant analysis which learns only linear boundaries for fitting the dense regions of data. The second one is quadratic discriminant analysis which is used to learn quadratic boundaries in a more flexible way.

2.3.1.4 Nearest Neighbors

The main principle behind nearest neighbors approach is based on finding the closest data points to label them as same class. There are two main classifiers that are based on principle of nearest neighbors. The first one called k-nearest neighbors (kNN) classifier which finds *k* data points closest in distance to the new data point and detect the label from these *k* data points. The distance can be measured by numerous measures such as Euclidean or Cosine. To decrease time complexity of finding *k* nearest neighbors, process of accessing data points is transformed into fast indexing structure such as KD-tree [23][24] or Ball tree.

The second main classifier is based on finding the neighbors that are close to the nearest centroid classifier. The idea of building nearest centroid classifier [25] is close to label update phase in K-means clustering algorithm. It calculates centroids for representing each class in training phase and uses these centroids for detecting the label of the new data point in testing phase. This classifier can be used as baseline classifier since it does not use any parameter. It also has limitations similar to K-means algorithm as it is not suitable to classify classes of non-convex shapes.

2.3.1.5 Generalized Linear Models

Linear models are based on performing recognition by detecting the target value to be a linear combination of inputs. These models are designed basically to deal with binary classification problem. Modified models use sigmoid function (logistic function) for solving multiclass classification problem. There are many models in this category such as stochastic gradient descent [26], logistic regression and passive aggressive classifiers [27]. Logistic regression is very effective and usually outperforms other linear models. Additionally, implementation concepts of logistic regression are used widely in deep learning when building neural network models.

2.3.1.6 Support Vector Machine

Support vector machine (SVM) [28] is basically a linear classifier that divides data points into two classes based on the gap located in the space between the data points of classes. There is another version of SVM for performing a non-linear classification by using kernel that maps data points to higher dimensional space. Efficiency of SVM is sensitive to the selected value of *C* parameter, and to which kernel is used. *C* parameter calibrates the margin size between the hyperplane (which separates the data points) and the nearest data point in each class.

2.3.1.7 Deep Learning

Deep learning is a powerful machine learning technique that provides learning functions by using networks of multiple layers. Deep learning has become popular nowadays and has been employed in many research areas. This interest has appeared after the decay in using traditional neural networks for about 20 years due to the current available powerful machines for high performance computing.

Neural networks [29] mimic working mechanisms of neurons located in human brain. It is dissimilar with logistic regression in adding non-linear layers between input and output layers. Thus, it can learn non-linear online (real-time) models. However, its classification accuracy may converge to suboptimal solution since its hidden layers have non-convex loss functions. It is also sensitive to feature scaling and a many parameters. To increase confidence of classification accuracy, we run the model many times with different random initializations.

There are numerous forms of neural networks that are proposed for deep learning. Convolutional neural network (CNN) [30] is a one of these forms which is employed basically in the field of computer vision by using convolutional and subsampling (pooling) layers. There are other forms of deep learning techniques such as recursive neural networks (RNN) [31], recurrent neural network [32], long short term memory (LSTM) [33] and gated recurrent unit (GRU) [34].

2.3.1.7.1 Word Embeddings

Word embeddings is a method for substituting each word in text by a numerical vector. This method has enabled researchers for applying standard machine learning methods to achieve numerous tasks of NLP. Word embeddings preserve similarity between similar words in meaning. Thereby, word embeddings convert words to vectors while the similarity between vectors mimics semantic similarity between words. These word embeddings have been used broadly in deep learning and play an important role in developing many NLP systems.

There are many forms of word embeddings and the most two effective ones are GloVe [35] and word2vec [36]. These word embeddings are formed by using different deep learning methods for learning vector representation of words. Word2vec embeddings are designed for the first time in 2013 by using unsupervised learning techniques developed by researchers at Google. GloVe is an abbreviation of global vectors for word representation and it differs from word2vec in that it is a count-based model while word2vec is a predictive model.

Typically, we needs to train word2vec and GloVe models from scratch to fit with own context. In this case, a large number of micro-blogs are required for generating accurate word embeddings. Alternatively, we can use pre-trained word embeddings available in the literature such as Polyglot [37] and fastText [38]. In our work, we used pre-trained word embeddings from different sources.

2.3.1.8 Pros and Cons of Supervised Learning

Using only learning-based technique for classifying sentiment polarities expressed in micro-blogs has a key advantage in enabling the classifier to learn automatically from all kinds of features [39]. Learning the classifier with various labeled data may lead to hit optimal results and usually gives high classification accuracy. However, we should take care of overfitting phenomena to enable the model for classifying unseen data when using it in a real environment.

In the same context, supervised learning techniques may work better with document classification since more feature attributes can be extracted from the document. While, using supervised learning classifiers with micro-blogs may provide worse results since each micro-blog contains little information. As a result of this, we cannot extract enough features from micro-blogs for training the model very well and may be the classification accuracy converges to local optimum solution.

On the other hand, preparing labeled data for training supervised learning models is a time consuming process since it is mainly achieved manually. Additionally, preparing labeled data manually may cause various errors and may generate biased data. Using automatic tools for labeling data may also generate incorrect labels.

2.3.2 Unsupervised Learning

Unsupervised learning technique uses only unlabeled data for training the model. The most famous topic in unsupervised learning belongs to data clustering. Data clustering methods are based on dividing data into groups (clusters) in which each group has similar properties. We tested efficiency of classifying opinions by using data clustering methods in comparison with other machine learning techniques. The next subsections describe some data clustering algorithms that are used in this dissertation work.

2.3.2.1 Birch

Birch [40] algorithm divides data by building a tree called the characteristic feature tree (CFT). Each node in CFT contains a number of subclusters that have closed characteristics. The Birch algorithm has two parameters: the threshold and the branching factor. The branching factor is used to specify the number of subclusters in each node of CFT. The threshold parameter specifies the value of distance which is used for merging each data point with the closest subcluster.

2.3.2.2 K-means Clustering

K-means [41] algorithm uses a 2-phase iterative algorithm to minimize the sum of distances between each data point and centroid, summed over all *k* clusters. During the first phase, each iteration helps in reassigning data points to their nearest cluster centroid and recalculating cluster centroids. In the second phase, each data point is reassigned individually.

The K-means has 2 main advantages: it is very easy to implement and the time complexity is only O(*n*) (*n* is number of data points). Thus, K-means algorithm is suitable for clustering large datasets. On the other hand, K-means suffers from some disadvantages. The user has to specify the number of classes in advance. The performance of the algorithm is data-dependent and it depends on the initial conditions. This often leads K-means to converge to suboptimal solutions.

2.3.2.3 Pros and Cons of Unsupervised Learning

The main advantage of using unsupervised learning for sentiment classification is that we can use only unlabeled data when building the model. However, we cannot use most of features when building unsupervised learning model since we do not have enough information for applying them on unlabeled data. Additionally, unsupervised learning methods can be improved easily by modifying simple rules. However, we need more efforts for finding the best rules that enables the model to work very well.

2.3.3 Semi-supervised Learning

Semi-supervised learning technique uses labeled and unlabeled data in the training phase [42]. Semi-supervised learning is a special form of learning that addresses the problem of preparing labeling data. Semi-supervised learning uses large amount of unlabeled data points, together with the labeled data points, to build better classifiers. Unlabeled data points may be relatively easy to collect. The simplest semi-supervised learning technique uses unsupervised word representations as extra features with a supervised classifier [43]. Semi-supervised learning techniques include also self-training, co-training, multiview learning, and graph-based.

Using semi-supervised learning techniques add more beneficial characteristics for building sentiment analysis system. One of these characteristics is related to decreasing the need for annotating numerous micro-blogs when training sentiment analysis systems. Another characteristic we should consider in this work that supervised learning machines may cause overfitting, while using semi-supervised learning techniques may decrease the effect of this phenomenon. We selected some semi-supervised learning techniques to achieve our research goals. The next subsections describe theoretical background of these selected techniques.

2.3.3.1 Label Propagation

Label propagation is based on an iterative method that propagates labels by detecting high density regions in unlabelled data. Their implementations are based on constructing a similarity graph over all data points in the dataset. Label propagation method is an improved version of k-Nearest-Neighbor (kNN) method developed for finding closer unlabelled data points that are similar to labeled data points [44]. There is another similar method called label spreading that uses additionally affinity matrix and soft clamping across labeled data points [45].

2.3.3.2 Semi-supervised K-means

Since K-means is sensitive to initializing the centroids, many techniques are proposed to decrease this effect. One of research direction is based on initializing the centroids from a little amount of labeled data points while the clustering process is applied as usual on unlabeled data [46] [47]. This direction makes K-means method mimics semi-supervised learning techniques.

2.3.3.3 Self-Training

Self-training (also called self-learning or self-labeling) is a well known technique used to learn from unlabeled data [48]. In this technique, we train a supervised learning model by using labeled data points. Then, we use the same model to detect sentiment polarities of the unlabeled data points. All unlabeled data points that generate high confidence predictions are added to the labeled data. After that, we learn again the supervised learning model with the bigger labeled data to increase the performance. This process should be repeated for many rounds to hit the best performance. There are many spatial cases of this technique such as semi-supervised text classification by using expectation maximization (EM) [49].

2.3.3.4 Quasi-Newton Semi-supervised Support Vector Machines

Quasi-Newton semi-supervised support vector machines (QN-S3VM) is an extended method of SVM to mimic a semi-supervised learning technique [50]. The QN-S3VM method deals with linear and non-linear kernels [51]. It uses Quasi-Newton optimization algorithm for finding local optimum solutions. It is also sensitive to setting some parameters initialized randomly.

2.3.3.5 Pros and Cons of Semi-supervised Learning

Semi-supervised learning techniques can merge advantages of supervised and unsupervised learning techniques. Using semi-supervised learning methods decreases the need for preparing labeled data since we can use partially labeled data [52]. Additionally, using both labeled and unlabeled data may decrease the effect of overfitting phenomena that is revealed with supervised learning techniques. As a result of this, semi-supervised learning methods may outperform other machine learning methods in some cases since they may avoid overfitting along with utilizing unlabeled data for improving the performance.

2.3.4 Cross Validation and Overfitting

Learning the classifier too much or testing its efficacy by using training data are inefficient strategies. Learning the classifier too much will make it fits only training data. Thereby, it will not be able to classify new unseen data. Testing the classifier by using same data that is applied during the training phase will provide high performance but it will fail to detect new unseen data points when solving real problems. This phenomenon is called ***overfitting*** and should be avoided for building more efficient classifiers.

One of the proposed solutions to avoid overfitting is a procedure called ***cross validation*** (CV for short). For applying this strategy, we divide the training data into *k* sets (*k* folds) while we use the testing data without any change for the final evaluation. During each round, the classifier is trained by using the *k*-1 folds and the trained classifier is validated on the remaining part of the data by evaluating the improvement in the performance. Finally, the overall performance is measured for the *k*-fold cross validation by calculating the average of all computed values.

2.3.5 Multiclass Strategies

There are some classifiers that are implemented specifically to solve problem of two classes which are called binary classifiers. To enable these classifiers for classifying more than two classes (multiclass), we need firstly to apply multiclass strategies [53]. The most two famous strategies are referred to as one-versus-rest (OvR) (called also one-versus-all) and one-versus-one (OvO).

For applying OvR strategy to classify three classes, we need to build three binary classifiers and select the outcome from one classifier when classifying each data point. Selection mechanism may be based on finding the classifier which provides the maximum classification confidence or using voting strategy. When building each classifier of the three binary classifiers, we select one class form the three original ones. While, the second class contains all data points of the other two classes.

Applying OvO strategy to solve the problem of classifying three classes needs also to build three binary classifiers. With each binary classifier, we select the first class as one of the three original classes. While, the second class contains only data points of one class of the other two classes. Selection mechanism may be used also with voting strategy or confidence property.

2.3.6 Dimensionality Reduction

Most real problems deal with high dimensional data and sometimes reducing the number of dimensions may improve performance of classifying the data. There are many methods for reducing number of dimensions (feature attributes) that represent each data point.

We used two methods of dimension reduction through our dissertation work. The first method is called principle component analysis (PCA) [54] which uses statistics to convert a set of correlated data points into a set of linearly uncorrelated values called principal components. The second method is called linear discriminant analysis (LDA) [55] which is based on Fisher's linear discriminant, pattern recognition and machine learning. LDA expresses each dependent variable as a linear combination of other features to separate classes of data points. LDA is used also as a linear classifier. LDA is related to analysis of variance (ANOVA) and regression analysis.

2.4 Open Domain Targeted Sentiment Analysis

This section presents a background to open domain targeted sentiment classification. There are mainly two scenarios for implementing open domain targeted sentiment classification. The first scenario consists of two subtasks: name entity recognitions (NER) and target-dependent sentiment classification. NER is used for detecting targets as entities in the micro-blog. While, target-dependent sentiment classification identifies sentiment towards detected targets.

The second scenario is based on detecting targets along with their sentiment polarities expressed in the micro-blog. This scenario is a sort of a structured prediction since it predicts two labels (target and sentiment) for each token (word) in the micro-blog. This scenario is implemented mainly by using a method that is referred to as *sequence labeling*. The next subsections describe the main subtopics that are related to open domain targeted sentiment classification. We also describe a Markovian SVM which has been used for the first time in our work for improving the performance of open domain targeted classification.

2.4.1 Name Entity Recognition

NER [56] is a main classification task in NLP which identifies named entities (such as name of person or organization) in readable text (such as micro-blog). The output of this operation is a categorization tag that describes each named entity. Open domain targeted sentiment analysis includes NER [57] task for identifying all named entities in the micro-blog. Then, the targeted topic is detected from the name entities.

2.4.2 Sequence Labeling

Since entity recognition deals with entities (elements) in the input sentence (such as micro-blog), the research direction is shifted from sentence level into word level. Thereby, we need to deal with a sequence of words that form each sentence. The most famous method used for classifying sequence of words is called *sequence labeling*. Sequence labeling [58] is used broadly in NLP for classifying each word in the input text.

Open domain targeted sentiment classification is based on representing each micro-blog (such as tweet) as a sentence of words (tokens). Then, sequence labeling is used for identifying all words that are related to names such as persons, organizations, etc. The typical way to implement sequence labeling problem is called BIO tagging. Each token is labeled as "B" (beginning) tag if it is the first element in the named entity, or it is labeled as "I" (inside) tag if it is a subsequent token in the named entity, otherwise the token will be tagged as "O" (outside) tag. We can use other encoding strategy with sequence labeling but BIO tagging is the most famous one and it is used also with open domain targeted sentiment classification.

There are three techniques that can be used for applying sequence labeling to open domain targeted sentiment classification. The first one converts the problem into a traditional classifying method by using BIO encoding. Thereby, we can use any classifier such as SVM. The second technique uses neural networks for building the model of open domain targeted sentiment classification. The third technique uses hidden Markov models such as HMM and CRF for building the model of open domain targeted sentiment classification.

2.4.3 Sequence Tagging with Structural Support Vector Machines

In this research work, we employ sequence tagging with structural support vector machines for developing open domain targeted sentiment classification. To the best of our knowledge, this technique has not been used before in this direction. Hidden Markov support vector machine [59] is a famous method used for sequence tagging with structural support vector machines. This method combines hidden Markov model with SVM to build a model for sequence labeling. We specifically use this method for improving the performance of open domain targeted sentiment classification.

CHAPTER 3

LITERATURE REVIEW

This chapter presents prominent related work and analyzes their strengths and shortcomings. The chapter concludes by identifying the gaps to be pursued in future work. Those gaps are used to formulate the research problem addressed in this dissertation.

3.1 Sentiment Analysis Feature Engineering Techniques

Features engineering is the most important phase in sentiment analysis since performance of classification technique depends on the used features. Various features have been proposed to improve performance of sentiment analysis system. The three categories of features: syntactic, semantic, and stylistic are discussed in the sequel.

Syntactic features [60] are the most used features in sentiment analysis. For example, N-gram [61] is a famous feature that consists of a continuous sequence of N items from a given text. Also, Part-of-Speech is another popular syntactic feature. This type of features is based on ignoring unimportant parts and using certain parts of speech (text) such as adjectives. Moreover, frequency of marks and punctuation [62] is also used as a syntactic feature. This type of features is used broadly with sentiment analysis since it provides accurate results. For example, Sayfullina [63] used bigrams, emoticons, syntax, unigrams and other syntactic features for showing efficacy of his solution in reducing sparsity by using dimensionality reduction.

Semantic features are based on extracting semantic properties which are related to the meaning of specific words in the input text. Some used semantic features are based on context and domain [64] and other ones are based on using emoticons and hash-tags [65]. Furthermore, some famous semantic features are based on using polarity lexicon and others deal with negation. Some researchers employ only this type of features for improving performance of sentiment analysis [66] which highlights the importance of using these features.

There are various stylistic features used in literature such as frequency of letters, number of characters per word, inclusion of re-tweet, frequency of digits or special characters, and so on [67]. This type of features has not been used broadly with sentiment classification since these features do not add significant information when classifying the data points. Thus, researchers combine this type of features with other the two types for improving the performance [68].

3.2 Sentiment Classification Techniques

Various techniques have been proposed in literature to classify sentiment polarities [69]. Classification techniques use data point to identify sentiment polarity expressed in the corresponding text. These techniques can be combined together to improve sentiment classification [70][71]. Employing classification techniques for improving the performance of sentiment classification is discussed in the sequel. The classification techniques are categorized into three classes: supervised, unsupervised, and semi-supervised.

Various learning-based techniques are used for sentiment classification. For example, support vector machines (SVM) and Naive Bayes (NB) have been used broadly to create a model for identifying sentiment polarities [72]. Numerous improved techniques have been proposed for improving the performance of sentiment classification. For example, Go et al. [73] employed distant supervision to train a supervised learning classifier.

Some unsupervised learning techniques have been developed for improving performance of sentiment classification such as lexicon-based method. Lexicon-based method is based mainly on a corpus or dictionary. These methods classify directly each data point by using a dictionary of words without training the model. For example, Thelwall et al. [74] proposed a lexicon-based technique called SentiStrength that assigns a sentiment polarity and strength level to the input text. Another example, Kumar and Sebastian [75] proposed a method to calculate a sentiment score for a tweet based on a sentiment lexicon.

The state of the art has recently shifted toward proposing novel semi-supervised learning techniques [76]. Some techniques, such as incorporating word embeddings to represent the context of words and concepts, have been proposed to train sentiment classifier by providing weakly supervised mechanisms [76]. Other techniques involve building models such as LCCT [77] for detecting sentiment polarities in tweets.

3.3 Sentiment Analysis on Social Media

There are many directions used in the literature for developing sentiment analysis systems. We designed some criteria for categorizing research works through presenting a comprehensive literature review and building a comparison framework. To the best of our knowledge, our literature review is the first work that includes these criteria together.

The included criteria are based on classification techniques, feature attributes, data source, data language, and whether the used dataset is public or collected. We also include a criterion that reflects the measure used to evaluate performance of proposed technique. Our review revealed that most of the research works use accuracy and F1-score to measure the performance. Moreover, additional criteria are designed for dealing with recent directions such as implementing real-time system, using emoticons (emotion icons) included in micro-blogs for increasing the performance, developing target-dependent sentiment classification system, reducing sparsity in the dataset, using external syntactic analyzer, and dealing with open domain targeted sentiment classification.

The real-time [78] technique is employed for enabling systems to work dynamically with limited resources. Regarding using emoticons, most prominent classifiers proposed in the literature filter emoticons as noise. However, some studies use emoticons to collect more information about sentiments expressed in micro-blogs which leads to improve the performance of identifying sentiment polarities.

Target-dependent sentiment classification performs a finer-grained analysis and improves the performance by using an aspect level of opinions [79]. Accordingly, we designed another criterion for categorizing the research works based on employing target-dependent sentiment classification. Table 3.1 summarizes different research works based on our designed criteria. It is worth noting here that the table only discusses what we believe to be prominent work in this research area. Other work has also been surveyed but not reflected in this table [80][81][82][83][84][85][86][87][88][89][90][91][92].

We observe from Table 3.1 that there are many gaps which have not been addressed yet by researchers. Some research works also focused on specific aspects rather than classifying data such as dealing with real-time aspect. We selected some gaps to be investigated in our thesis such as employing semi-supervised learning techniques for developing target-dependent sentiment classification by using partially labeled data.

On another aspect, we noticed that most of research works are reported with classification accuracy in the range of 60%-70%. Even the highest result (between 90% and 100%) is restricted to a special collected data with specific feature and technique. The results also are sensitive to setting some parameters while the table illustrates only the best values and ignores the average of all experimental results. Thus, the results provided by using methods proposed for achieving same research goals are still limited since the experimental environments are different. The low performance of some research works introduces a motivation that there is still more work that can be done in this research area.

3.4 Target-dependent Sentiment Analysis on Social Media

Recent studies have been conducted to develop target-dependent sentiment analysis systems. Dong et al. [93] integrated target information with recursive neural network for employing the strength of deep learning. While, Changqin Quan and Fuji Ren [94] proposed a similarity based approach to provide more fine grained sentiment analysis.

Some effective features are extracted [95] and employed to build classifiers by using supervised classification technique. Duy-Tin Vo and Yue Zhang [96] proposed a technique that does not use external syntactic analyzers, by leveraging distributed word representations and rich automatic features. We extended our literature review in the previous section by focusing on summarizing research works that deal specifically with target-dependent sentiment analysis as illustrated in Table 3.2.

We categorized the selected research works by illustrating whether they are using external syntactic analyzer. Developing a system that does not depend on external syntactic analyzer improves the performance of classifying micro-blogs regardless with which language they are used. However, decreasing the dependency on the external syntactic analyzer is a more challenging scenario. We also categorized the works by illustrating whether they deal with open domain targeted sentiment classification. Our literature review shows that developing open domain targeted sentiment system without using external analyzer is still an open issue and limited works are done in this direction.

Table 3.1: Summary of various sentiment analysis works.

Reference	Techniques			Features			Real-Time	Emotic	Target Depend	SPAR	Data		Lang	Data Sour	Acc (%) ⚙	F1-score (%) ⚙
	Supr	Semi	Unsu	Syn	Sem	Sty					Pub	Col				
Hu et al. [97]			AF	√				√			√		EN	TW	74.71	
Jiang et al. [98]	SVM			√					√	√		√	EN	TW	68.30	71.0
Mohammad et al. [99]	SVM			√	√			√		√		√	EN	TW		88.93
Bosch [100]		IR		√								√	EN	TW	95.33	
Duarte [101]			SC	√			√	√				√	PO	TW		88.93
Fang et al [102]	NS			√								√	EN	AM		94.0
Smailovic [103]	SK		LB	√			√				√		MU	TW	81.42	81.57
Bifet et al. [78]	NH						√					√	EN	TW	82.80	
Suchdev et al. [104]	√		NB	√							√		EN	TW	100♦	
Sayfullina [63]	EN			V						M	√		EN	TW	60.93	
Kiritchenko et al [105]	SVM			√	√					C	√		IE	TW		89.50
Narr et al. [106]	NB	E		N				√		√		√	IN	TW	81.30	
Gebremeskel [107]	BS		NB	√				√				√	EN	TW	90.79	
Gunther [108]	NP	ST		√							√		EN	TW	86.07	85.91
Valkanas et al. [109]			LB				√					√	EN	TW	N/A ♠	
Scholar et al. [110]	DS		KC	√		√						√	EN	TW	72.33	
Zhao et al. [111]	NB			√			√	√				√	CH	WE	N/A ♠	
Cimino et al. [112]	SVM			√									IT	TW		69.80
Batool et al. [113]			NB	√								√	EN	TW	89.50	
Saif et al. [66]	NB				√							√	EN	TW		85.60
Bakliwal et al. [114]		SS		√							√		EN	TW	88.40	
Refaee et al. [68]	SM			√	√	√	√					√	AR	TW	87.74	88.0
Duwairi et al. [115]	SK							√				√	AD	TW	76.78	
Abdul-Mageed [116]	SVM			√								√	AD	TW	70.30	88.89
Ibrahim et al. [117]	SVM			√								√	AD	TW	100♦	100♦
El-Makky et al. [118]			LS		√							√	EG	TW	84.0	89.0
Al-Kabi et al. [119]			LB	√								√	CR	TW	96.90	
Hamouda et al. [120]	ND			√		√						√	AR	FA	73.40	
Shoukry [121]	SVM		SC	√		√						√	EG	TW	79.35	80.60
Ahmed [122]	ML			√	√	√			√			√	AR	TW	92.54	
Yang et al. [123]		LC			√						√		EC	TW	81.50	

Supr (Supervised): SVM= Support Vector Machines, **NB**= Naive Bayes, **NS=**NB, Random Forest, and SVM, **SK**= SVM, NB, and k-Nearest Neighbors, **NH**=NB, stochastic gradient descent, and the Hoeding tree, **EN**= Extreme Learning Machine and NB, **BS=** NB and SVM, **NP**= NB, Perceptron and SVM, **DS=**Decision Trees and SVM, **SM=** SVM and majority baseline, **ND=** NB, Decision Tree, and SVM, **ML=**Simple Linear Regression classifier, SVM, NB, PART, AdaBoost, and J48 tree. ♠ **This work focused on real-time aspect rather than classifying data.**
Semi (Semi-supervised): E=semi emoticon heuristic, **ST**=self-learning, and tri-learning, **IR=** incremental learning and random-walk algorithm, **SS=**semi-supervised SVM, **LC**= Lexicon-based and Corpus-based Co-Training. **Dataset: Pub=**Public, **Col**=Collected.
Unsu (Unsupervised): AF=unsupervised analysis framework, **NB=** knowledge-based unsupervised, **LB=**lexicon-based, **KC**=K-means, **SC**= semantic, **LS**= hybrid (LB+ SC). ♦ **Reported result for a specific dataset. Sem=**Semantic. **Syn (Syntactic): V**= bigrams, topics, slang words, booster words, negations, prepositions, emoticons, syntax, celebrities, places, links, and unigrams, **N**=n-gram. **Sty=**Stylistic.
SPAR (Reduce Sparsity): M= Least Angle Regression, Information Gain, Mutual Information, and tf-idf score, **C**= token clustering.
Lang (Languages): EN= English, **PO**= Portuguese, **MU**= Multiple, **IE** = Informal English, **IN**= Independent, **CH**=Chinese, **IT**= Italian, **BR**= Brazilian, **AR**= Arabic, **AD**= Arabic Dialects, **EG**=Egyptian, **CR**=Colloquial Arabic, **EC**= English and Chinese. **Data Source: TW**=Twitter, **AM**=Amazon, **WE**=Weibo, **FA**=Facebook. ۞ **These are the best results reported by researchers.**

Table 3.2: Summary of various target-dependent sentiment analysis works.

Reference	Tech	Features			Open Domain	Use Syntactic Analyzer	Real-Time	Dataset		Lang	Data Source	Acc ۞	F1-score (%) ۞
	Sup	Synt	Sema	Styl				Pub	Col				
Jiang et al. [98]	SVM	√				√			√	EN	TW	68.30	71.0
Ahmed [122]	ML	√	√	√		√			√	AR	TW	92.54	
Mitchell et al. [124]	CRF	√	√		√				√♠	ES	TW	89.50*	
Dong et al. [93]	NN	√	√			√			√♯	EN	TW	66.30	65.90
Zhang et al. [125]	NN	√	√		√			√♠		ES	TW	N/A	65.76/43.04■
Vo et al. [96]	SVM		√					√♯		EN	TW	71.10	69.9
Wang et al. [126]	RC	√				√	√		√	CH	NT	84.80	85.0
Tang et al.[127]	LS		√					√♯		EN	TW	71.50	69.50
Zhang et al. [128]	GN	√	√					√		EN	TW	71.96	69.55
Farra et al. [129]	CRF	√	√		√	√		√		AR	AL	76.0	61.8/44.2■
Li et al. [130]	SS	√	√		√			√♠		ES	TW	N/A	68.45/43.55■
Jabreel et al.[131]	Bi-GRU	√	√					√♯		EN	TW	72.25	70.47

Sup (Supervised): SVM= Support Vector Machines, **ML**=Simple Linear Regression classifier, SVM, Naive Bayes, PART, AdaBoost, and J48 tree, **NN**=Neural Networks, **CRF**= Conditional Random Fields, **RC**=clustering data for selecting training data (labeled manually) for NN, **LS=** Long Short-Term Memory, **GN**= Gated Neural Networks, **SS**= Sentiment Scope model, **Bi-GRU**= Bi-directional Gated Recurrent Unit.
۞ **These are the best reported results by researchers. Lang (Languages): EN**= English, **AR**= Arabic, **ES**= English and Spanish, **CH**=Chinese. **Dataset: Pub**=Public, **Col**=Collected. **Data Source: TW**=Twitter, **AL**=Aljazeera newspaper, **NT**=Tencent Weibo (Chinese micro-blogging website).
*** This result for classifying Spanish tweets and it is better than using English tweets. ■ These results are reported for entity recognition/sentiment analysis. {♯, ♠}=Same dataset.**

CHAPTER 4

RESEARCH PROBLEM AND SOLUTION APPROACH

In this chapter, we state the research problem that we address in this research. The goals and scope of this thesis along with a discussion of the methodology followed to achieve the stated goals are described in next sections.

4.1 Problem Statement

Considering the gaps identified by our literature review presented in Chapter 3, we explored the potential of using semi-supervised learning techniques in sentiment analysis. Our research focuses on open domain targeted sentiment classification which was revealed to have been not adequately investigated. A major objective is to improve the performance of open domain targeted sentiment classification. The corresponding hypotheses are:

1. *"Using semi-supervised techniques in identifying sentiment polarities will improve the performance of open domain targeted sentiment classification for micro-blogs in social media."*

2. *"There is no super classifier that can identify correctly all sentiment polarities expressed in micro-blogs."*

3. *"There is no statistical difference between different classifiers used for improving the performance of open domain targeted sentiment classification."*

Our research tests the first hypothesis by evaluating different semi-supervised techniques against many collected micro-blogs and examines their effectiveness in comparison with some prominent techniques. We also investigate in the second hypothesis improving the performance of open domain targeted sentiment classification. Our work is extended to see whether we can develop a super classifier the works better in all situations. Through our research work, we check as well the statistical differences among different proposed techniques to test the third hypothesis.

4.2 Motivation

Several reasons motivated us to select the open domain targeted sentiment classification and semi-supervised learning domains as the topic for this PhD study. The potential impact of this topic in academia and the commercial world is one of the major motives in conducting this study. Many applications [132] rely on sentiment analysis which means that any improvement in this area will result in an important impact on a large range of domains. There are also recent advances [133] in semi-supervised learning domain that can be employed to improve the performance of sentiment analysis systems. In addition, the limited work conducted on open domain targeted sentiment classification motivated us to employ more machine learning techniques in this direction.

4.3 Research Objectives

Most of the studies reported in the literature have considered sentiment analysis within the context of target independent classification. To the best of our knowledge, the problem of open domain targeted sentiment classification using a mix of labeled and unlabeled data has not been addressed before. Consequently, we set the main goal of our research effort to investigate semi-supervised learning techniques to maximize efficiency of open domain targeted sentiment classification.

Additionally, we set a goal for developing a context-based analysis system that identifies the context of a given set of micro-blogs and analyzes their sentiments within that context. Another goal is set to develop new techniques that are capable of improving the performance of open domain targeted sentiment classification. In order to achieve these goals, the following objectives are set:

RO1. Complete the comparison framework for open domain targeted sentiment classification.

RO2. Develop a context-based analysis system for open domain targeted sentiment classification.

RO3. Develop a semi-supervised learning approach for open domain targeted sentiment classification.

RO4. Develop and validate techniques for improving the performance of open domain targeted sentiment classification.

4.4 Research Approach

This section describes the framework which guides our work to achieve the research objectives. This section describes also the measures that are used for evaluating our work.

4.4.1 Research Framework

In this research work, we develop an open domain targeted sentiment classification technique that does not use external syntactic analyzer. The framework of our research is illustrated in Figure 4.1. It is inspired by the work of Vo and Zhang [96]. Initially, the name entities (topics or targets) are detected in the tweet, and then the sentiment polarity towards each target is identified by extracting a rich set of features.

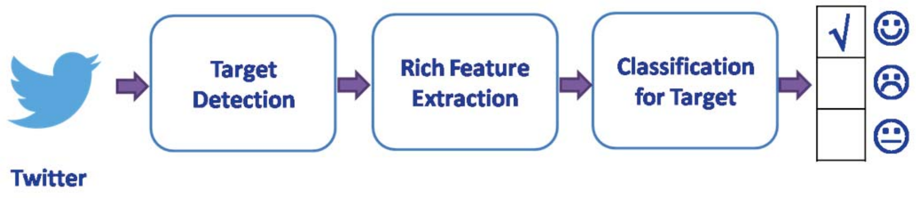


Figure 4.1: Framework for open domain targeted sentiment classification.

We explore efficacy of using different methods for extracting a set of rich features such as using neural pooling functions and word embeddings. We also investigate efficacy of using traditional features such as lexicon, n-gram, and part of speech (POS) [106]. Feature extraction methods are affected by the target (topic) included in the micro-blog. Thus, we need to adapt the extracted feature attributes to be sensitive to the included target. This issue is not mutilated in target-independent systems.

Figure 4.2 shows an example which is also inspired by the work of Vo and Zhang [96]. The figure shows how sentiment polarity is identified by using target-independent system. The system classified the input tweet as positive sentiment since all words in the tweet are positive. It is clear from the figure that the target-independent system did not consider the presence of more than one target (Windows and iOS).

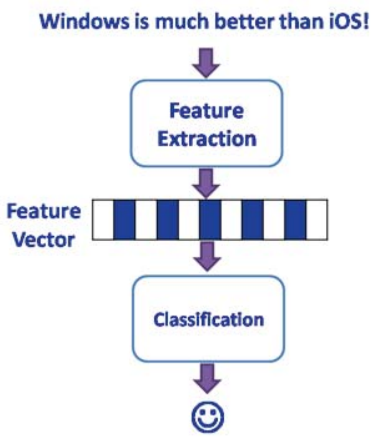


Figure 4.2: Example of sentiment analysis in target-independent system.

Figure 4.3 shows how sentiment polarity is identified by using target-dependent system. It is clear from the figure that feature attributes is modified based on the selected target. The output of the same input tweet (used by Figure 4.2) will be positive if the target is "Windows" and its corresponding output will be changed to negative if the requested target is "iOS".

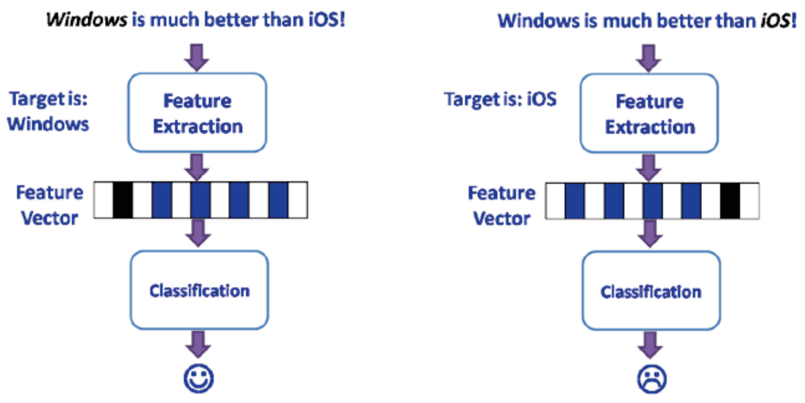


Figure 4.3: Example of sentiment analysis in target-dependent system.

4.4.2 Research Directions

In this research work, we develop a system for context-based target-dependent sentiment analysis. The lower level of our research work is based on developing new solutions in two main directions:

1. We explore some advanced semi-supervised learning techniques that are proposed for solving general classification problem and adapt them for developing target-dependent sentiment classification with partially labeled data. To the best of our knowledge, our work is the first research that employs semi-supervised learning techniques for target-dependent sentiment classification. We also propose new solutions for improving the performance of target-dependent sentiment classification.

2. We developed a new technique for applying open domain target-dependent classification by using partially labeled data. To the best of our knowledge, our work is the first research that uses partially labeled data with open domain targeted sentiment classification. We also propose new solutions that use fully labeled data for improving the performance of open domain targeted sentiment classification. As a result of this, our research work has filled some gaps that are described in Section 3.4.

4.5 Evaluation Measures

Empirical results obtained from experiments provide a good way to evaluate performance of both target-dependent and open domain targeted sentiment classification. This section describes the measures that are used to assess our proposed solutions. We use accuracy, precision, recall, specificity and F1-score [134] for evaluating tasks of sentiment classification and NER. Other evaluating measures have been specifically used for evaluating the performance of open domain targeted sentiment classification.

Accuracy

The accuracy is the ratio of all samples (such as micro-blogs) that are classified correctly. We can simply calculate it by using the following formula:

$$\text{Accuracy} = \frac{\text{Correclty\_Classified\_Samples}}{\text{All\_Samples}} \times 100\% \tag{4.1}$$

Precision

Precision is the ratio of samples which are correctly classified as positive to all samples classified as positive. Next formula is used to calculate it:

$$\text{Precision} = \frac{\text{True\_Positive\_Samples}}{\text{True\_Positive\_Samples} + \text{False\_Positive\_Samples}} \times 100\% \tag{4.2}$$

Recall

Recall (which also known as sensitivity or true positive rate) is the ratio of samples which are classified correctly as positive to all positive samples.

$$\text{Recall} = \frac{\text{True\_Positive\_Samples}}{\text{True\_Positive\_Samples} + \text{False\_Negative\_Samples}} \times 100\% \tag{4.3}$$

Specificity

Specificity (which also known as true negative rate) is the ratio of samples which are classified correctly as negative to all negative samples.

$$\text{Specificity} = \frac{\text{True\_Negative\_Samples}}{\text{False\_Positive\_Samples} + \text{True\_Negative\_Samples}} \times 100\% \tag{4.4}$$

F1-score

The F1-score (also known as F-score or F-measure) is the harmonic mean of precision and recall, and its best value is 1 while the worst score is 0. It is calculated as:

$$F1 - score = 2 \times \frac{\mathrm{Pr}\,ecision \times \mathrm{Re}\,call}{\mathrm{Pr}\,ecision + \mathrm{Re}\,call} \qquad (4.5)$$

The F1-score is basically used with binary classification and there are different modifications [135] to use it with multiclass classification such as the macro-average F1-score, and the micro -average F1 score.

The macro-average F1-score is straight forward. It is calculated by taking the average of the precision and recall of the system on different sets. While, each set is generated by using binary classifier applied to two selected classes. In micro-average F1-score, we firstly calculate the individual true positives, true negatives, false positives, and false negatives of each set. Then, we use the sum of these values to find the micro-average precision and the micro-average recall. Finally, the micro-average F1-score will be the harmonic mean of the micro-average precision and the micro-average recall. We use macro-average method for studying how the system performs across overall sets of data. Micro-average method can be used when dataset varies in size to come up with a specific decision.

Acc-all

This measure [124] is used specifically with open domain targeted sentiment classification. It measures the accuracy of the entire named entity span along with the sentiment span. It primarily measures the correctness of O labels.

Acc-Bsent

We use this measure [124] to evaluate specifically performance of open domain targeted sentiment classification. It measures accuracy of identifying the start of a named entity (B-labels) along with the sentiment expressed towards it. Thus, it focuses only on the beginning of named entities.

Zero/one-error

The zero/one-error calculates percentage of micro-blogs that had at least one misclassified tag. We use this measure to validate efficacy of our proposed solution in improving performance of open domain targeted sentiment classification.

CHAPTER 5

EXPERIMENT DESIGN

This chapter describes the experiment environment used for conducting numerous experiments to show performance of both target-dependent and open domain targeted sentiment classification. We also describe the used datasets and analyzing them. The chapter presents also feature engineering methods used in this work.

5.1 Datasets

In this section, we describe the characteristics of all datasets that are used for conducting our experiments. We used different datasets for showing the performance of our solutions proposed for both target-dependent and open domain targeted sentiment classification. Table 5.1 shows a summary of all datasets used in this research work.

Table 5.1: Summary of Datasets used for conducting experiment work.

Name	Source	Language	Original Use	Experiment Use
DatasetA	Dong et al. [93].	English	T	T/O
DatasetB	Mitchell et al. [124]	English/Spanish	O	O
DatasetC	Zhang et al. [128]	English	T	O
DatasetD	El-Kilany et al. [136]	Arabic	O	O

Used for: **T**= target-dependent sentiment. **O=** open domain targeted sentiment

We conducted experiments for target-dependent sentiment classification on a popular dataset (DatasetA) that is compiled by Dong et al. [93]. The dataset consists of 6248 tweets for training and 692 tweets for testing. The distribution of sentiment polarities of micro-blog (in both training and testing data) is 25% are positive tweets, 25% are negative tweets, and the rest 50% are neutral tweets. The assigned labels which are used in the dataset are 2 for representing positive tweets, 0 for representing negative tweets, and 1 for representing neutral tweets.

Our experiments conducted for showing the performance of open domain targeted sentiment classification are applied to two datasets. The first dataset (DatasetB) is collected originally by Mitchell et al. [124] which is available publically[1]. This dataset is used also by other researchers [125][130]. Thus, using this dataset enables us to make real comparisons with previous related works. The dataset includes both English and Spanish tweets where each word (token) is located in a separated line. Table 5.2 shows statistics of the dataset as described by Zhang et al. [125]. The second dataset (DatasetC) used in this research direction is collected by Zhang et al. [128]. Table 5.3 shows statistics of the dataset as described in their research work. Both datasets consists of 10 folds and each fold is divided into training, testing, and development sets. Additionally, we used the training set (1999 tweets) of dataset (DatasetD) that is collected by [136] for showing performance of applying our proposed context-based system to Arabic language.

Table 5.2: Statistics of DatasetB used for open domain targeted sentiment classification.

Domain	**#Sent**	**#Entities**	**#+**	**#-**	**#0**
English	2,350	3,288	707	275	2,306
Spanish	5,145	6,658	1,555	1,007	4,096

Table 5.3: Statistics of DatasetC used for open domain targeted sentiment classification.

	#Targets	**#+**	**#-**	**#0**
Training	9,489	2,416	2,384	4,689
development	1,036	255	272	509
Testing	1,170	294	295	581

[1] http://www.m-mitchell.com/code/index.html

5.2 Feature Engineering

We use the code[2] provided by Vo and Zhang [96] to extract feature attributes for target-dependent sentiment classification. The code generates word2vec embeddings[3] that are suitable for target-dependent sentiment classification. Word2vec embeddings represents each word of micro-blog in a lexicon by using low dimensional vector. Words with similar meanings have vectors with close values that reflect the distance with the other words of different meanings.

We used same discrete features that are generated by Mitchell et al. [124] and used by Zhang et al. [125] and Li et al. [130] for conducting experimental work of open domain targeted sentiment classification. These discrete features are shown in Table 5.4. We used these features form the implementation code provided by Zhang et al. [125]. Additionally we used continuous features (word2vec embeddings) that are generated also by Zhang et al. [125].

It is interesting to clarify that number of discrete features in English data equals 10717. While, number of discrete features in Spanish data equals 20033 features. Moreover, as illustrated in Table 5.2, data of Spanish language is larger than data of English language. Thus, we expect that using Spanish data will provide more accurate results since it will train the used machine learning models much better in comparison with using English data.

[2]*https://github.com/duytinvo/ijcai2015*

[3] *https://code.google.com/p/word2vec*

Table 5.4: Discrete features used for open domain targeted sentiment classification.

Surface Features
binned word length, message length, sentence position; Jerboa features; word identity; word lengthening; punctuation characters, has digit; has dash; is lower case; is 3 or 4 letters; first letter capitalized; more than one letter capitalized, etc.
Linguistic Features
function words; can syllabify; curse words; laugh words; words for good/bad; slang words; abbreviations; intensiers; subjective suffixes and prefixes (such as diminutive forms); common verb endings; common noun endings
Brown Clustering Features
cluster at length 3; cluster at length 5
Sentiment Lexicon Features
is sentiment-bearing word; prior sentiment polarity

5.3 Data Enhancement

We noticed that 1st fold is missed (it is a copy of 2nd fold) in DatasetB. Thus, we developed a code to find feature vectors of missing words. We generated all files of the 1st fold (testing, training, and development sets) to complete the dataset and conducting our experiments.

To achieve our goal, we used testing and training sets of 1st fold that are included in the original dataset. We used the dataset which is provided by Mitchell et al. [124]. Since data included by Zhang et al. [125] splits original training data provided by Mitchell et al. into training and development sets, we faced a problem in determining number of tweets in development set of 1st fold. To solve this problem, we used number of tweets in training data equals 1903 tweets which is equal to number of tweets that are used in training set of 2nd fold.

Thereby, we selected the first 1903 tweets from the original training set of 1st fold provided by Mitchell et al. [124] as training set of 1st fold in our dataset. While, the rest of tweets (212 tweets) in the original training set of 1st fold (provided by Mitchell et al.)

are used as development set of 1st fold in our dataset. Then, we found all corresponding discrete and continuous features for preparing all sets of 1st fold.

5.4 Data Scaling

The original data of DatasetA suffered from scaling problem where feature attributes that represent each data point are generated by using different scales. Based on our experiment results, we noticed that the result of using unscaled data is bad and the accuracy achieved did not exceed 50%. Scaling the data (feature scaling) would make all feature attributes with the same scale and range [137]. Scaling the data increases classification accuracy since it increases distances between data points in high dimensional space which helps in separating classes more efficiently. Thus, we scaled all training and testing data using the *LibLinear* library[4]. Scaling the data by using this library has also reduced number of feature attributes from 3600 into 3450 since the values of the removed 150 attributs were too close to zero.

5.5 Data Visualization

It is important to get a view of the topology of the dataset for interpreting the behavior of applied machine learning techniques and improving the performance. Since the dataset includes *n*-dimensional data points, we cannot visualize the data in a convenient 2D plot. In an attempt to get a bird-eye view on the classification of the tweets in two dimensions, we used two methods of dimensionality reduction: principle component analysis (PCA) and linear discriminant analysis (LDA). These methods are applied to DatasetA for interpreting the behavior of target-dependent sentiment classification.

[4] http://www.csie.ntu.edu.tw/~cjlin/liblinear/

Figure 5.1 and Figure 5.2 visualize training and testing data after reducing number of dimensions by using PCA. We can note from the figures that the three sentimental classes (negative, positive, and neutral) have complex shapes and they are too close to each other. These figures illustrate clearly that classifying this dataset is not an easy task. Applying PCA reduction did not provide well results because working mechanism of PCA is based only on using unlabeled data which decreases ability to separate the three classes during the process of reducing dimensions.

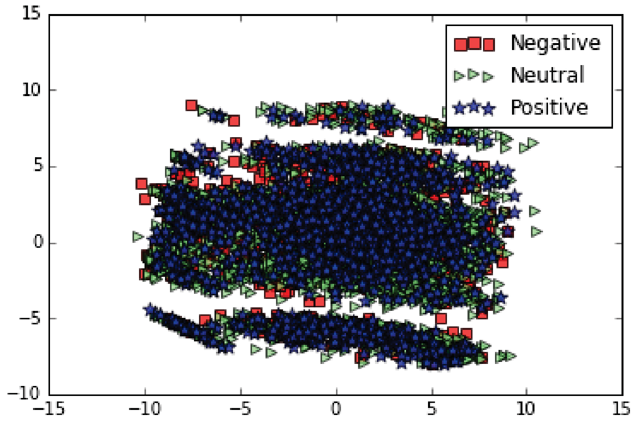


Figure 5.1: Visualizing training data reduced by PCA.

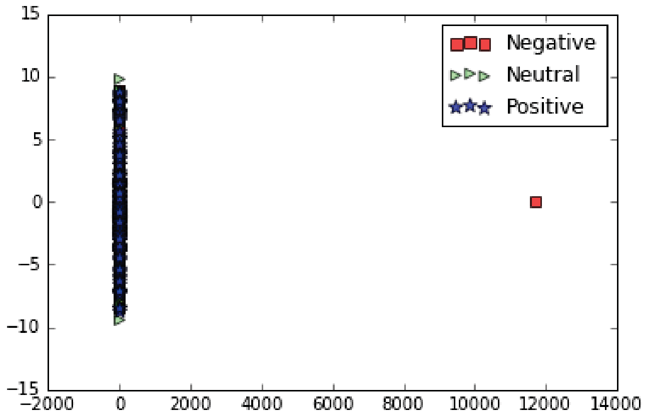


Figure 5.2: Visualizing testing data reduced by PCA.

Figure 5.3 and Figure 5.4 visualize training and testing data after applying LDA for reducing number of dimensions. The figures show clearly shapes of the three classes.

LDA provides more clear results because it builds a learning model based on the actual labels. The figures show that the classes are connected with each other. It is clear also that some data points cannot be classified easily by using a simple classifier. Moreover, the shapes of the three classes are more complicated in testing data.

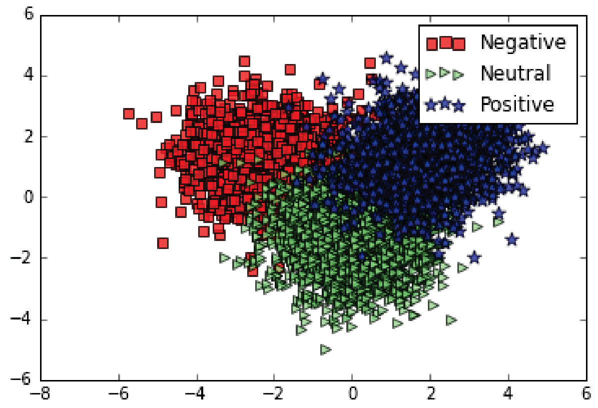


Figure 5.3: Visualizing training data reduced by LDA.

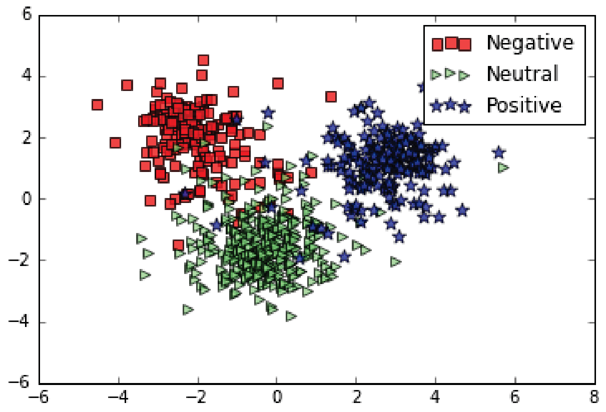


Figure 5.4: Visualizing testing data reduced by LDA.

5.6 Experiment Setup

We used some machines and tools for conducting many experiments in this research work. The development tools and hardware platform specification are described in Table 5.5 and Table 5.6 respectively. We use same tools for conducting experiment work for both target-dependent and open domain targeted sentiment classification. We used same machine (Machine A) for conducting most of experiments. Additionally, we used specifically a machine with limited specification (Machine B) for adding more challenges when applying some deep learning methods.

Table 5.5: Tools and programs.

Tool	Version	Purpose
Python	2.7	Extracting Features, building and learning models for developing experiments, classifying micro-blogs, and computing results.
Anaconda	4.2.0	Open data science platform powered by Python for providing an environment that facilitates developing our experiments.
Spyder	2.3.8	Graphical platform for editing, testing and debugging Python codes.
LibLinear	2.1	Scaling and learning data for building SVM models.
QN-S3VM	2012	Building and learning semi-supervised SVM models.
MS Excel	2016	Analyzing data and plotting graphs.
Kutools	16.50	A Powerful tool used for MS Excel that helping in performing quickly time-consuming operations.
Minitab	18.1.0	Analyzing data and plotting graphs.
Vim	7.4	Text editor for editing huge data files.

Table 5.6: Platform specifications.

Component	Machine A	Machine B	Virtual Machine
CPU	Intel(R) Core (TM) i7-3720 3.40 GHZ	Intel(R) Celeron(R) 1.6GHZ	Intel(R) Core (TM) i7-3720 3.40 GHZ
Memory	8.00 GB	4.00 GB	2.00 GB
OS	Windows 8 (64-bit)	Windows 10 (64-bit)	Upuntu 14.0 (32-bit)

Moreover, we developed some experiments for open domain targeted sentiment classification by using virtual machine with specifications that are included in Table 5.6. We installed on the virtual machine some of tools that are included in Table 5.5 for developing more experiments. Moreover, we conducted some experiments by using high performance computing (HPC) account[5] that is provided by KFUPM. We used HPC account because some experiments need huge memory for training the models specifically when using Spanish dataset.

[5] http://hpc.kfupm.edu.sa

5.7 Conclusion

This chapter describes the datasets that are used for conducting our experiment work. The chapter presents also the features that are used in this work. We analyzed the used datasets by using data visualization methods. Based on our analysis of experiment results, we deduced different conclusions. We conclude that the dataset used with target-dependent sentiment classification should be normalized firstly before applying any machine learning technique. Thus, we normalized (scaled) data to the same scale for improving the performance.

We conclude also that using LDA is better than using PCA when reducing number of feature attributes because PCA is based on employing unlabeled data which mimics unsupervised learning technique. While, LDA learns the model by using actual labeled data before applying dimension reduction.

CHAPTER 6

TARGET-DEPENDENT SENTIMENT ANALYSIS

Recent studies showed that target-dependent sentiment analysis increases significantly classification accuracy. In this chapter, we present different solutions for improving the performance of target-dependent sentiment classification by employing numerous machine learning techniques. We successfully employed semi-supervised learning techniques to improve the performance of target-dependent sentiment classification in comparison with employing other supervised and unsupervised learning techniques.

Additionally, we propose new semi-supervised learning technique (which meets research objective RO3 in our dissertation) to decrease the need for using labeled data when training models of target-dependent sentiment classification. Our semi-supervised learning solution provides a comparable performance to pervious supervised learning techniques that are proposed in the state of the art. Moreover, we present new supervised learning solutions (which meets research objective RO4 in our dissertation) for improving the performance of target-dependent sentiment classification. The proposed solution improves the performance in comparison with pervious prominent work.

6.1 The Approach

Our goal in this research direction is based on developing a system for target-dependent sentiment classification that can to be applied easily to any language (language independent). To achieve this goal, we work on developing sentiment classification system that does not use an external analyzer. Getting rid of using external analyzer enables us to develop a language independent system. Thus, we did not need to use any NLP tools for designing external analyzer when building the developed system.

Additionally, we did not use sentiment lexicons [138] as exploited in traditional systems. Such traditional systems are meant to detect sentiment polarity expressed in the micro-blog by summing sentimental scores that represent included words (tokens). These scores are extracted directly from the sentiment lexicons. Thus, the traditional system depends mainly on these sentiment lexicons. As a result of this, using sentiment lexicons decreases possibility of developing language independent system. Thereby, we avoid using these sentiment lexicons when building the proposed system.

Recent studies use word embeddings to map each word in input text into a vector of numerical values. This method generates close values to words that have similar meanings and forms longer distances to represent words with different meanings. We depend on this method for extracting feature attributes since this method is flexible and can be applied easily to any language. Our work is based on using a famous form of word embeddings called word2vec [139]. We use specifically a word2vec embeddings that are designed for target-dependent sentiment classification. As a result of this, our proposed system converts each input micro-blog into one vector of numerical values. We refer to this resulted vector as a data point. Then, the data points are classified into positive, negative or neutral sentiments.

6.2 Proposed Solutions

We propose novel techniques for improving the performance of target-dependent sentiment classification. Some of proposed techniques are based on employing semi-supervised learning for increasing classification accuracy with partially labeled data. The other proposed techniques improve the performance of target-dependent sentiment classification by using fully lapped data. We describe all the implementation details of the techniques in the sequel.

6.2.1 Performance Enhancement

We evaluated performance of applying numerous supervised and unsupervised learning methods that have not been used before in this research direction. We also enhanced working mechanism of some learning methods for improving the performance of target-dependent sentiment classification. For example, we suggested different mechanisms for initializing centriods of K-means method to improve its efficacy with target-dependent sentiment classification. Chapter 8 gives all relevant details. Additionally, we evaluated performance of employing some deep learning techniques that have not been used before for developing target-dependent sentiment classification. All relevant details are included in Chapter 7.

Moreover, we employed different semi-supervised learning techniques for developing target-dependent sentiment classification with partially labeled data. We added some contributions for improving efficacy of some existing semi-supervised learning methods in solving our research problem. As a result of this, enhancing the semi-supervised learning methods improved the performance of target-dependent sentiment classification with partially labeled data. Some of our contributions in this direction are described briefly in the sequel while all relevant details are included in Chapter 8.

We improved efficacy of applying QN-S3VM classifier for improving target-dependent sentiment classification. QN-S3VM has been proposed basically to solve the problem of binary classification (deals only with two classes). Thus, we used multiclass classification strategy to make this classifier suitable to deal with our classification problem that includes three classes (positive, negative, and neutral). We also modified the working mechanism of QN-S3VM by providing numerical values that measure distances from the decision boundary for each data point. Then, we used these values to apply multiclass strategies.

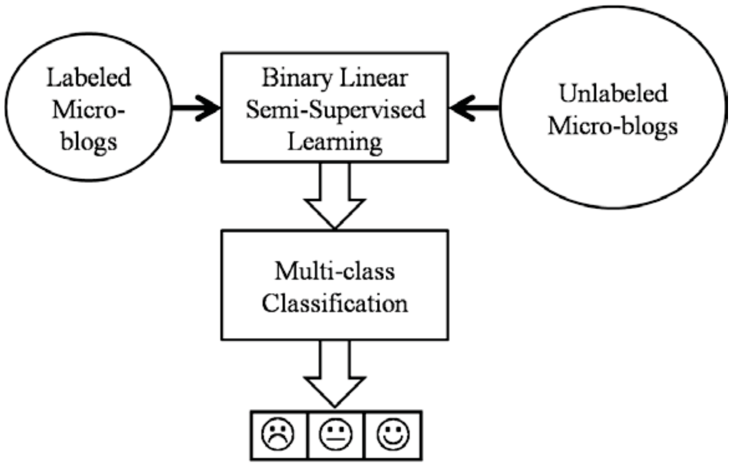
We also applied self-training technique by using different methods. One of these methods is based on calculating confidence by using a formula presented by S. Ravi [48]. We applied this formula to fit our classification problem that includes three classes (positive, negative, and neutral) and constructing a hyperplane decision boundary of three dimensions. This work adds the unlabeled data points to a labeled training set and removes them from the unlabeled set by using the following equation:

$$|d - \mu_y| < \sqrt{\tfrac{1}{1-\delta}}\sigma_y \tag{6.1}$$

where d is the distance between data point x and the decision boundary, y is the label of data point x according to the decision boundary, μ is mean of distances to the decision boundary, σ is the standard deviation of distances to the decision boundary, and δ is a selected threshold.

6.2.2 Improved Semi-supervised Target-Dependent Sentiment Classification

The high level overview of the system components used for target-dependent sentiment classification with partially labeled data is illustrated in Figure 6.1. The objective of the proposed system is to allow classifying micro-blogs into more than two classes and not be restricted to only solve problems of binary classification as the case with many systems proposed in the literature. As a proof of concept and for the sake of simplicity, the system is currently implemented to classify micro-blogs into three categories: positive, negative, and neutral. Thus, our proposed system is a multiclass sentiment classifier.


Labeled Micro-blogs
Binary Linear Semi-Supervised Learning
Unlabeled Micro-blogs
Multi-class Classification
Figure 6.1: High level overview of the system components.

The proposed system is based on reducing the need for annotating micro-blogs when training the system of target-dependent sentiment classification. As a result, our system can be used for target-dependent sentiment classification with partially labeled micro-blogs. We show, through experiments, that the performance improves gradually as we increase the percentage of unlabeled micro-blogs during the learning process of the proposed system. However, the performance reaches a peak and starts to decrease as the percentage of unlabeled data increases further. Corresponding experiments are discussed in Chapter 8.

We selected some semi-supervised learning methods and applied them to target-dependent sentiment classification. Then, we used the most suitable one for developing our proposed technique. Additionally, we added some contributions to propose new technique for improving the performance of target-dependent sentiment classification with partially labeled data. We selected specifically self-training technique for developing the proposed technique since it provides competitive accuracy in comparison with other semi-supervised learning methods. Based on our experiment work, we noticed also that self-training technique provides more stationary results (high confidence) in comparison with other semi-supervised learning techniques such as QN-S3VM which is sensitive to setting parameters.

Figure 6.2 illustrates the training model of the proposed solution. For training the model, we input three sets of data that are denoted as X_u, X_l, and L. X_u symbolizes all data points that are generated by using word2vec embeddings for representing unlabelled micro-blogs. X_l represents all labeled data points that are generated by using word2vec embeddings. L denotes values of labels that are corresponding to data points X_l. Such that, if there is a data point $v \in X_l$, then L_v contains the corresponding label value in L.

In our proposed technique, we use two SVM models: SVMT and SVME. It is noteworthy that we used SVM classifiers because it provides competitive performance. However, other classifiers, such as linear logistic regression, can be used. Our experiment results showed that using linear logistic regression provides also high performance with target-dependent sentiment classification.

SVMT represents a support vector machine model used for applying self-training technique. While, SVME denotes to a support vector machine model used for predicting misclassified data points. SVME is used to achieve our idea of improving classification accuracy by trying to predict misclassified data points before detecting the sentiment polarity expressed in the micro-blog. SVME is trained by grouping two data classes from the training set. The first class includes all labeled data points that are classified correctly by using SVMT model. The other class contains all labeled data points that are misclassified by using SVMT model. The data points included in the two classes are denoted as X_{class} and their corresponding labels are denoted as L_{class}.

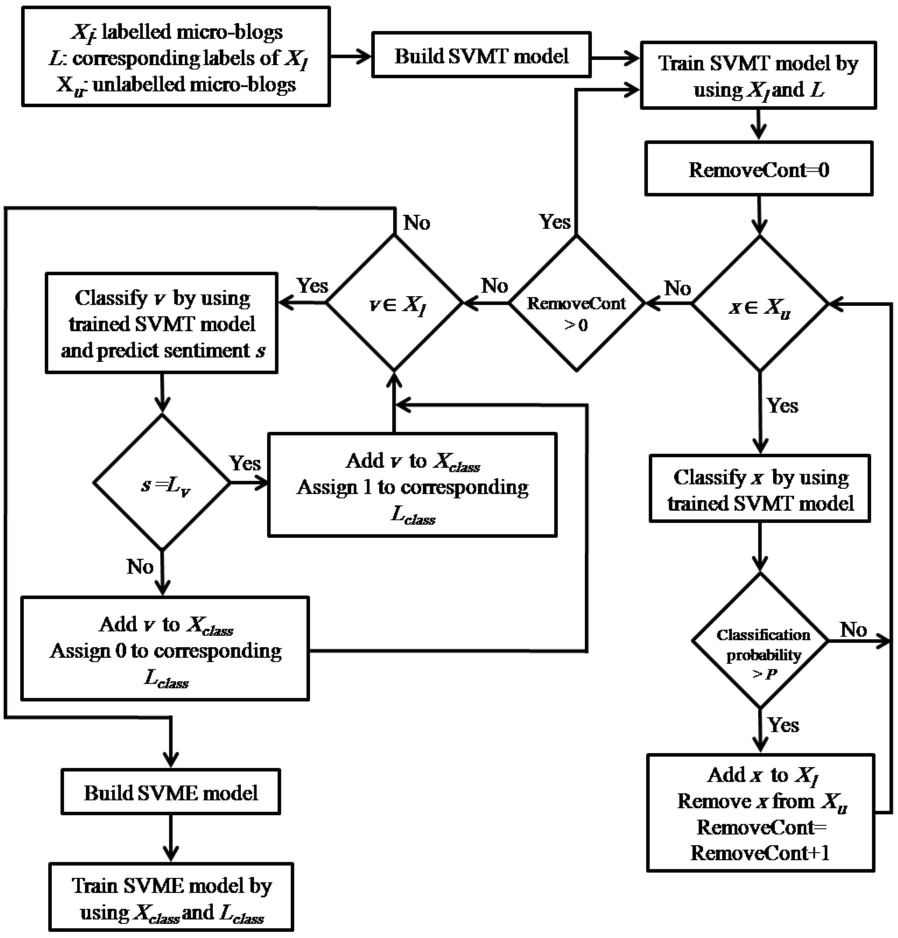

X_l: labelled micro-blogs
L: corresponding labels of X_l
X_u: unlabelled micro-blogs
Build SVMT model
Train SVMT model by using X_l and L
RemoveCont=0
$x \in X_u$
No
Yes
RemoveCont >0
No
Yes
$v \in X_l$
Yes
No
Classify v by using trained SVMT model and predict sentiment s
$s = L_v$
Yes
No
Add v to X_{class} Assign 1 to corresponding L_{class}
Add v to X_{class} Assign 0 to corresponding L_{class}
Classify x by using trained SVMT model
Classification probability $>P$
No
Yes
Add x to X_l Remove x from X_u RemoveCont= RemoveCont+1
Build SVME model
Train SVME model by using X_{class} and L_{class}

Figure 6.2: Training model of the proposed semi-supervised learning technique for target-dependent sentiment classification.

The proposed technique uses input parameter P for specifying confidence level of selecting unlabelled data points that are classified to be added to an incremental labeled data during each round in self-training phase. We initialize this parameter before running the technique and use trial and error to select the optimum value that performs high classification accuracy. In addition, we select the best value of C parameter that is used for building SVM models (SVMT and SVME). The proposed technique is sensitive to the initial value of C parameter and its performance is affected by efficacy of SVMT and SVME classifiers.

Figure 6.3 shows how our proposed technique detects sentiment polarity of input micro-blog. We use SVMT for calculating the three probabilities (P_+, P_-, P_o) toward input micro-blog. P_+ value determines confidence probability of detecting the sentiment polarity as positive, while P_- and P_o refer to confidence probabilities of negative and neutral polarities respectively.

The prediction model is based also on SVME classifier for predicting whether a data point m is misclassified or classified correctly. If SVME classifies m as correct class (c = 1), then the sentiment polarity will belong naturally to the opinion which refers to the highest value of the three probabilities P_+, P_- and P_o (max(P_+, P_-, P_o)). While, if m is predicted as misclassified (c = 0), then the model will select the sentiment polarity that refers to the second highest value of the three probabilities P_+, P_- and P_o (max2(P_+, P_-, P_o)).

Working mechanism of the proposed technique is based on an expectation that the smallest value of the three probabilities P_+, P_- and P_o refers usually to wrong sentiment polarity. Thereby, the outcome will belong usually to the sentiment polarity which refers to the maximum value of the three probabilities P_+, P_- and P_o when SVME model predicts the micro-blog as correctly classified. Otherwise, the technique will select the other sentiment polarity which refers to the second highest value of the three probabilities P_+, P_- and P_o.

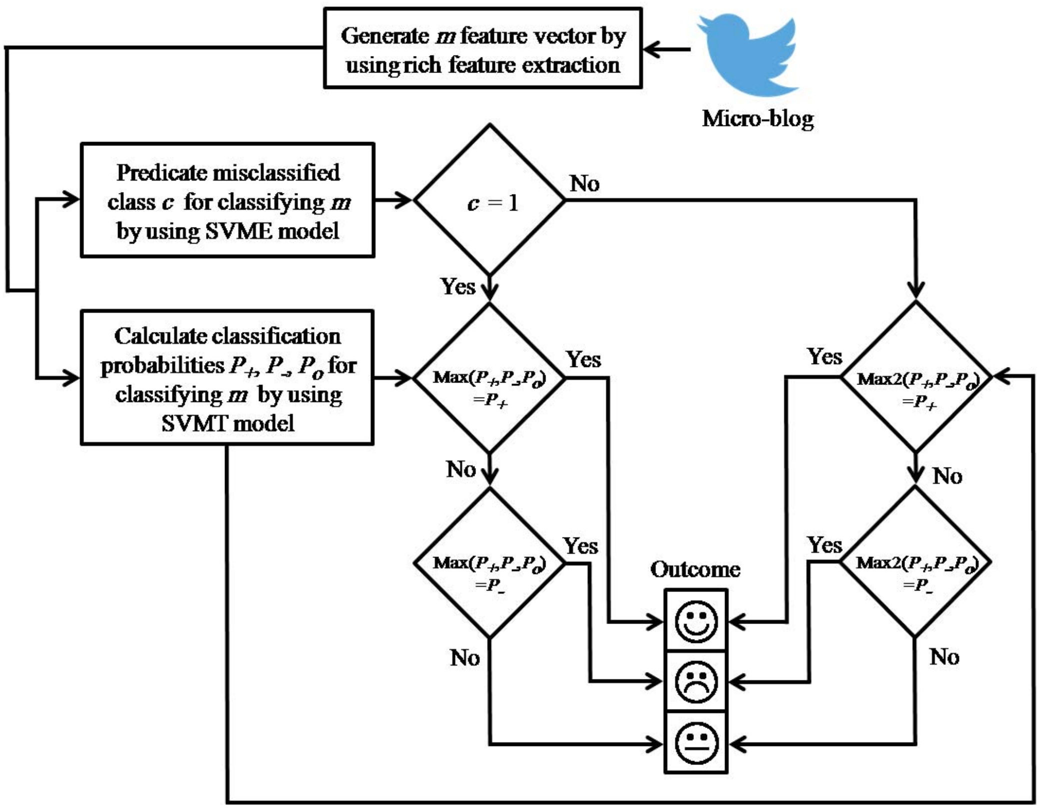

Generate *m* feature vector by using rich feature extraction
Micro-blog
Predicate misclassified class *c* for classifying *m* by using SVME model
c = 1
No
Yes
Calculate classification probabilities P_+, P_-, P_o for classifying *m* by using SVMT model
Max(P_+,P_-,P_o) =P_+
Yes
No
Max(P_+,P_-,P_o) =P_-
Yes
No
Outcome
Max2(P_+,P_-,P_o) =P_+
Yes
No
Max2(P_+,P_-,P_o) =P_-
Yes
No

Figure 6.3: Flowchart of detecting sentiment polarity towards input micro-blog by using the proposed semi-supervised learning technique for target-dependent sentiment classification.

6.2.3 Supervised Classification using Dimension Reduction

This section describes two proposed techniques for improving the performance of target-dependent sentiment classification by using fully labeled data. These techniques are based on reducing number of feature attributes that represent each micro-blog in the dataset. Then, each technique uses a different method for detecting sentiment polarities. All details are discussed in the sequel.

6.2.3.1 Supervised Classification using K-means with Dimension Reduction

This technique is based on exploiting efficacy of applying dimension reduction by using LDA method. Using LDA makes the sentimental classes (positive, negative, and neutral) more separable and easy to classify by reducing number of dimensions of each data point in the dataset and transferring the problem into a lower dimensional level. We applied K-means method after applying LDA for increasing classification accuracy of target-dependent sentiment classification.

Figure 6.4 shows the training model of the proposed technique. The unlabeled data is used as input to the K-means method for applying data clustering. Since K-means method is sensitive to the initial values of centriods, we develop new methods for selecting the initial values of centriods. Process of initializing centriods is based on selecting randomly each centriod from its corresponding class.

There are three centriods since the dataset include three sentimental classes. The first centriod is selected randomly from the data points located in positive class. Similarly, the second and third centriods are selected randomly from the data points located in corresponding negative and neutral classes. Thus, the output of the proposed training model is a three centriods that represents the three resulted clusters (positive, negative, and neutral). Each centriod represents one cluster (group) of data points which forms one of corresponding classified classes (positive, negative, or neutral).

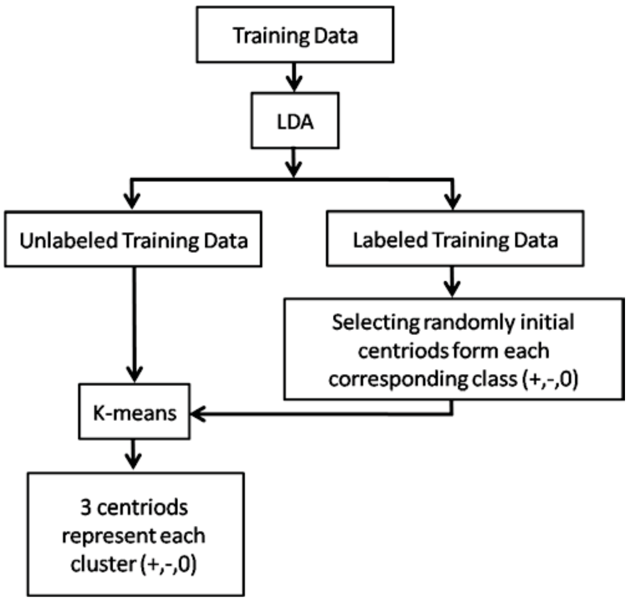

Training Data
LDA
Unlabeled Training Data
Labeled Training Data
Selecting randomly initial centriods form each corresponding class (+,-,0)
K-means
3 centriods represent each cluster (+,-,0)

Figure 6.4: Training model of the proposed supervised learning technique that combines dimension reduction with K-means.

The reason behind initializing each centriod to its corresponding labeled class is two folds. First, it reduces the randomness in initializing the centriods and increases probability of locating each centriod in its correct corresponding class. Second, it increases classification accuracy since it decreases the randomness in initializing centriods. Experiment results are shown in Chapter 8 for validating these two folds.

Figure 6.5 illustrates how our proposed technique detects sentiment polarities expressed in input micro-blog. Firstly, feature vector is extracted from the input micro-blog. It is noteworthy that the feature vector (represents each micro-blog) is normalized as described in Chapter 6. Then, the feature vector is reduced to lower level by using LDA. After that, the technique finds which centriod is closest to the reduced feature vector. Finally, the output label is specified based on the closest centriod to the reduced feature vector. For example, if the positive centriod was the closest one, then the technique will assign *positive* as a sentiment polarity to the given micro-blog.

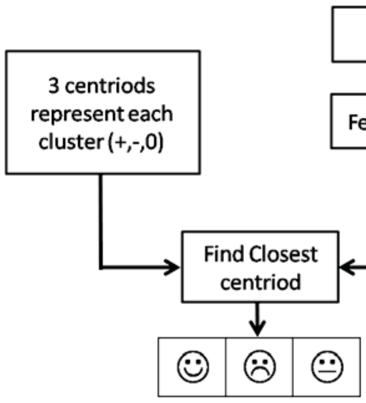


Figure 6.5: Flowchart of detecting sentiment of micro-blog by using proposed supervised solution.

To find the closest centriod to the reduced feature vector, we use different distance measures [140]. Thus, we select the best distance measure that provides the best classification accuracy when classifying the dataset. Section 9.3.1 shows efficiency of using different distance measures when applying our proposed technique. Moreover, Section 9.3.2 shows efficiency of our proposed technique when applying different clustering scenarios.

6.2.4 Combined Supervised Learning Technique

As an alternate to the technique presented above, we propose a technique that combines two supervised learning methods for improving the performance of target-dependent sentiment classification. Our contribution here is due to the dimension reduction. The proposed solution combines both LDA with other supervised learning method such as linear logistic regression. We use LDA method for reducing number of feature attributes and then apply linear logistic regression for classifying the output of LDA. We used LDA for reducing number of dimensions because it is more efficient in comparison with other methods such as PCA as shown in Chapter 6.

In this technique, we use linear logistic regression instead of K-means method. In contrast to the K-means method, linear logistic regression does not require to initialize centriods. However, it requires adjusting a parameter C when building the model. Moreover, we use neural networks instead of linear logistic regression to employ deep learning in our proposed technique. In this case, we tune more parameters to hit the best accuracy. We discuss the corresponding experiment design in Section 9.4.

CHAPTER 7

EXPERIMENT RESULTS: SUPERVISED LEARNING FOR TARGET-DEPENDENT SENTIMENT ANALYSIS

This chapter presents all results of applying numerous supervised learning techniques to DatasetA for target-dependent sentiment analysis. We also include comparisons with previous results reported in the literature. We selected basically the research work of Vo and Zhang [96] as a benchmark to make comparisons. The rationale behind selecting this work is that it meets research objective RO1 in our dissertation. We discuss the application of different supervised learning techniques in the sequel.

7.1 Simple Classifiers

This section shows results of evaluating some selected classifiers. The section includes these selected classifiers together since they are applied easily without a need to setting many parameters. While, the next sections in the chapter discuss individually results of applying other classifiers.

Applying decision tree classifier provides always fixed results. The resulted classification accuracy is 51.9% while the macro-average F1-score is 48.8%. We evaluated performance of applying two forms of Naive Bayes classifiers. We used ***Gaussian Naive Bayes*** classifier which provides always fixed results. The resulted classification accuracy is 53.6% and the macro-average F1-score is 50.6%. We also evaluated performance of

Naive Bayes classifier for multivariate Bernoulli models which provides always classification accuracy that is equal to 62.7% while macro-average F1-score equals 61.6%.

Discriminant analysis methods are designed basically for solving binary classification problem (includes only two classes). Thus, we applied multiclass strategy to apply these methods to the used dataset which includes three classes (positive, negative, and neutral). Linear and quadratic discriminant analysis methods are used in conducting these experiments. ***Linear discriminant analysis*** provides always fixed results. Using one-vs-rest (on-vs-all) strategy provides classification accuracy that is equal to 56.4% and the macro-average F1-score equals 53.7%. While, applying one-vs-one strategy provides lower classification accuracy that is equal to 49.9% and lower macro-average F1-score that is equal to 47.6%. Using ***quadratic discriminant analysis*** provides lower accuracy. When using one-vs-rest strategy the classification accuracy becomes 50.4% and the macro-average F1-score equals 25.5%. While, using one-vs-one strategy provides classification accuracy that is equal to 50.1% and the macro-average F1-score equals 24.7%.

We used ***k-Nearest neighbors classifier*** to check its efficacy in classifying the dataset. For simplicity, we set leaf size parameter to 30 when constructing the tree while the selected distance measure was Minkowski with uniform weights. This classifier applies the k-nearest neighbors vote. Thereby, it is sensitive to the value of parameter which represents the k-nearest neighbors.

To find the best results that can be achieved by k-nearest neighbors classifier, we changed the value of k-nearest neighbors from 1 into 45 by using increasing step that is equal to one. When we set k-nearest neighbors to a value that is equal to one, then the classification accuracy becomes 56.5% and the macro-average F1-score equals 53.8%. The performance is improved gradually by increasing value of k-nearest neighbors until it reaches the best classification accuracy.

The best classification accuracy achieved by this classifier is equal to 62.9% while the corresponding macro-average F1-score equals 57.9% when setting k-nearest neighbors to 18. Then, the performance is decreased again by increasing value of k-nearest neighbors over 18 until it reaches 58.8% and 50.6% for classification accuracy and macro-average F1-score respectively when the value of k-nearest neighbors equals 45. We also tested efficiency of ***nearest centroid classifier*** by setting Minkowski distance measure. However, using other distance measures did not change the results. The reported classification accuracy and the macro-average F1-score are 56.2% and 55.3% respectively.

7.2 Generalized Linear Models

We tested efficacy of applying different linear classifiers to the dataset. We used ***ridge regression*** which provides always fixed classification accuracy that is equal to 60.0% while macro-average F1-score equals 57.3%. We also used ***logistic regression with cross validation*** (CV) for classifying the dataset. This classifier is based on logistic regression and uses *liblinear* library which is the same library used for implementing the classifier that is selected by Vo and Zhang [96]. We tested logistic regression with CV that is equal to 5 folds which provided classification accuracy that is equal to 70.4% while the corresponding macro-average F1-score equals 68.0%. The performance is decreased when using CV that is equal to 10 folds by providing classification accuracy that is equal to 70.2% and the macro-average F1-score equals 67.8%.

Additionally, we applied ***regularized linear models with stochastic gradient descent (SGD) learning*** for classifying the same dataset. The regularizer adds a penalty to the loss function for shrinking model parameters towards the zero vectors. We tested effects of changing penalty as shown in Table 7.1 by using either the Squared Euclidean norm L2 or the absolute norm L1 or a combination of both (Elastic Net).

Table 7.1: Regularized linear model with different settings.

Loss	**Penalty**	**Accuracy**	**Macro-F1**
Hinge	Norm L1	62.3	62.4
Hinge	Norm L2	65.6	57.0
Hinge	Elasticnet	63.0	56.4
Hinge	None	**67.5**	**65.9**
Log	None	67.5	64.7
modified_huber	None	66.6	63.6
squared_hinge	None	64.5	58.3
Perceptron	None	62.6	62.5

We also checked effects of changing loss function as illustrated in Table 7.1. We tested 'hinge' loss function for giving a linear SVM. Additionally, we tested 'log' loss function for giving logistic regression as a probabilistic classifier. Another smooth loss function called 'modified_huber' is checked also to bring tolerance to outliers as well as probability estimates. The 'squared_hinge' loss function is tested as well to mimic 'hinge' loss function with quadratic penalty. Moreover, we tested 'perceptron' loss function to build linear loss that can be used by the Perceptron method.

The best results are reported with 'hinge' loss function and without adding penalty. We run the experiment of best results 12 times to find confidence interval with confidence equals 95%. The maximum reported results are 67.5% and 65.9% for accuracy and macro-average F1-score respectively. While, the average reported values are 64.8% and 62.7% for accuracy and macro-average F1-score respectively with confidence interval that is equal to ± 1.5% for accuracy and ± 1.2% for macro-average F1-score. For more details, refer to Table I.1 in Appendix I to see all numerical values that are provided when conducting this experiment.

Moreover, we used ***ridge classifier with built-in cross validation*** for classifying the dataset. We tested this classifier without CV and applied also 5 and 10 folds, but the results did not change when selecting different folds of CV. The provided results are 65.5% and 63.0% for classification accuracy and macro-average F1-score respectively. We also tested efficiency of two other generalized linear classifiers called logistic regression and passive aggressive. All details of the experiment work developed by using these two classifiers are described in the sequel. These classifiers acquired on our concern because they are too close (in implementation) to the baseline classifier which is used by Vo and Zhang [96].

Logistic regression classifier is implemented by using *liblinear* library which is the same library used for implementing the classifier that is selected by Vo and Zhang [96]. Vo and Zhang tried to find the best value of *C* parameter for optimizing the performance. Best classification accuracy provided by using their implementation code is 69.9% at *C* value that is equal to 0.001 while the best reported accuracy is 71.1%. They explained the difference in results by clarifying that reported accuracy is achieved by evaluating more fine grained values of *C* parameter.

Vo and Zhang tried to find the best *C* value for increasing classification accuracy of the training data (which is divided into training and development sets). Then, they used the best selected *C* value for classifying the testing data. Table 7.2 shows results of classifying the training data by using their implementation code. They used 5 folds for training the classifier with cross validation. They selected value of *C* that is equal to 0.001 for providing the maximum accuracy of using 5-fold training. Then, they classified testing data by using this value of *C* parameter. The resulted classification accuracy of classifying testing data with *C*=0.001 is 69.9% which means that 484 tweets out of 692 tweets are classified correctly.

Table 7.2: Classifying training data with different *C* values.

Average accuracy	***C***	**Average accuracy**	***C***
53.0	10^{-5}	66.7	0.01
59.3	3×10^{-5}	65.1	0.03
62.1	5×10^{-5}	64.1	0.05
63.8	7×10^{-5}	63.5	0.07
64.5	9×10^{-5}	62.8	0.09
65.0	0.0001	62.5	0.1
66.9	0.0003	60.9	0.3
67.4	0.0005	60.3	0.5
67.6	0.0007	59.9	0.7
67.8	0.0009	59.9	0.9
67.8	**0.001**	59.6	1.0
67.5	0.003	58.9	3.0
67.2	0.005	58.3	5.0
67.2	0.007	58.6	7.0
66.8	0.009	58.8	9.0

In our work, we mimic the same strategy in finding the best value of *C* parameter by using the same values illustrated in Table 7.2. We tried to find the best *C* value that increases classification accuracy of testing data instead of evaluating training data. Table 7.3 shows results of our experiment by learning logistic regression classifier. The left columns show accuracy and F1-score of classifying training data to illustrate whether there is an overfitting during learning phase at each *C* value. The right columns show classification accuracy and macro-average F1-score of classifying testing data after learning the model by using training set.

Table 7.3: Unbalanced logistic regression.

Classifying Training Data		**Classifying Testing Data**		***C***
Accuracy	**F1-score**	**Accuracy**	**F1-score**	
50.0	22.2	50.1	22.6	10-5
50.7	24.1	50.6	24.2	3×10-5
52.0	27.6	50.7	26.1	5×10-5
53.2	30.8	51.9	29.9	7×10-5
54.2	33.8	54.0	34.9	9×10-5
54.9	35.6	55.1	37.2	0.0001
62.4	52.7	61.7	52.9	0.0003
65.3	58.5	64.6	58.2	0.0005
66.8	61.0	65.3	59.8	0.0007
67.4	62.3	65.9	60.8	0.0009
67.9	63.0	66.9	62.1	0.001
71.5	68.4	69.2	65.8	0.003
72.8	70.0	69.9	67.1	0.005
73.9	71.4	70.5	67.8	0.007
74.5	72.1	**71.0**	**68.4**	**0.009**
74.9	72.6	70.8	68.2	0.01
77.8	76.0	70.4	68.0	0.03
79.6	77.9	69.8	67.5	0.05
80.6	79.2	69.4	67.1	0.07
81.5	80.1	69.4	67.1	0.09
81.8	80.5	68.9	66.7	0.1
86.0	85.1	67.1	65.0	0.3
87.8	87.1	66.3	64.2	0.5
89.2	88.6	65.9	63.8	0.7
90.1	89.6	65.5	63.4	0.9
90.5	90.1	65.6	63.7	1.0
94.0	93.8	63.4	61.5	3.0
95.7	95.6	62.3	60.2	5.0
96.7	96.6	61.4	59.4	7.0
97.4	97.4	60.5	58.6	9.0

Since the dataset contains unequal distribution of three classes (positive, negative, and neutral) we tested a "balanced" mode for changing weight of each class. The balance mode adjusts automatically weights inversely proportional to class frequencies in the input data. All classes are supposed to have weight one, since no class weight is given in the dataset. Table 7.3 shows results of using unbalanced mode while Table 7.4 shows results of applying balanced mode. We noticed that classification accuracy is converged to maximum value at *C* that is equal to 0.009 which is different to the *C* value revealed when applying implementation code provided by Vo and Zhang [96]. We also noticed that execution time of each run is increased sharply with increasing *C* value.

Table 7.4: Balanced logistic regression.

Classifying Training Data		**Classifying Testing Data**		***C***
Accuracy	**F1-score**	**Accuracy**	**F1-score**	
50.9	24.8	50.6	24.2	10-5
55.5	37.5	55.8	39.6	3×10-5
60.0	48.3	60.8	51.0	5×10-5
62.5	54.0	61.7	53.9	7×10-5
63.5	56.6	62.6	56.3	9×10-5
63.9	57.6	62.7	56.6	0.0001
67.2	64.4	66.3	63.4	0.0003
68.1	66.0	68.1	65.8	0.0005
68.2	66.4	68.2	66.1	0.0007
68.5	66.7	68.2	66.3	0.0009
68.8	67.1	68.5	66.5	0.001
70.7	69.3	68.1	66.6	0.003
71.5	70.3	68.6	67.2	0.005
72.3	71.1	68.4	67.0	0.007
73.1	71.9	68.5	67.1	0.009
73.3	72.1	68.4	67.0	0.01
77.2	76.2	**68.9**	**67.5**	**0.03**
79.1	78.3	68.4	66.9	0.05
80.3	79.5	68.4	67.1	0.07
81.4	80.6	68.4	67.0	0.09
81.8	81.1	67.9	66.6	0.1
86.0	85.5	66.3	65.0	0.3
88.2	87.8	66.3	64.9	0.5
89.7	89.4	65.5	64.0	0.7
90.5	90.2	65.5	63.9	0.9
90.9	90.6	65.3	63.7	1.0
94.4	94.2	63.6	61.9	3.0
96.0	95.9	62.9	61.2	5.0
96.8	96.8	61.7	60.1	7.0
97.3	97.3	61.3	59.6	9.0

We also checked effect of overfitting by applying 5-fold validation by using two selected *C* values that are reported in Table 7.4. To make a good comparison, we selected the first

C value equals 0.009 since it provides the best classification accuracy while the other *C* value equals 9.0 to provide worse accuracy. The results provided by using 5-fold CV of the two states are illustrated in Table 7.5. As shown in the table, the overfitting status (when *C*=9.0) generates low accuracies in all 5 folds and the resulted average accuracy is low. Whereas, setting *C*=0.009 provides higher classification accuracies in all 5 folds and the resulted average accuracy is high as well which means that there is no overfitting in this case.

Table 7.5: Effect of overfitting.

C	**Fold 1 Accuracy**	**Fold 2 Accuracy**	**Fold 3 Accuracy**	**Fold 4 Accuracy**	**Fold 5 Accuracy**
0.009	68.7	68.1	70.1	67.3	68.5
9.0	60.2	60.3	60.4	60.0	57.9

We evaluated performance of applying ***passive aggressive classifier*** by using same strategy with same *C* values. All results provided by this classifier are reported in Table I.2 in Appendix I. The best results achieved by using this classifier are 69.7% and 67.3% for classification accuracy and macro-average F1-score respectively when *C* equals 0.0007.

7.3 Support Vector Machine

This section shows efficiency of applying two SVM models that are implemented by using two different libraries: linear support vector classification and C-support vector classification. We developed the experiments by applying the same strategy which is used in previous section when evaluating generalized linear models.

7.3.1 Linear Support Vector Classification

Linear support vector classification (SVC) is a support vector machine tool with linear kernel. It is implemented in terms of *liblinear* library which is used by Vo and Zhang [96]. We tested this classifier with two multiclass strategies: one-vs-the-rest (OvR) and one-vs-one (OvO) as illustrated in Table I.3 and Table I.4 respectively in Appendix I. Applying linear SVC again by using same *C* value and multiclass strategy provides always the same results. Thus, we did not repeat running these experiments for calculating confidence intervals.

The best results achieved by using linear SVC with OvR multiclass strategy was 70.5% and 68.1% for classification accuracy and macro-average F1-score respectively when *C* value equals 0.003. While, the best results that achieved by using linear SVC with OvO multiclass strategy was 70.7% and 67.9% for classification accuracy and macro-average F1-score respectively when *C* value equals 0.0009. We noticed clearly that there is an overfitting in the learning model when increasing *C* value more than 0.003. We also noticed that there is no significant difference between using OvR or OvO multiclass strategies when finding the maximum classification accuracy.

7.3.2 C-Support Vector Classification

C-support vector classification is a support vector machine model that is implemented in terms of libsvm[6] library. Table I.5 and Table I.6 in Appendix I show results of applying C-SVC when using unbalanced and balanced modes respectively. Based on our experiment results, we noticed that using OvO and OvR multiclass strategies with C-SVC

[6] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

generate approximately similar results. Thus, we selected OvO as default multiclass strategy.

Since the dataset is unbalanced (25% positive, 25% negative, and 50% neutral). We set the parameter *C* of each class *i* to class_weight[i]**C*. Table I.6 shows results of applying balanced mode. As shown in the table, there is no improvement in the performance in comparison with unbalanced mode. Thus, in the reset of experiment work we used only unbalanced mode.

The maximum result achieved by using unbalanced C-SVC linear kernel was 70.2% and 67.7% for classification accuracy and macro-average F1-score respectively when *C* value equals 0.01. We noticed clearly from Table I.5 that increasing *C* value more than 0.01 causes overfitting (accuracy of classifying training data is high) and decreases classification accuracy sharply (accuracy of classifying testing data is low).

The maximum result achieved by using balanced C-SVC linear kernel was 67.6% and 66.9% for classification accuracy and macro-average F1-score respectively when *C* value equals 0.009. We noticed clearly from Table I.6 that increasing *C* value more than 0.009 causes overfitting (accuracy of classifying training data is high) and decreases classification accuracy sharply (accuracy of classifying testing data is low).

We also checked effect of selecting different kernels when building different C-SVC models. Tables from Table 7.6 to Table 7.11 illustrate results of selecting different kernels. The maximum result achieved by using C-SVC with RBF kernel was 70.4% and 66.9% (as shown in Table 7.6) for classification accuracy and macro-average F1-score respectively when *C* value equals 5.0. While, the maximum result achieved by using C-

SVC with sigmoid kernel was 70.1% 66.2% (as shown in Table 7.7) for classification accuracy and macro-average F1-score respectively when *C* value equals 7.0.

Table 7.6: C-SVC with RBF kernel.

Classifying Training Data		**Classifying Testing Data**		***C***
Accuracy	**F1-score**	**Accuracy**	**F1-score**	
50.0	22.2	50.0	22.2	10^{-5} ~0.05
50.4	23.1	50.0	22.6	0.07
51.4	26.2	50.0	24.3	0.09
51.5	26.3	50.1	24.6	0.1
56.1	38.9	56.4	41.5	0.3
60.5	49.4	60.7	51.1	0.5
62.9	54.1	61.6	53.4	0.7
64.7	57.3	63.4	56.5	0.9
65.4	58.4	64.3	57.7	1.0
70.5	66.7	68.8	64.5	3.0
73.1	70.1	**70.4**	**66.9**	**5.0**
74.9	72.3	69.9	66.6	7.0
76.2	73.8	69.9	66.7	9.0

Table 7.7: C-SVC with sigmoid kernel.

Classifying Training Data		**Classifying Testing Data**		***C***
Accuracy	**F1-score**	**Accuracy**	**F1-score**	
50.0	22.2	50.0	22.2	10^{-5} ~0.1
52.3	28.5	51.2	27.2	0.3
55.1	36.2	55.1	37.7	0.5
58.3	44.4	59.1	47.6	0.7
60.2	48.8	61.0	51.2	0.9
60.9	50.3	61.6	52.6	1.0
68.0	63.0	66.2	60.9	3.0
69.8	65.8	68.4	64.0	5.0
71.2	67.7	**70.1**	**66.2**	**7.0**
72.2	69.2	69.5	65.8	9.0

The maximum performance achieved by using C-SVC with poly kernel was 70.5% 67.6% (as shown in Table 7.10) for classification accuracy and macro-average F1-score respectively when *C* value equals 230.0. As shown in tables from Table 7.9 to Table 7.11, we added more scales of selected *C* values to show maximum classification accuracy that can be reached when using this kernel. We noticed that the learning phase converged to maximum accuracy which is equal to 70.4% when value of degree equals 1, 2, and 4.

Table 7.8: C-SVC of poly kernel with degree =1.

Classifying Training Data		Classifying Testing Data		*C*
Accuracy	**F1-score**	**Accuracy**	**F1-score**	
50.0	22.2	50.0	22.2	10^{-5} ~0.09
50.1	22.3	50.0	22.2	0.1
52.3	28.6	51.2	27.1	0.3
55.2	36.6	55.2	38.3	0.5
58.6	45.1	59.1	47.6	0.7
60.3	49.0	60.8	51.2	0.9
61.1	50.6	61.6	52.6	1.0
68.1	63.1	66.5	61.3	3.0
70.0	66.0	68.6	64.4	5.0
71.4	68.1	**70.4**	**66.7**	**7.0**
72.6	69.6	69.2	65.4	9.0

Table 7.9: C-SVC of poly kernel with degree =2.

Classifying Training Data		Classifying Testing Data		*C*
Accuracy	**F1-score**	**Accuracy**	**F1-score**	
50.0	22.2	50.0	22.2	10^{-5}~0.03
50.9	24.7	50.1	24.3	0.5
51.6	26.7	50.1	24.6	0.7
51.9	27.5	50.9	26.4	0.9
52.5	28.9	51.3	27.4	1.0
60.7	49.8	60.8	51.2	3.0
64.7	57.0	63.4	56.1	5.0
66.8	60.7	65.3	59.4	7.0
68.3	63.0	67.2	62.0	9.0
69.7	65.1	68.2	63.5	11.0
70.6	66.5	68.2	63.8	13.0
71.3	67.4	68.2	63.8	15.0
71.6	68.0	68.6	64.4	17.0
72.3	68.9	69.8	65.8	19.0
73.6	70.5	69.7	65.8	23.0
74.4	71.5	**70.4**	**66.8**	**27.0~100.0**
88.8	88.1	69.9	67.5	210.0

The best results achieved by using C-SVC with poly kernel was 70.5% and 67.6% for classification accuracy and macro-average F1-score respectively when *C* equals 230.0 and degree equals 3. This result is the best reported accuracy among all C-SVC settings. Applying C-SVC again with the same settings provides always the same results. Thus, we did not repeat running these experiments for calculating confidence intervals.

Table 7.10: C-SVC of poly kernel with degree =3.

Classifying Training Data		**Classifying Testing Data**		***C***
Accuracy	**F1-score**	**Accuracy**	**F1-score**	
57.7	42.6	57.2	43.4	10^{-5} $\sim 9\times10^{-5}$
57.7	42.6	57.2	43.4	11.0
57.7	42.6	57.2	43.4	13.0
80.7	78.9	70.5	67.3	210.0
81.5	79.8	**70.5**	**67.6**	**230.0**
82.1	80.6	70.5	67.5	250.0
82.9	81.5	70.1	67.2	270.0
84.9	83.7	70.1	67.3	330.0

Table 7.11: C-SVC of poly kernel with degree =4.

Classifying Training Data		**Classifying Testing Data**		***C***
Accuracy	**F1-score**	**Accuracy**	**F1-score**	
50.3	22.9	50.1	22.6	10^{-5} $\sim 9\times10^{-5}$
72.5	68.5	68.2	63.6	250.0
74.3	70.9	69.2	65.0	330.0
75.2	72.1	69.4	65.3	370.0
76.4	73.6	69.8	65.9	430.0
77.0	74.3	69.4	65.4	470.0
78.2	75.7	69.7	65.9	530.0
79.7	77.6	69.9	66.5	630.0
80.9	79.1	**70.4**	**67.1**	**730.0**
82.2	80.5	69.7	66.4	830.0
83.3	81.8	70.1	66.9	930.0
84.0	82.7	70.2	67.2	1030.0
85.1	83.8	69.8	66.6	1130.0

7.4 Deep Learning

This section shows results of evaluating two deep learning techniques that have not been used before in this research direction. We evaluated efficiency of using both neural networks and convolutional neural networks. We applied these two deep learning models to the used dataset for making fair comparisons with other related works.

7.4.1 Neural Networks

This section shows efficiency of applying neural networks with using different settings. We tested performance of using different activation functions such as ‘identity’, ‘logistic’, ‘tanh’, and ‘relu’. The ‘identity’ activation function has no activation and it is useful to implement linear bottleneck by returning f(x) = x. The ‘logistic’ activation function uses logistic sigmoid function by returning f(x) = 1 / (1 + exp(-x)). The ‘tanh’ activation function uses the hyperbolic tan function by returning f(x) = tanh(x). While, the ‘relu’ activation function uses rectified linear unit function that returns f(x) = max(0, x)).

Efficacy of using different solvers is checked also. The evaluated solvers include ‘lbfgs’ (an optimizer in the family of Quasi-Newton methods), ‘sgd’ (stochastic gradient descent), and ‘adam’ (stochastic gradient-based optimizer). Moreover, we changed value of Epsilon parameter which is used only with ’adam’ solver for applying numerical stability. All relevant results are included in Table 7.12.

The best performance is reported by using ‘relu’ activation function and 'adam' solver. We run this experiment 12 times for calculating confidence interval with confidence equals 95% when selecting the best setting. The maximum reported performance is 70.8% and 68.6% for accuracy and macro-average F1-score respectively. While, the average values are 69.9% and 67.2% for accuracy and macro-average F1-score respectively with confidence interval equals ± 0.5% for accuracy and ± 0.7% for macro-average F1-score. For more details, refer to Table I.7 in Appendix I to see all numerical values that are generated from this experiment.

Table 7.12: Neural networks with different settings.

Hidden layers #	**Neurons 1st layer #**	**Neurons 2nd layer #**	**Neurons 3rd layer #**	**Activate**	**Solver**	**Eps**	**Acc**	**F1-score**
1	1000			'relu'	'adam'	10-8	64.5	62.5
1	1000			'relu'	'adam'	0.01	66.3	65.9
1	1000			'relu'	'adam'	0.1	70.4	67.9
1	1000			'relu'	'adam'	0.9	70.5	67.4
1	1000			'relu'	'adam'	1.0	69.4	66.3
1	1000			'relu'	'adam'	1.3	68.5	64.9
1	1000			'relu'	'adam'	3.0	65.8	61.0
1	100			'relu'	'adam'	0.9	68.1	64.8
1	900			'relu'	'adam'	0.9	70.1	67.0
1	1000			'relu'	'adam'	0.9	70.5	67.4
1	1100			'relu'	'adam'	0.9	70.2	67.1
1	1500			'relu'	'adam'	0.9	69.8	66.9
1	1000			'relu'	'sgd'	0.9	68.4	65.6
1	1000			'relu'	'lbfgs'	0.9	63.9	62.2
2	1000	10		'relu'	'adam'	0.9	70.1	67.8
2	1000	50		'relu'	'adam'	0.9	70.7	68.4
2	1000	100		'relu'	'adam'	0.9	70.7	68.3
2	1000	300		'relu'	'adam'	0.9	69.4	66.4
2	1000	600		'relu'	'adam'	0.9	70.5	67.7
2	1000	700		'relu'	'adam'	0.9	70.8	68.1
2	1000	900		'relu'	'adam'	0.9	69.7	67.0
2	1000	700		'identity'	'adam'	0.9	70.5	68.4
2	1000	700		'logistic'	'adam'	0.9	50.0	22.2
2	1000	700		'tanh'	'adam'	0.9	70.5	68.3
2	1000	700		'relu'	'adam'	0.9	70.8	68.1
3	1000	700	50	'relu'	'adam'	0.9	70.2	67.3
3	1000	700	100	'relu'	'adam'	0.9	69.2	66.9
3	1000	700	300	'relu'	'adam'	0.9	69.2	66.7
3	**1000**	**700**	**500**	**'relu'**	**'adam'**	**0.9**	**70.8**	**68.3**
3	1000	700	600	'relu'	'adam'	0.9	68.2	66.1

7.4.2 Convolutional Neural Networks

We evaluated efficacy of applying convolutional neural networks (CNN) to the used dataset. We selected CNN for conducting our experiments since this method has not been used before in this research direction. CNN is proposed basically for dealing with image processing. To make CNN model fits our research problem, we used convolutional layer of one dimension instead of two dimensions.

This experiment was conducted by using a machine with limited specifications (Machine B) as described in Table 5.6 for making our work more competitive. However, applying deep learning methods needs a powerful machine. This experiment was implemented by using specific packages [141] that are used by Python platform for evaluating deep learning techniques. We built a CNN model that includes only one convolutional layer and one hidden layer. We reported the best classification accuracy that is provided when changing settings of pool function, size of pool function, split ratio for validation, optimizer, batch size, padding method, dropout value, activation function, and number of epochs. The best achieved classification accuracy was 62.31%.

7.5 Discussion

Experiment results show clearly that using balanced mode does not improve the performance. The balanced mode is not suitable with the used dataset because the dataset does not include specific weight for each tweet. Using specific weight for each tweet with best mode will change definitely classification accuracy and it may be an effective method in the future.

We also noticed that the optimum values of *C* parameter are not similar when using different classification methods such as linear regression and support vector machine. This change is expected since these models are implemented by using different libraries and they are based on different theoretical backgrounds. For example, logistic regression is converged to maximum accuracy at *C* value equals 0.009 which is different to the *C* value that is picked by using implementation code of Vo and Zhang work. This means that when using different models we need to adjust the *C* parameter again.

Additionally, we noticed obviously that increasing degree of polynomial kernel will increase *C* value that is required to hit maximum accuracy. This phenomenon is appeared because using higher degree of polynomial kernel makes equation of discussion boundary more complex and leads to duplicating values of *C* parameter. We also noticed that using all training data for learning each model causes overfitting (after selecting the best value of *C* parameter). This illation opens the door for employing semi-supervised learning techniques to decrease effect of overfitting and improving classification accuracy. Decreasing amount of labeled data used for learning the model may improve performance of classifying testing data.

Moreover, we noticed that there is no significant difference between using OvR or OvO multiclass strategies when finding the maximum classification accuracy. Our explication for this phenomenon is based on complexly of the used dataset. In addition to that, using multiclass strategy with binary classifiers provides alternative misclassified data points. Most of misclassified data points are belong basically to neutral tweets since they reserve the dominant amount of the dataset (50%). Mainly, classifying neutral tweet is not an easy task by nature since it may tend to positive or negative sentiment polarities. Employing multiclass strategies increase difficulty of classifying neutral tweets and may lead to misclassifying them when applying OvR and OvO strategies. As a result of this, the accuracy of using OvR and OvO strategies become close to each other.

Table 7.13 shows a summary of the best achieved results provided when evaluating supervised learning models. As shown in the table, linear classifiers, SVM, and neural networks provide the best results while logistic regression classifier outperforms all evaluated models. Logistic regression classifier is the best classifier among all linear classifiers since its learning phase mimics deep learning technique.

Table 7.13: Summary of best results provided by using supervised learning methods.

Method		**Setting**	**Acc**	**Macro-F1**
Decision Tree Classifier			51.9	48.8
Naive Bayes	Gaussian Naive Bayes		53.6	50.6
	Naive Bayes classifier for multivariate Bernoulli models		62.7	61.6
Discriminant Analysis	Linear Discriminant Analysis	OVR	56.4	53.7
		OVO	49.9	47.6
	Quadratic Discriminant Analysis	OVR	50.4	25.5
		OVO	50.1	24.7
Nearest Neighbors	k-Nearest neighbors classifier	Leaf size=30, Measure= 'minkowski', *k* neighbors=18	62.9	57.9
	Nearest Centroid Classifier		56.2	55.3
Generalized Linear	Ridge Regression		60.0	57.3
	Logistic Regression CV	CV=5 folds	70.4	68.0
	Stochastic Gradient Descent learning	Penalty= none, Loss Function= Log	67.5	64.7
	Ridge classifier with built-in cross validation	CV=5	65.5	63.0
	Logistic Regression	*C*=0.009	**71.0**	**68.4**
	Passive aggressive classifier	*C*=0.0007	69.7	67.3
Support Vector Machine	Linear Support Vector Classification	OVR, *C*=0.003	70.5	68.1
		OVO, *C*=0.0009	70.7	67.9
	C-Support Vector Classification	Kernel=RBF, *C*=5.0	70.4	66.9
Neural Networks		Hidden layers=3, Activation='relu', Solver='adam', Eps=0.9	70.8	68.3

CV= Cross Validation
OVR= One-VS-Rest (One-VS-ALL) multiclass strategy
OVO= One-VS-One multiclass strategy

We also noticed that using neural networks with 'adam' solver achieves the best accuracy among all settings since this solver works pretty well on relatively large datasets (with thousands of training data points or more). For smaller datasets, however, 'lbfgs' can converge faster and perform better. It is interesting as well to clarify that using deep learning provides competitive results.

7.6 Conclusion

Based on our experiment results, the best results are achieved by using linear classifiers such as SVM with linear kernel and also neural networks. Linear classifiers are the best choice for classifying the used dataset since they provide comparable results. In the same context, linear regression provided the best results. While, neural networks provided the highest confidence for its confidence interval that include mean value. Performance of discriminant analysis classifiers is not so good but these classifiers are faster and they provide consistence results when using scaled or unscaled data.

Additionally, our experimental results show clearly that using all training data for learning each model will cause overfitting. This illation opens the door for employing semi-supervised learning techniques to decrease effect of overfitting and improving classification accuracy. We also noticed that there is no significant difference between using OvR or OvO multiclass strategies when comparing the maximum achieved accuracies.

Moreover, the optimum values of *C* parameter are different when using linear regression and support vector machine. Thus, when applying different models, we need firstly to adjust the *C* parameter by using an optimization method. Additionally, the best value of *C* parameter that provides the optimum accuracy is different when using different implementations of SVM.

CHAPTER 8

EXPERIMENT RESULTS: SEMI-SUPERVISED LEARNING FOR TARGET-DEPENDENT SENTIMENT ANALYSIS

This chapter includes all results that are provided by applying different semi-supervised learning techniques to DatasetA. We also make comparison with previous related works that are reported in literature. We selected basically research work achieved by Vo and Zhang [96] as baseline to make our comparisons and we use the same dataset for making fair comparisons. Moreover, this chapter shows performance of our proposed solutions for developing target-dependent sentiment classification with partially labeled data.

8.1 Labeling Models

Labeling models work by constructing a similarity graph over all data points in the input dataset. Thus, changing number of input data points will be affected directly on the working mechanism of these models. Table II.1 in Appendix II shows effect of changing labeling ratio (ratio of labeled data) on classification accuracy. The values of labeling ratio are increased from 0.01 up to 0.63 with using increasing step that is equal to 0.02. For each selected ratio we reported results of four runs (R1, R2, R3, and R4). We use in this experiment a label propagation method with RBF kernel when the value of Gamma parameter is fixed at 0.07. This table illustrates clearly that classification accuracy will be changed directly when modifying percentage of labeled tweets over unlabeled ones.

It is clear also from Table II.1 that increasing ratio of labeled data causes overfitting and decreases classification accuracy significantly. Based on this note, we conducted another experiments by changing ratio of labeled data from 0.01 up to 0.63 with using increasing step that is equal to 0.02. For each selected ratio we reported results of 12 runs (R1-R12). We selected the labeled data from different parts in the training data. In the first run (R1), we selected the labeled data from the first part of training data. The second run (R2) generates results by selecting labeled data from the last part of training data. The reset of runs (R3-R12) shows results of selecting randomly labeled data from the training data by shuffling training data randomly and then select labeled data from first part or last part of shuffled training data. The next subsections describe all details of our experiment work.

8.1.1 Label Propagation

We used kNN kernel and evaluated effect of changing number of neighbors. When conducting this experiment, we fixed number of labeled training data (51% of training data) to be approximately equals unlabeled training data. In next experiments, we selected labeled data from the first part of training data. We changed number of neighbors (*k*) from 1 into 55 by using increasing step that is equal to 2. Table II.2 in Appendix II shows all numerical values that are generated from these experiments. The maximum achieved result was 61.7% and 62.7% for classification accuracy and macro-average F1-score respectively when number of neighbors equals one.

To make our results more obvious, we classified testing data by using kNN kernel with different ratio of labeled data and fixed number of neighbors to 21 (this value generates one of top ten maximum accuracies). As described above, we also selected the labeled data from different parts in the training data when conducing the 12 runs. Figure 8.1 illustrates results of each selected ratio with its corresponding confidence interval for the 12 runs. For more details, refer to Table II.3 in Appendix II to see all numerical values that are provided from this experiment.

As shown in Figure 8.1, modifying ratio of labeled data will change classification accuracy significantly. This illation is obvious in the figure since there are some mean values of specific ratios (such as ratios between 0.45 and 0.63) do not fall inside the confidence intervals resulted by using other labeling ratios. We also noticed that there is no significant change in the classification accuracy after reaching labeling ratio equals 0.45. The maximum achieved classification accuracy was 54.91% when setting labeling ratio to 0.41.

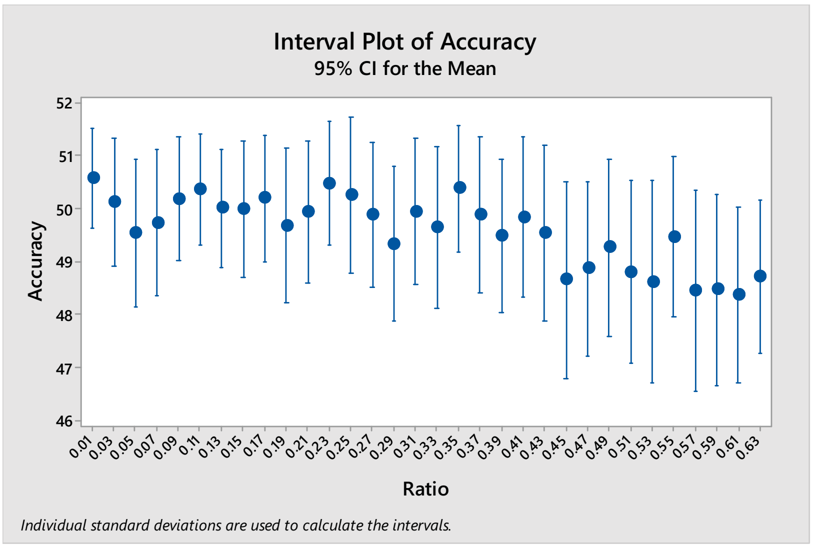


Figure 8.1: Effect of changing ratio of labeled data when applying label propagation. with kNN kernel (*k*=21)

Figure 8.2 shows results of applying label propagation when using kNN kernel with different ratio of labeled data and fixing number of neighbors to 1 (which generated maximum accuracy). For more details, refer to Table II.4 in Appendix II to see all numerical values that are provided when conducting this experiment. We noticed from the table that this experiment provides higher accuracy while differences between confidence intervals are high. The maximum classification accuracy was 56.36% when the labeling ratio equals 0.01.

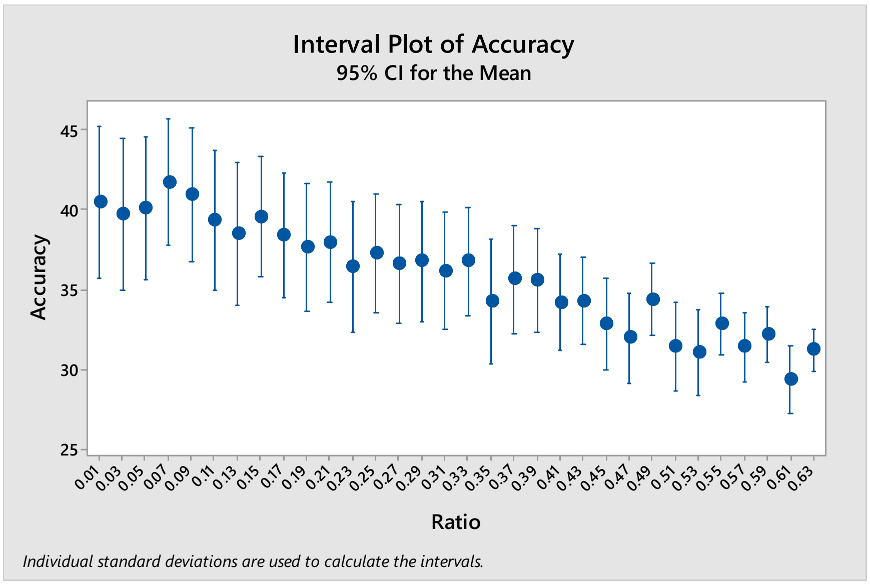


Figure 8.2: Effect of changing ratio of labeled data when applying label propagation with kNN kernel (*k*=1).

We also developed an experiment to evaluate performance of using RBF kernel when changing value of Gamma parameter. When conducting this experiment, we fixed number of labeled training data to 51% of training data. We changed values of Gamma parameter from 0.01 to 2.23 by using increasing step that is equal to 0.02. For more details, refer to Table II.5 in Appendix II to see all numerical values that are provided when conducting this experiment. The maximum reported result was 59.0% and 54.2%

for classification accuracy and macro-average F1-score respectively when setting Gamma parameter to 0.13. We also noticed that the performance becomes fixed when the value of Gamma parameter is equal to or greater than 2.23. Using these settings provides the lowest performance among all runs with 25.0% and 13.3% for classification accuracy and macro-average F1-score respectively.

Figure 8.3 illustrates results of applying label propagation when using kNN kernel. We use here a fixed value of Gamma parameter which equals 0.07 since it provides the highest classification accuracy. For more details, refer to Table II.6 in Appendix II to see all numerical values that are provided when conducting this experiment. The maximum classification accuracy was 60.84% when the corresponding labeling ratio equals 0.07.

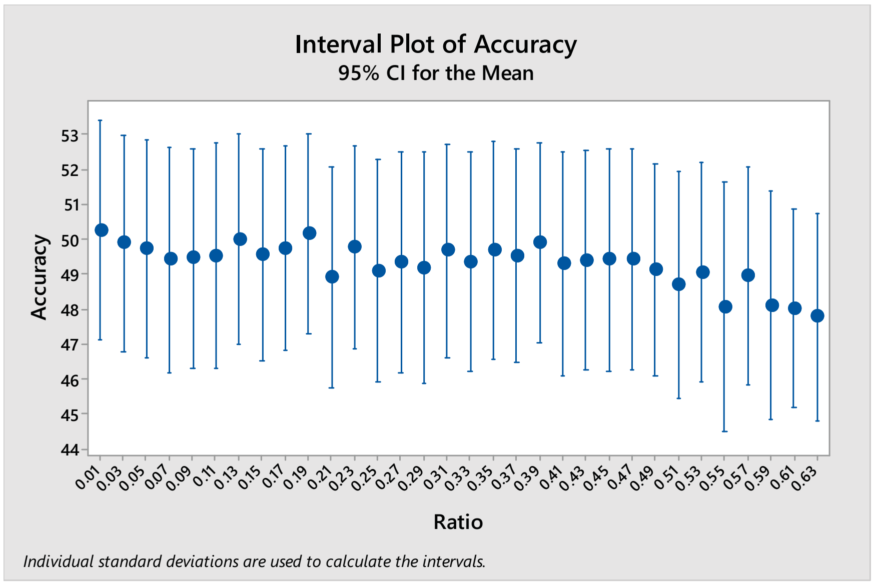


Figure 8.3: Effect of changing ratio of labeled data when applying label propagation with RBF kernel (Gamma=0.07).

8.1.2 Label Spreading

We applied label spreading method with kNN kernel and evaluated effect of changing number of neighbors. When conducting this experiment, we fixed ratio of labeled training data to 51%. The maximum reported result was 58.1% and 53.1% for classification accuracy and macro-average F1-score respectively when number of neighbors equals 7. For more details, refer to Table II.7 in Appendix II to see all numerical values that are provided when conducting this experiment.

Figure 8.4 illustrates results of applying label spreading when using RBF kernel. We used a fixed number of neighbors that is equal to 7 since it provides the highest classification accuracy. For more details, refer to Table II.8 in Appendix II to see all numerical values that are provided when conducting this experiment. The maximum classification accuracy was 59.83% when labeling ratio equals 0.27.

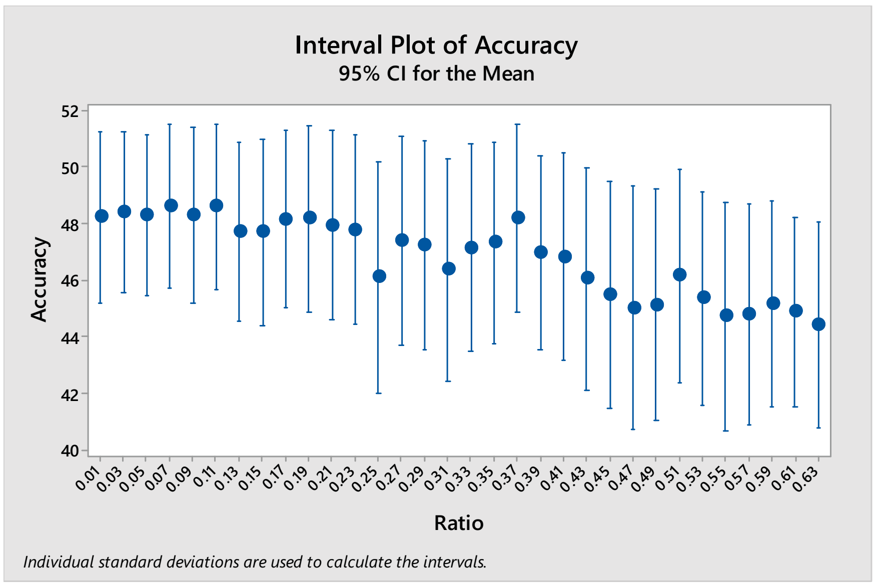


Figure 8.4: Effect of changing ratio of labeled data when applying label spreading with kNN kernel (k=7).

Figure 8.5 illustrates results of applying label spreading with RBF kernel. We used a fixed value with Gamma parameter which equals 0.19 since it provides the highest classification accuracy as shown in Table II.9 in Appendix II. For more details, refer to Table II.10 in Appendix II to see all numerical values that are provided when conducting this experiment. The maximum classification accuracy was 61.42% when labeling ratio equals 0.05.

Table II.9 in Appendix II shows effect of changing values of Gamma parameter when applying label spreading with RBF kernel. When conducting this experiment, we also fixed number of labeled data to be 51% of training data. The maximum reported result was 59.2% and 54.7% for classification accuracy and macro-average F1-score respectively when setting Gamma parameter to 0.19. We also noticed that the performance becomes fixed when the value of Gamma parameter equals or greater than 2.55. These settings provide the lowest performance among all runs with 25.0% and 13.3% for classification accuracy and macro-average F1-score respectively.

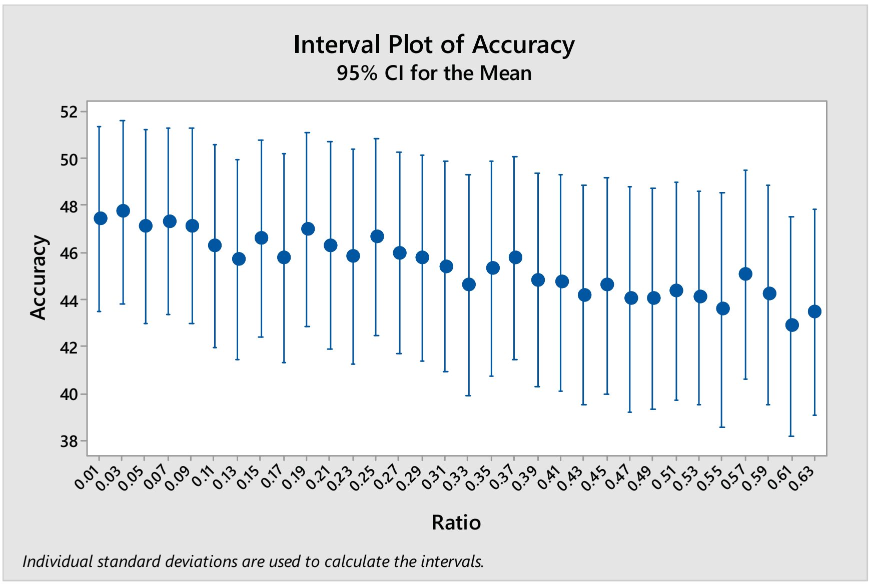


Figure 8.5: Effect of changing ratio of labeled data when applying label spreading with RBF kernel (Gamma=0.19).

8.2 Semi-supervised K-means

We evaluated performance of applying semi-supervised K-means. Figure 8.6 illustrates results of applying semi-supervised K-means on each selected ratio along with its corresponding confidence interval of the 12 runs. We noticed clearly that the mean of classification accuracy is increased gradually when increasing ratio of labeled data. For more details, refer to Table II.11 in Appendix II to see all numerical values that are provided when conducting this experiment. The maximum achieved classification accuracy was 46.82% when using labeling ratio equals 0.37. This maximum value of classification accuracy is provided also when the labeling ratio equals 0.57 and 0.41. We reported only the least ratio that can be used to provide the best performance.

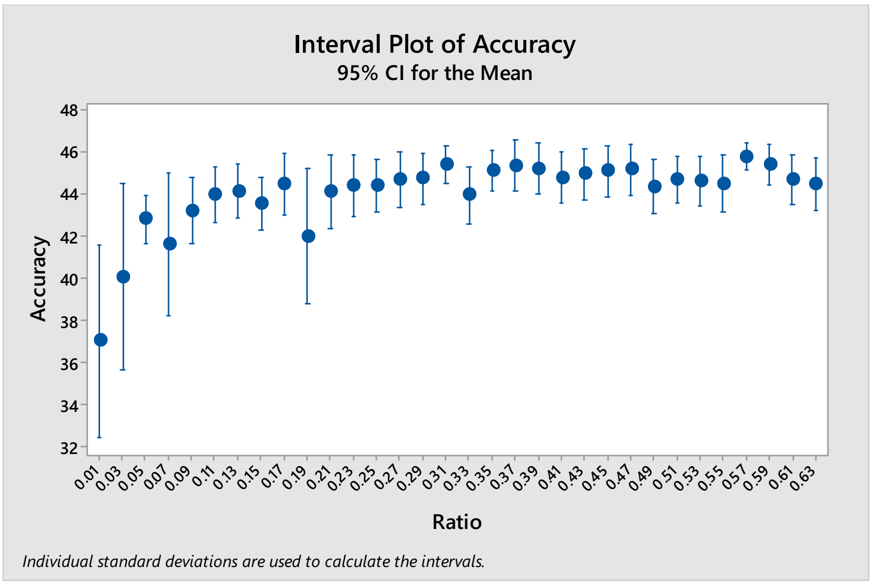


Figure 8.6: Effect of changing ratio of labeled data when applying semi-supervised K-means.

8.3 Self-Training

This section shows results of using self-training technique by applying different supervised methods. We used logistic regression classifier for evaluating performance of applying self-training technique. We used two measures to calculate confidence level for removing specific data points from unlabelled set and adding them to labeled set during learning phase. The first measure is the distance between the date point and the decision hyperplane. While, the other measure is the confidence probability of predicting each data point as positive, negative, or neutral sentiment. The next subsections describe results of all experiments that are conducted for checking efficacy of these methods.

8.3.1 Self-Training Using Logistic Regression Classifier With Distance Confidence

Another experiment is conducted to evaluate performance of applying self-training by using logistic regression with distance confidence. The distance confidence is calculated by using equation 6.1. We applied this formula to the three classes (positive, negative, and neutral) which constructs three dimensional hyperplane decision boundary. We tried to select the best value of threshold δ by changing threshold value and run the experiment again. We used only one round at each run since we need only to find the best threshold value. Based on the experiment work, we noticed that accuracy does not change after three rounds and no significant change within the first three rounds.

The logistic regression classifier is used in this direction because it provides combative results as shown in Chapter 7. We set C=0.009 with OvR multiclass strategy while ratio of labeled data is fixed to 0.51. Labeled data is selected from the first part of training set. The maximum achieved accuracy was 70.2% when value of threshold δ equals 0.81. For more details, refer to Table II.12 in Appendix II to see all numerical values that are provided when conducting this experiment.

Based on the best results provided when applying previous experiment, we selected a value of threshold δ that is equal to 0.81. Then, we checked efficiency of applying self-training using logistic regression classifier with distance confidence. We selected this threshold value as the least threshold that gives maximum accuracy for increasing the chance to train more unlabeled data. We also checked efficiency of applying this semi-supervised learning technique with different ratios of labeled data. We also selected same ratios (from 0.01 to 0.63) that are used in pervious experiments.

Efficiency of applying logistic regression to the same ratios of labeled data is check individually. We conducted this experiment to make a comparison between logistic regression and self-training using logistic regression (using distance confidence). Figure 8.7 illustrates results of applying logistic regression classifier on each selected ratio with its corresponding confidence interval among the 12 runs.

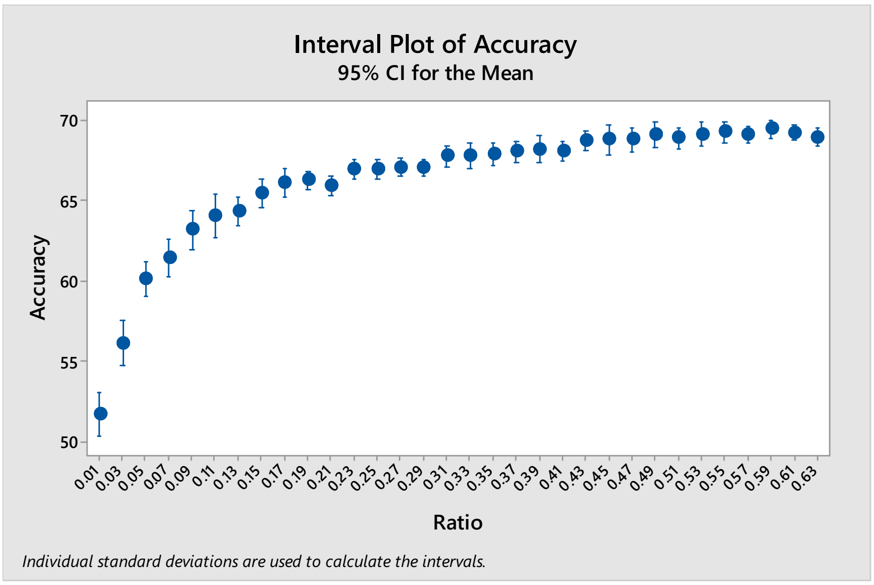


Figure 8.7: Effect of changing ratio of labeled data when applying logistic regression classifier.

We noticed clearly from Figure 8.7 that the mean of classification accuracy is increased gradually when increasing ratio of labeled data. The maximum reported accuracy was 71.97% with labeling ratio that is equal to 0.45. For more details, refer to Table II.13 in

Appendix II to see all numerical values that are provided when conducting this experiment.

We also evaluated performance of applying self-training with logistic regression classifier (using distance confidence) to all selected ratios of labeled data as shown in Figure 8.8. We set threshold δ to a value equals 0.81. The maximum reported accuracy was 70.81% with labeling ratio that is equal to 0.59. For more details, refer to Table II.14 in Appendix II to see all numerical values that are provided when conducting this experiment.

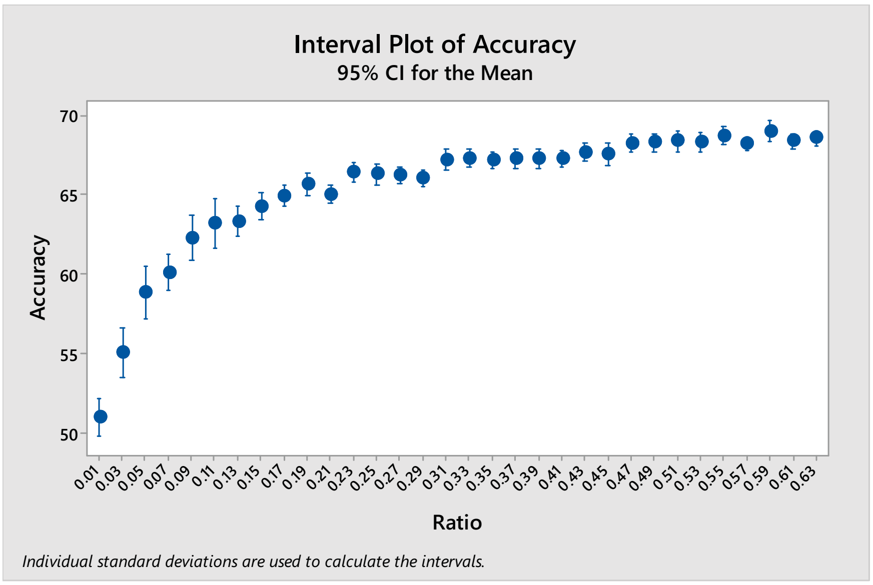


Figure 8.8: Effect of changing ratio of labeled data when applying self-training with distance confidence (δ=0.81).

8.3.2 Self-Training Using Logistic Regression Classifier with probabilistic confidence

Performance of self-training model is evaluated again when using logistic regression with probabilistic confidence. The mode is learned for three rounds. Instead of using Equation 6.1, we use in this experiment a probability of predicting each tweet as positive, negative or neutral sentiment. To select the best probabilistic threshold, we fixed ratio of labeled data to 0.45. The labeled data are selected also from the first part of training set. This ratio is selected because it provided best performance when applying logistic regression classifier as shown in previous section. We reported the performance when changing probabilistic threshold from 0.01 to 0.99 with increasing step that is equal to 0.01. The maximum reported accuracy was 72.1% at probabilistic threshold equals 0.9. For more details, refer to Table II.15 in Appendix II to see all numerical values that are provided when conducting this experiment.

Figure 8.9 illustrates results of evaluating this model when changing ratio of labeled data. We set here probabilistic threshold (Prop) to a value that is equal to 0.9 which provided the best accuracy. The maximum reported accuracy was 72.11% when the labeling ratio equals 0.45. For more details, refer to Table II.16 in Appendix II to see all numerical values that are provided when conducting this experiment.

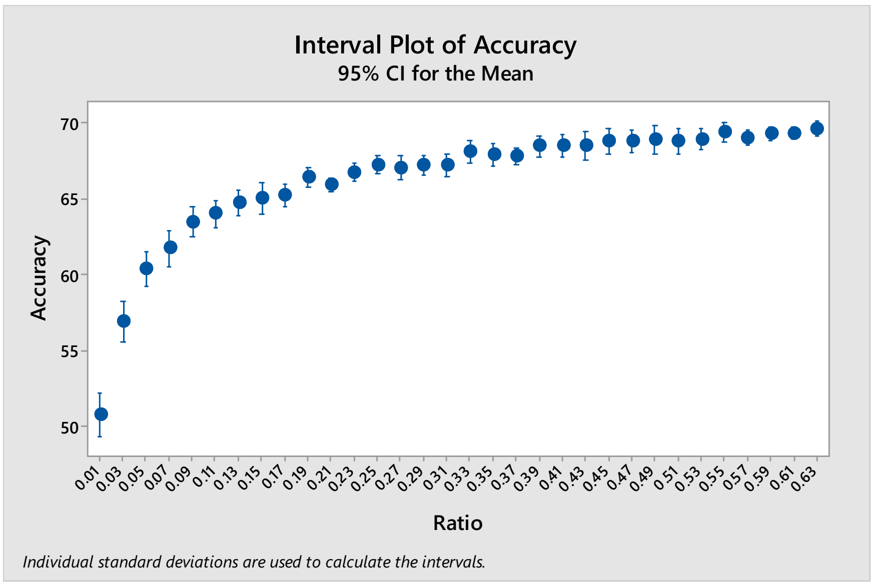


Figure 8.9: Effect of changing ratio of labeled data when applying self-training with probabilistic confidence (Prop=0.9).

8.4 QN-S3VM BFGS Optimizer for Semi-supervised SVM

For evaluation performance of QN-S3VM, we used a tool[7] which is implemented to classify only two classes (positive and negative). Thus, we applied multiclass classification strategy to make this tool suitable to classify our dataset which includes three classes. We modified the implemented code of the used tool to provide numerical values that measure distances from the decision boundary to each data point. For example if the numerical value is -1.9, this means that the corresponding data point is placed in the negative class. This value clarify also that the distance between the data point and the decision boundary is 1.9. Thereby, if this value is increased, then the confidence of adding this data point to the negative class will be increased. As a result of this, these numerical values are employed for building the multiclass classification strategies.

We applied strategies of one-versus-rest (OvR) (called also one-versus-all) and one-versus-one (OvO) for developing our experiments. We applied OvR strategy by building three binary classifiers and selecting the outcome from the classifier that provides the maximum distance between the data point and the decision hyperplane. While, when applying OvO strategy we build three binary classifiers and select the dominant sentiment polarity for each classified micro-blog by using voting strategy.

This tool uses different parameters such as Lamda, LamdaU, Sigma, and Kernel_type. Lamda is a regularization parameter (default 1, must be a float > 0). LamdaU is a cost parameter that determines influence of unlabeled patterns (default 1, must be float

[7] *http://www.fabiangieseke.de/index.php/code/qns3vm*

> 0). If LamdaU equals 0, then the model mimics a supervised learning technique. Sigma is a parameter used width RBF kernel (default 1.0, must be a float > 0).

We used Kernel_type parameter to select linear or RBF kernel when building the model. We use linear kernel by default when conducting the experiments. We noticed that using RBF kernel provides always accuracy that is equal to 50% even when changing values of Lamda and LamdaU parameters from 0.001 to 17.0. We also changed value of Sigma parameter when fixing Lamda and LamdaU values to 1 while the accuracy remained also 50%.

8.4.1 Using One-VS-Rest Strategy

We evaluated performance of applying OvR strategy when changing values of Lamda parameter. We set same value to both Lamda and LamdaU parameters. This tool uses also another parameter initialized randomly. However, we fixed the value used with this parameter to find the best values of Lamda and LamdaU that provide high performance. We changed Lamda from 0.001 to 0.497 with increasing step that is equal to 0.001. The maximum reported accuracy was 71.53% at Lamda equals 0.025. For more details, refer to Table II.17 in Appendix II to see all numerical values that are provided when conducting this experiment. We also noticed that when setting value 0.025 for both Lamada and LamdaU parameters, the classification accuracy hits also the same maximum value (71.53%). Thus, we selected this value for both Lamada and LamdaU parameters when conducting next experiments.

Effect of modifying value of LamdaU is checked by fixing Lamda value to 0.025 and changing values of LamdaU as shown in Table II.18 in Appendix II. We fixed here ratio of labeled data to 51% and changed incrementally values of LamdaU until we observed that the accuracy is decreased. The maximum reported accuracy was 71.53% at LamdaU equals 0.025, 0.009, and 0.005. We noticed that when setting value 0.025 to both Lamada and LamdaU parameters, the accuracy value hits also the same maximum value (71.53%) as described above.

Effect of fixing both Lamda and LamdaU to value 0.025 and changing ratio of labeled data is evaluated also. Figure 8.10 illustrates results of applying QN-S3VM to each selected ratio of labeled data along with its corresponding confidence interval among the 12 runs. The maximum reported accuracy was 71.82% when labeling ratio equals 0.63. For more details, refer to Table II.19 in Appendix II to see all numerical values that are provided when conducting this experiment.

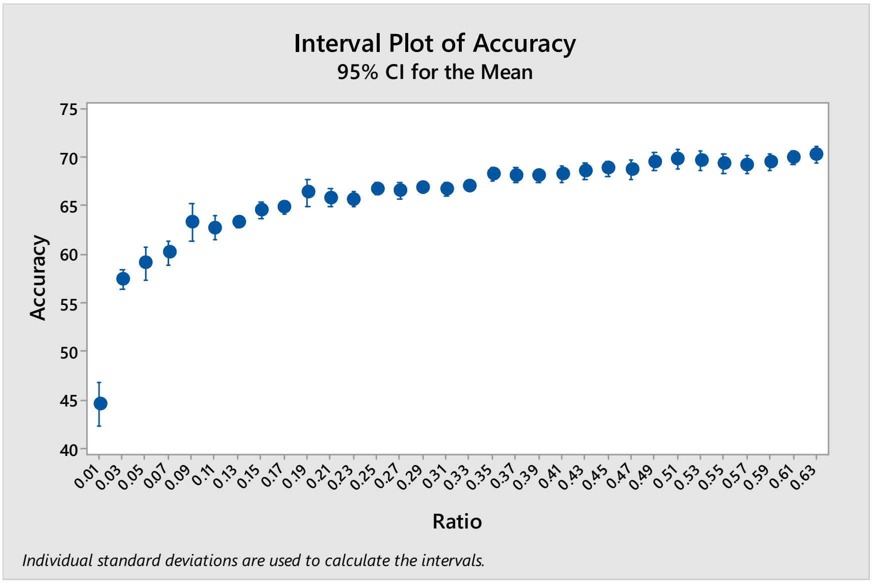


Figure 8.10: Effect of changing ratio of labeled data when applying QN-S3VM BFGS optimizer for semi-supervised SVM (Lamda=0.025) with OvR multiclass strategy.

8.4.2 Using One-VS-One Strategy

We built three models of QN-S3VM for developing a semi-supervised learning technique that classifies the dataset by using one-vs-one strategy. We fixed the ratio of labeled data to 51% and changed both Lamda and LamdaU as shown in Table II.20 in Appendix II. We changed values of both Lamda and LamdaU from 0.001 to 0.261 by applying increasing step that is equal to 0.004. The maximum reported accuracy was 70.1% when setting value of both Lamda and LamdaU parameters to 0.045.

Additionally, effect of changing ratio of labeled data is evaluated when fixing value of both Lamda and LamdaU parameters to 0.045. The maximum provided accuracy was 70.52% when setting ratio of labeled data to 0.61. For more details, refer to Table II.21 in Appendix II to see all numerical values that are provided when conducting this experiment. Similarly, Figure 8.11 illustrates results of applying QN-S3VM with OvO strategy.

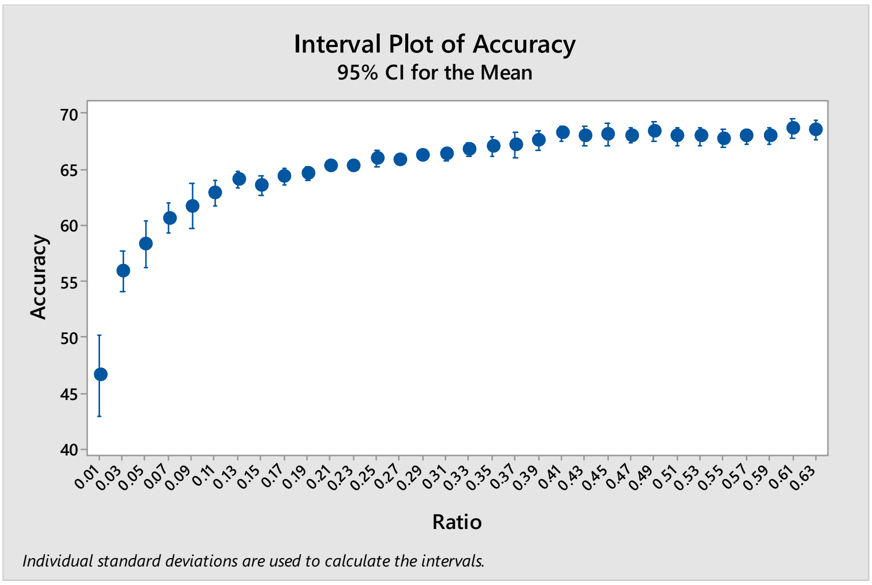


Figure 8.11: Effect of changing ratio of labeled data when applying QN-S3VM BFGS optimizer for semi-supervised SVM (Lamda=0.045) with OvO multiclass strategy.

8.5 Unsupervised Learning Techniques

This section discuss our experiment work for illustrating performance of applying clustering techniques to DatasetA by using fully unlabeled data. Using clustering techniques shows performance of applying semi-supervised learning technique when decreasing extremely number of labeled micro-blogs. It is noteworthy that we selected some clustering methods instead of using semi-supervised learning methods for conducting all relevant experiments. We selected the clustering methods because they are designed mainly for working with unlabelled data. We converted the clustering problem into a classification problem by clustering the dataset into three clusters since the dataset includes three classes (positive, negative, and neutral).

We use K-means and Birch methods for conducting our experiments. We did not use other clustering methods such as DBSCAN, MeanShift, Agglomerative, and affinity propagation because they do not use a parameter for specifying number of required clusters in advance while we need to specify exactly three clusters. Moreover, using other clustering methods provide results that are sensitive to many parameters such as *eps and min_samples.* Parameter *eps* is used with some methods for specifying maximum distance between two data points to be considered as in the same neighborhood. While, *min_samples* parameter is used for specifying number of data points in a neighborhood of a selected data point to be considered as a core data point. Thus, using other methods reveal some difficulties in validating experimental results.

8.5.1 Birch

This subsection shows results of applying Birch clustering method. Birch uses two parameters: the threshold and the branching factor. In our experiments, we set value of the threshold parameter to 0.5. A description of these parameters is presented in Section 2.3.2.1. Clustering testing data into 3 clusters by using the Birch method provides a poor result that was 24.4% and 18.5% for classification accuracy and macro-average F1-score, respectively.

This result is provided because the Birch method generated a cluster that contains only one sample and the majority of samples are assigned to incorrect cluster as shown in Figure 8.12. The figure shows two views for illustrating the result clearly. The first view (Figure a) shows how the classes are distributed in each cluster. While, the second view (Figure b) shows how the clusters are grouped in each class. The horizontal axis in each graph represents accuracy of the distribution, while the other one shows corresponding clusters or classes. For more details, refer to Table II.22 in Appendix II to see all numerical values that are provided when conducting this experiment.

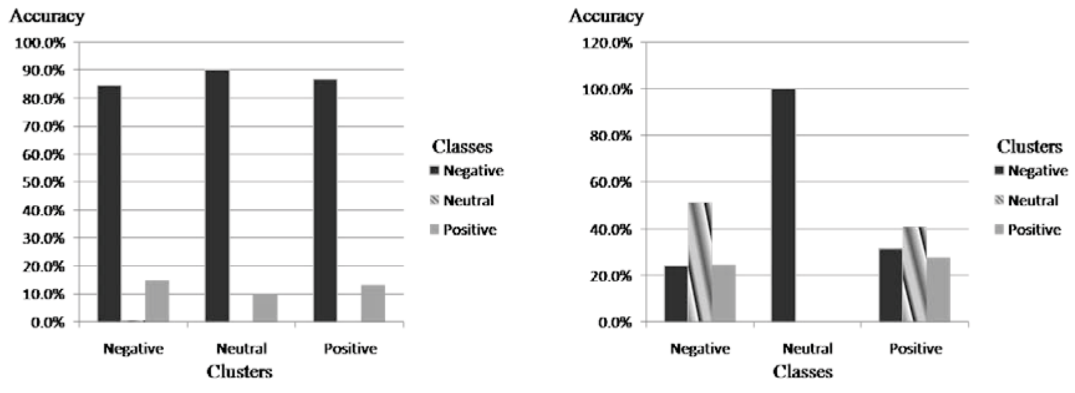


Figure 8.12: Distribution of data points provided when clustering testing data by using Birch method.

Clustering training data into 3 clusters provides classification accuracy that is equal to 27.6% while the macro-average F1-score equals 26.5%. It is clear that the accuracy is improved in comparison with the result shown above when clustering the testing data. This is an expected result because number of data points in training data is larger which leads to learning Birch model more accurately. In general, the accuracy is still worse because majority of data points are assigned to incorrect cluster as shown Figure 8.13. For more details, refer to Table II.23 in Appendix II to see all numerical values that are provided when conducting this experiment.

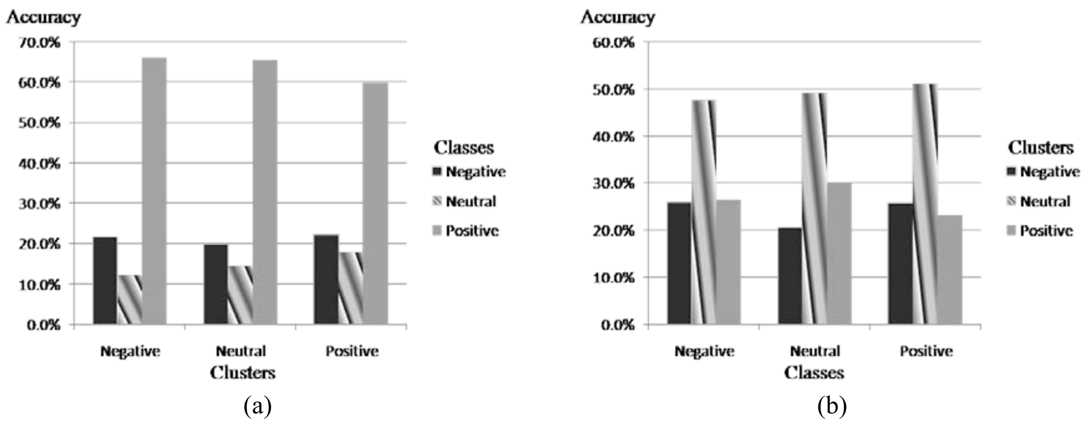
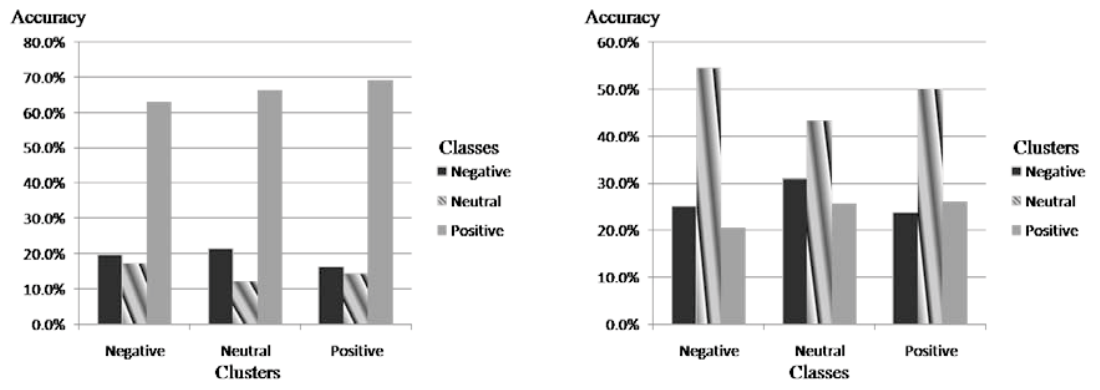


Figure 8.13: Distribution of data points provided when clustering training data by using Birch method.

To improve the performance, we applied firstly the Birch method to the training data and then the testing data is classified. The result was 28.3% and 26.3% for classification accuracy and the macro-average F1-score respectively. As we noted, the resulted accuracy provided from this experiment is better than previous two experiments since the model is learned further. However, the results are still poor in general since the Birch method assigns the majority of data points to incorrect cluster as illustrated in Figure 8.14. For more details, refer to Table II.24 in Appendix II to see all numerical values that are provided when conducting this experiment.



(a) (b)

Figure 8.14: Distribution of data points provided by clustering testing data after clustering training data by using Birch.

8.5.2 K-means

This section shows performance of applying K-means method for clustering the dataset. To check the effect of initializing the centroids used by K-means clustering, we used three methods: 1) Initialize centriods by using K-means++ algorithm [142]. 2) Initialize centriods fully randomly from the data. 3) Initialize each centriod at random from data points included in its corresponding class. Experiment results provided by using these methods are reported in the sequel.

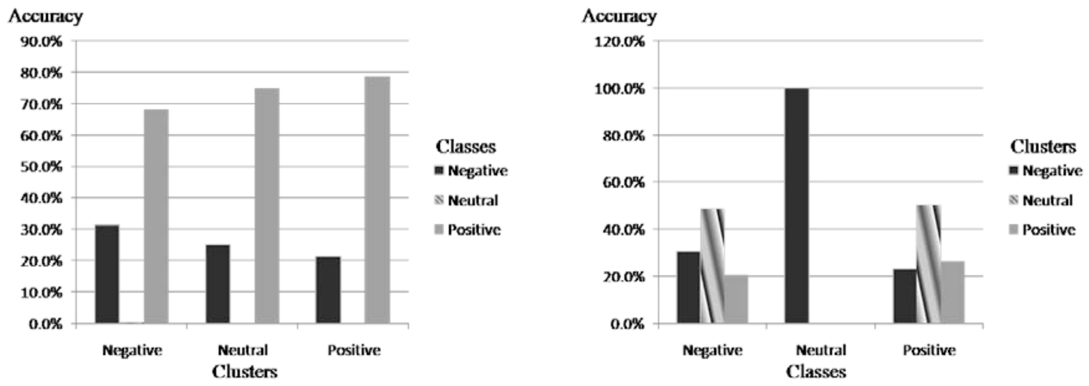
8.5.2.1 Clustering Testing Data

We used K-means method for clustering testing data into 3 clusters. Different methods are used as well for initializing the centriods as discussed in next subsections.

8.5.2.1.1 Initialized by Using K-means++

When conducting this experiment, the centriods are initialized by using K-means++ method. We run this experiment 11 times to find confidence interval with confidence equals 95%. The maximum reported result was 27.9% and 23.8% for accuracy and macro-average F1-score respectively. While, the average was 24.7% and 20.8% for accuracy and macro-average F1-score respectively with a confidence interval equals ±1.9% for both accuracy and macro-average F1-score. For more details, refer to Table II.25 in Appendix II to see all numerical values that are provided when conducting this experiment.

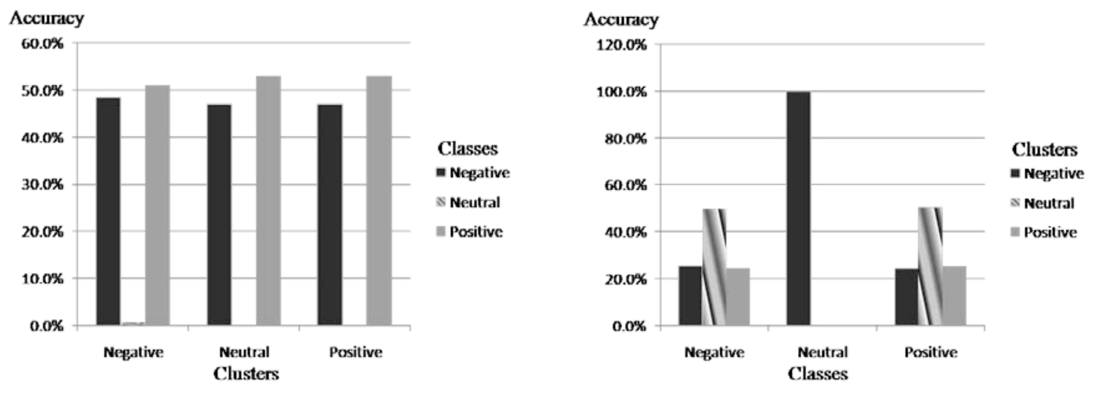
This poor result shows that there is a cluster contains only one data point and majority of data points are assigned incorrectly to the clusters. Experiment result provided from a sample run (accuracy equals 27.6%) is shown in Figure 8.15. As shown in the figure, the neutral cluster contains only one data point which belongs to negative class. Additionally, majority of data points included in negative and positive clusters belong actually to neutral class. For more details, refer to Table II.26 in Appendix II to see all numerical values that are provided when conducting this experiment.



(a) (b)

Figure 8.15: Distribution of data points provided when applying K-means while centroids are initialized by using K-means++.

To show the results more clearly, we run this experiment again 10 times and plotted the average distribution as shown in Figure 8.16. We used Braycurtis distance measure for achieving this experiment. For more details, refer to Table II.27 in Appendix II to see all numerical values that are provided when conducting this experiment.



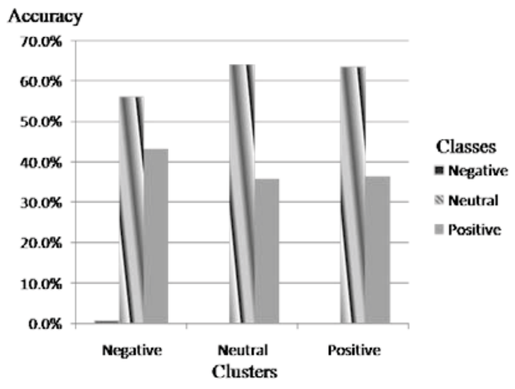
(a) (b)

Figure 8.16: Average of distributions provided when clustering testing data by applying K-means while centroids are initialized by using K-means++.

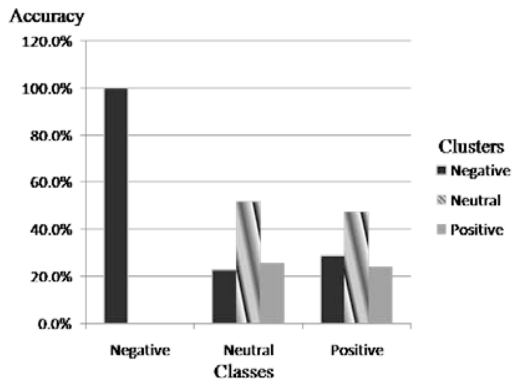
8.5.2.1.2 Randomly Initialization

We developed another experiment to test efficiency of initializing centroids. The centroids are initialized fully randomly from the whole testing data. The classification accuracy is increased in comparison with the previous experiment which uses K-means++ algorithm. We run this experiment 10 times to find confidence interval with confidence equals 95%. The maximum achieved result was 44.9% and 30.4% for accuracy and macro-average F1-score respectively. The average values are 33.6% and 25.3% for accuracy and macro-average F1-score respectively with confidence interval that is equal to ±5.1% for the accuracy and ±1.8% for the macro-average F1-score. For more details, refer to Table II.28 in Appendix II to see all numerical values that are provided when conducting this experiment.

Figure 8.17 shows the distribution of data points among the clusters when conducting a specific experiment run. For more details, refer to Table II.29 in Appendix II to see all numerical values that are provided when conducting this experiment. Figure 8.18 illustrates average of distributions among the clusters and classes. For more details, refer to Table II.30 in Appendix II to see all numerical values that are provided when conducting this experiment.

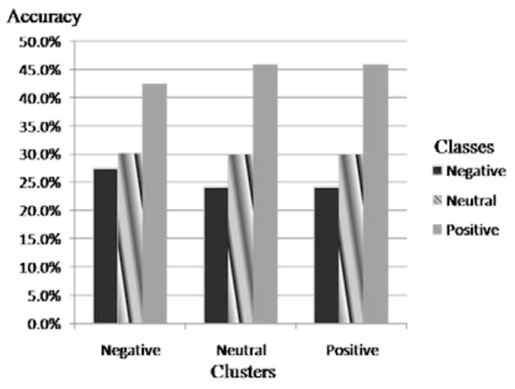


(a)

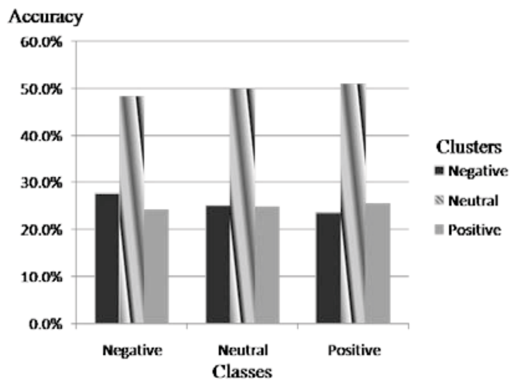


(b)

Figure 8.17: Distribution of data points provided by using K-means while centroids are initialized fully randomly.



(a)

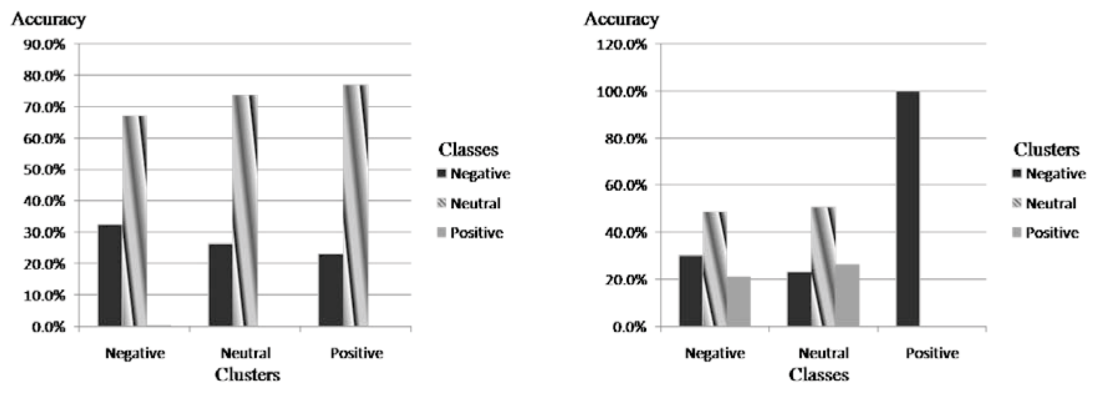


(b)

Figure 8.18: Average of distributions provided when clustering testing data by using K-means while centroids are initialized fully randomly.

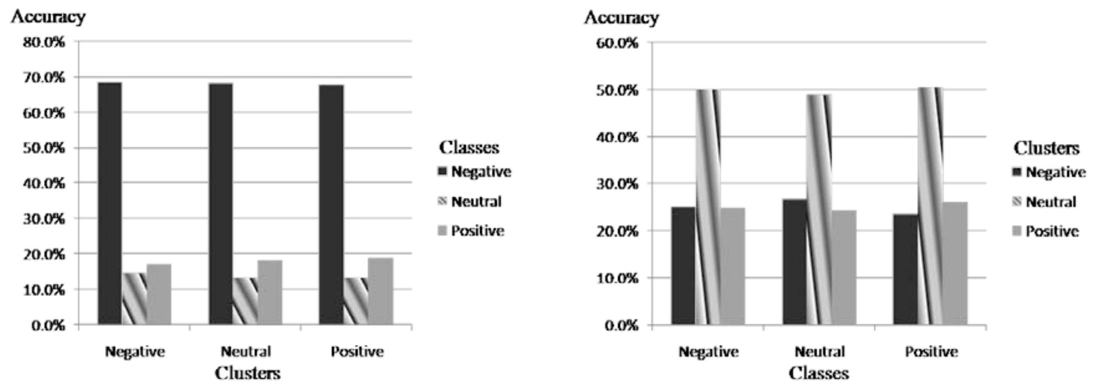
8.5.2.1.3 Initialized at Random From Corresponding Classes

To improve the performance, we initialized randomly each centroid from the actual corresponding class. Figure 8.19 illustrates the results when the classification accuracy was 44.9%. For more details, refer to Table II.31 in Appendix II to see all numerical values that are provided when conducting this experiment. To make this figure clearer, we run this experiment 10 times and calculated the average as illustrated in Figure 8.20. For more details, refer to Table II.32 in Appendix II to see all numerical values that are resulted from these experiments.



(a) (b)

Figure 8.19: Distribution of data points provided when clustering testing data by using K-means while centroids are initialized randomly from the corresponding classes.



(a) (b)

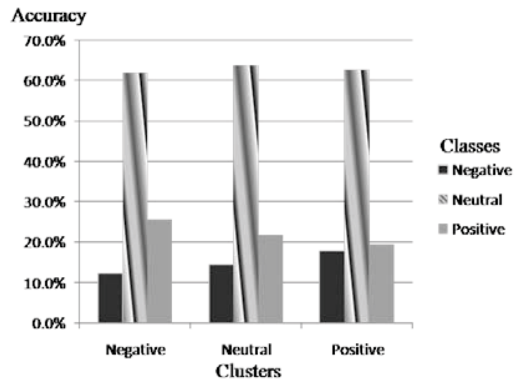
Figure 8.20: Average of distributions provided when clustering testing data by using K-means while centroids are initialized randomly from the corresponding classes.

To find confidence interval with confidence equals 95%, we run again this experiment 33 times. The maximum provided performance was 51.2% and 30.4% for accuracy and macro-average F1-score respectively. The averages are 36.8% and 22.0% for accuracy and macro-average F1-score respectively with confidence interval that is equal to ±4%

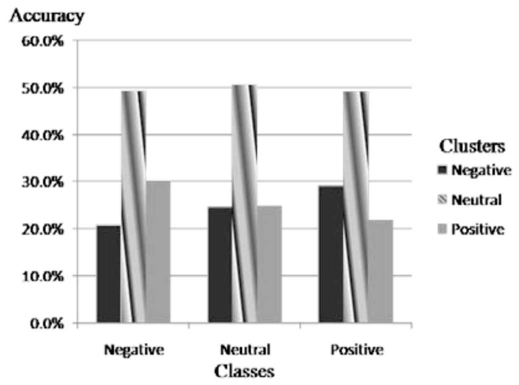
for the accuracy and ±1.7% for the macro-average F1-score. For more details, refer to Table II.33 in Appendix II to see all numerical values that are provided when conducting this experiment.

8.5.2.2 Clustering Training Data

K-means method is used also for clustering the training data into 3 clusters. Figure 8.21 shows the result provided when initializing each centroid fully randomly while the accuracy was 39.8%. For more details, refer to Table II.34 in Appendix II to see all numerical values that are provided when conducting this experiment. To find confidence interval with confidence equals 95%, we run again this experiment 33 times. The maximum achieved performance was 43.9% and 39.3% for accuracy and macro-average F1-score respectively. The averages are 34.4% and 30.1% for accuracy and macro-average F1-score respectively with confidence interval that is equal to ±2.1% for accuracy and ±1.3% for macro-average F1-score. For more details, refer to Table II.35 in Appendix II to see all numerical values that are provided when conducting this experiment.



(a)

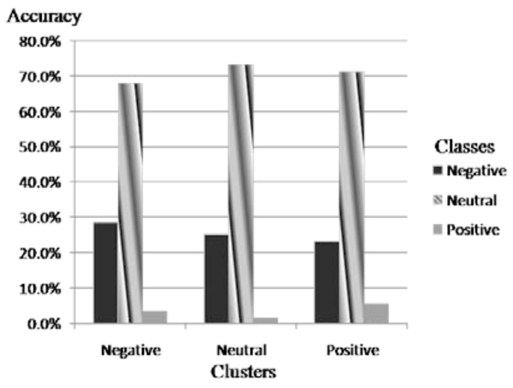


(b)

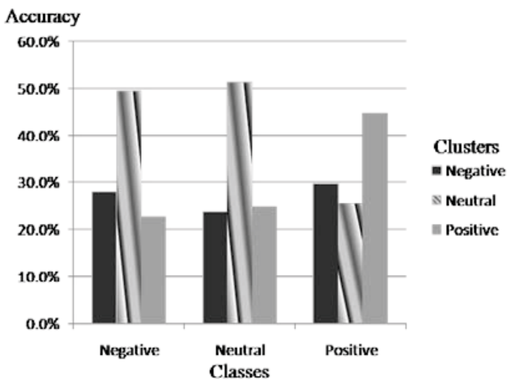
Figure 8.21: Distribution of data points provided when clustering training data by using K-means while centroids are initialized fully randomly.

Figure 8.22 shows the result provided when initializing randomly each centroid from its corresponding class while the classification accuracy was 45.1%. For more details, refer to Table II.36 in Appendix II to see all numerical values that are provided when conducting this experiment. To find confidence interval with confidence equals 95%, we run again this experiment 33 times.

The maximum achieved performance was 51.7% and 39.3% for accuracy and macro-average F1-score respectively. The averages are 42.9% and 32.1% for accuracy and macro-average F1-score respectively with confidence interval that is equal to ±1.7% for the accuracy and ±1.3% for the macro-average F1-score. For more details, refer to Table II.37 in Appendix II to see all numerical values that are provided when conducting this experiment. We reported only distinct results since many runs provide similar results.



(a)



(b)

Figure 8.22: Distribution of data points provided when clustering training data by using K-means while centroids are initialized randomly from the corresponding classes.

8.5.2.3 Clustering testing data after clustering training data

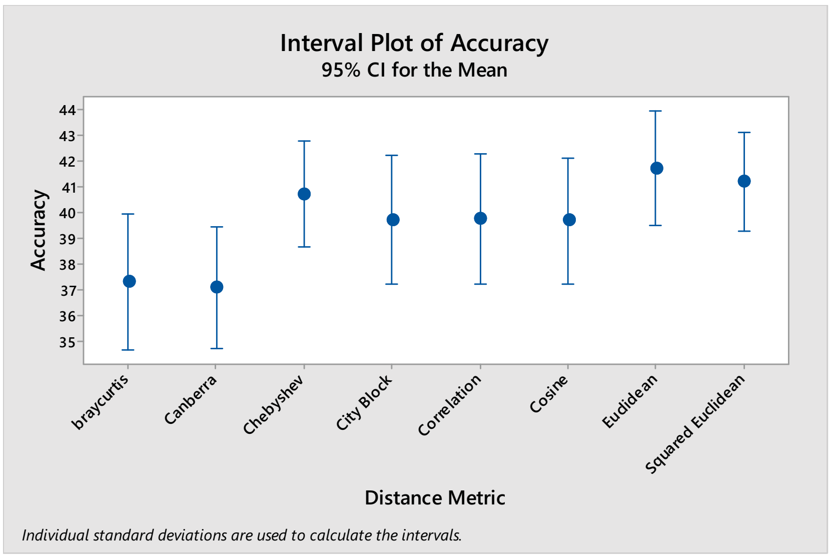
This section discusses performance of cluster training data and then using the resulted centroids for clustering testing data. Result of clustering training data is provided by initializing randomly each centroid from its corresponding class. We used here this method for initializing the centroids because it provided competitive results as discussed in pervious sections. We also applied different distance measures[8] to check efficiency of changing distance measures. We used distance measures: Braycurtis, Canberra, Chebyshev, City Block (Manhattan), Correlation, Cosine, Euclidean, and Squared Euclidean. We selected these distance measures because they are suitable for working with high dimensional data. We noticed also that these distance measures generated better results in comparison with other distance measures evaluated in our work. This result is compatible with many findings shown in the literature.

We run each experiment 33 times to calculate confidence interval with confidence equals 95%. Figure 8.23 previews interval plot for illustrating confidence intervals when changing distance measure. This figure shows clearly that using different measure measures will affect on classification accuracy since there are some mean values do not fall inside confidence intervals of other ones. For more details, refer to tables from Table II.38 into Table II.45 in Appendix II to see all numerical values that are resulted from these experiments.

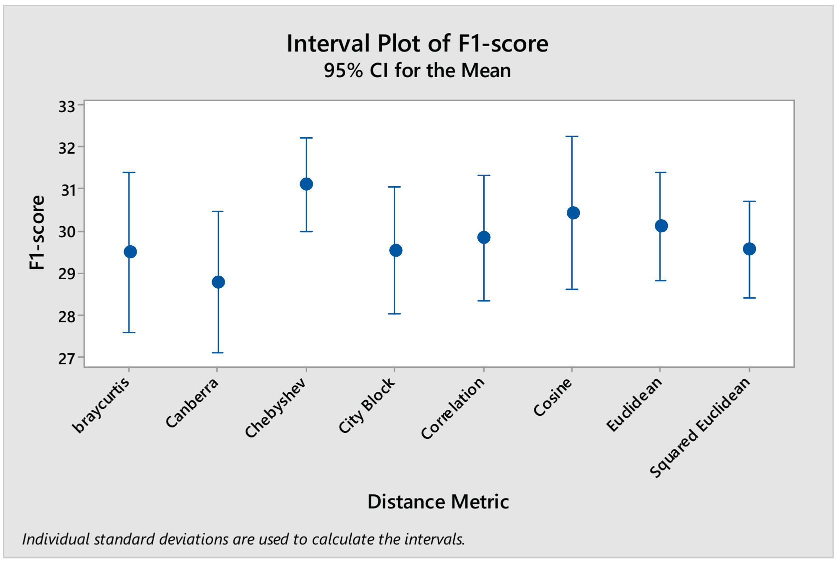
Based on these experiment results, Cosine distance measure provided the best results. This result is compatible with many findings shown in the literature since we use word2vec embeddings based on Cosine distance similarities. The best achieved result was 50.7% and 41.5% for accuracy and macro-average F1-score respectively. The averages are 39.7% and 30.4% for accuracy and macro-average F1-score respectively with confidence interval that is equal to ±2.4% for the accuracy and ±1.8% for the macro-

[8] http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.DistanceMetric.html#

average F1-score. For more details, refer to Table II.43 in Appendix II to see all numerical values that are provided when conducting this experiment.



(a)



(b)

Figure 8.23: Effect of using different distance measures: a) Accuracy, b) F1-score.

Using Euclidean distance measure provided also the same maximum accuracy (50.7%) achieved by using Cosine distance measure but the macro-average F1-score was low that is equal to 39.5%. This result is compatible also with many findings shown in the literature since many works recommend to using Euclidean distance measure. The average of accuracies is equal to 41.7% while the average of macro-average F1-scores is equal to 30.1%. The confidence interval in this case is better which is equal to ±2.2% for the accuracy and ±1.3% for the macro-average F1-score. For more details, refer to Table II.44 in Appendix II to see all numerical values that are provided when conducting this experiment.

It is noteworthy that using City Block (Manhattan) distance measure provided the best macro-average F1-score that is equal to 41.8%. While, it provides the second maximum accuracy that is equal to 50.1%. However, the average of accuracies is similar to the case of using Cosine distance measure (39.7%) while the average value of macro-average F1-scores is low which is equal to 29.5%. The confidence interval equals ±2.5% for the accuracy and ±1.5% for the macro-average F1-score. For more details, refer to Table II.41 in Appendix II to see all numerical values that are provided when conducting this experiment.

8.6 Improved Semi-supervised Target-Dependent Sentiment Classification

We evaluated performance of applying our technique proposed for using target-dependent sentiment classification with partially labeled data. The technique employs self-training with SVM classifier (or other classifiers such as logistic regression) by using probabilistic confidence. Table II.46 in Appendix II shows experiment results when setting the threshold of probability to 0.9.

We changed the threshold when setting ratio of labeled data to 0.45. We noticed that setting threshold to 0.9 provides the best results. The maximum reported accuracy was 72.25%. For more details, refer to Table II.46 in Appendix II to see all numerical values that are provided when conducting this experiment. Figure 8.24 illustrates experiment results as interval plot. We can claim that the provided performance has a high confidence since each confidence interval is so small.

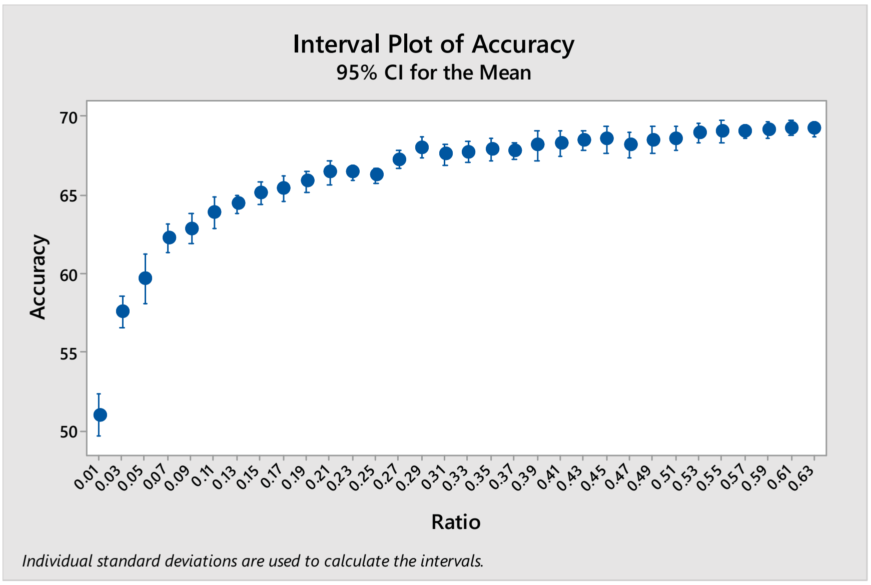


Figure 8.24: Effect of changing ratio of labeled data when applying improved semi-supervised target-dependent sentiment classification.

8.7 Discussion

We deduced form experiment work that using unscaled data provides poor results. Unscaled data is revealed when using different scales and ranges when extracting features. Thereby, it will complicate classification task. Thus, for improving the performance, we normalized the data to the same scale before applying machine learning techniques.

It is noteworthy that labeling models behave differently in comparison with other evaluated semi-supervised techniques. The periods of confidence intervals are large which means that the confidence level is low. While, behavior of other evaluated semi-supervised learning techniques (including our proposed technique) are completely different and work conversely. Our explanation for these phenomena is based on the implementation scope of labeling models. Labeling models build structured tree based on the training data which make training phase precise and sensitive to the data points included in training set. Thus, increasing ratio of labeled data will increase effect of overfitting and decrease the accuracy.

Using probabilistic confidence outperform the other method which uses distance confidence when applying self-training with logistic regression. It is an expected result since calculating distance from each data point to the decision boundary is a difficult task. Calculating each distance is affected by different factors such as the measure used for calculating the distances and number of dimensions that represents data points. While, calculating probabilistic confidence is based on the performance of the classifier. As a result of this, using probabilistic confidence is better than using distance confidence.

Experiment results show clearly that semi-supervised K-means gives better results in comparison with original K-means method. Semi-supervised K-means provides better results since using labeled data helps in initializing centriods within their correct corresponding classes. As a result of this, these good locations enable K-means method to converge usually to better results.

In general, using data clustering methods for classifying the dataset performs poor results. Thereby, data clustering is not suitable for classifying the used dataset and we should use other machine learning techniques for achieving this task. This poor result is provided because unsupervised learning methods work better with well separated classes. While, the used dataset includes three sentimental classes that are overlapped and their sizes are not equal. Additionally, applying unsupervised learning techniques is suitable for low dimensional data while the used dataset has large number of dimensions.

Table 8.1 describes all techniques used for making comparisons. All compared supervised learning models are reported by Tang et al. [127] except SSWE which is proposed by Tang et al. [76] and reported by Vo and Zhang [96] as comparable model. The rest of Table 8.1 presents a description to all evaluated semi-supervised learning techniques used for making comparisons with our proposed solution.

Table 8.2 shows a summary of experiment results for comparing proposed solution with previous related works in the state of the art. The last part in the table illustrates accuracy and macro-average F1-score of predicating sentiment polarities. The reported results indicate to the highest accuracy and macro-average F1-score that are reported when training each model with the lowest ratio of labeled data.

Our solution proposed for improving the performance of target-dependent Sentiment classification (ImproveSelfTrP) using partially labeled data outperforms all evaluated semi-supervised learning techniques as shown in Table 8.2. The proposed technique does not provide the best macro-average F1-score score in comparison with S3VMOVR.

However, the proposed technique outperforms (in terms of both accuracy and macor-F1 score) TC-LSTM which has been published recently.

Table 8.1: Description of all compared methods used for target-dependent sentiment classification.

Method	Description	Class
SSWE	Sentiment-specific word embedding model [76].	S
SVM-indep	SVM classifier uses only target-independent features.	S
SVM-dep	SVM classifier uses target-independent features concatenated with target-dependent features provided by Jiang et al. [98].	S
RecursiveNN	Standard RNN with target-dependent dependency tree [93].	S
AdaRNN-w/oE	Adaptive recursive neural network (RNN) [93].	S
AdaRNN-w/E	Adaptive recursive neural network (RNN) [93].	S
AdaRNN-comb	Adaptive recursive neural network (RNN) [93].	S
Target-dep	SVM uses rich target-independent and target-dependent features [96].	S
Target-dep+	SVM uses rich target-independent, target-dependent, and sentiment lexicon features [96].	S
LSTM	Long short-term memory model (recurrent neural network) uses Glove vector. It classifies target-dependent sentiment based on target independent strategy [127].	S
TD-LSTM	Target-Dependent LSTM [127].	S
TC-LSTM	Target-Connection LSTM [127].	S
Bi-GRU	Bi-directional gated recurrent unit for target-dependent sentiment classification [131]	S
SK-means	Semi-supervised K-means method with Cosine distance measure (which performs better results than Euclidian distance measure).	SM
LabelProK	Label propagation by using kNN kernel.	SM
LabelProR	Label propagation by using RBF kernel.	SM
LabelSpK	Label spreading by using kNN kernel.	SM
LabelSpR	Label spreading by RBF kernel.	SM
S3VMOvOVote	QN-S3VM with OVO strategy. The voting strategy is used to select the most dominant perdition.	SM
S3VMOvR	QN-S3VM with OVR strategy.	SM
SelfTrH	Self-training with SVM method that uses distance from the hyperplane for calculation confidence level. The used formula is inspired by work [128].	SM
SelfTrP	Self-training with SVM method that uses prediction probability for calculating prediction confidence.	SM
ImproveSelfTrP	Our proposed technique that improves self-training with SVM by using prediction probability for calculating prediction confidence	SM

Class: S=Supervised learning technique, **SM**= Semi-supervised learning technique.

Additionally, our proposed semi-supervised learning solution provides comparative accuracy in comparison with previous related supervised learning methods in the state of the art. Moreover, the proposed solution provides the maximum accuracy which is achieved also by a very recent work used a deep learning method (Bi-GRU). It is interesting as well to clarify that the proposed semi-supervised learning technique achieved the maximum classification accuracy when using only 45% of labeled data. Whereas, learning the evaluated methods with partially labeled data may decrease the classification accuracy. The proposed technique provides the highest accuracy with less number of labeled (only 45%) in comparison with other methods such as S3VMOVR which used 63% of labeled data.

Table 8.2: Summary of different techniques proposed for target-dependent sentiment classification.

Method, year	**Setting**	**Acc**	**Macro-F1**	**Labeling Ratio**
SSWE, 2014		62.4	60.5	100%
SVM-indep, 2011		62.7	60.2	100%
SVM-dep, 2011		63.4	63.3	100%
RecursiveNN, 2014		63.0	62.8	100%
AdaRNN-w/oE, 2014		64.9	64.4	100%
AdaRNN-w/E, 2014		65.8	65.5	100%
AdaRNN-comb, 2014		66.3	65.9	100%
Target-dep, 2015		69.7	68.0	100%
Target-dep+, 2015		71.1	69.9	100%
LSTM, 2016		66.5	64.7	100%
TD-LSTM, 2016		70.8	69.0	100%
TC-LSTM, 2016		71.5	69.5	100%
Bi-GRU, 2018		72.3	70.5	100%
SK-means	Cosine distance measure	46.8	43.0	37%
LabelProK	kNN kernel, neighbours #=1	56.4	53.6	1%
LabelProR	RBF kernel, Gamma =0.07	60.8	55.4	7%
LabelSpK	kNN kernel, neighbours #=7	59.8	53.6	27%
LabelSpR	RBF kernel, Gamma= 0.19	61.4	56.6	5%
S3VMOVOvote	Linear kernel♠, Lamda=0.045	70.5	68.4	61%
S3VMOVR	Linear kernel♠, Lamda=0.025	71.7	70.0	63%
SelfTrH	*C*=0.009, Threshold=0.81	70.8	67.9	59%
SelfTrP	*C*=0.009, Prob Threshold=0.9	72.1	69.5	45%
ImproveSelfTrP ♦	*C*=0.009, *P*=0.9	72.3	69.7	45%

♠ Using RBF kernel does not provide classification accuracy more than 50%.
♦ Proposed solutions

It is noteworthy that our proposed technique did not provide the best macro-average F1-score. Our explanation for this result tends to the nature of used dataset which has a large class of neutral sentiment polarity. Our proposed technique detects many neutral tweets correctly while it may misclassify some positive or neutral ones. As a result of this, the classification accuracy increased while the macro-average F1-score does not match this improvement.

The proposed technique works better when classifying initially neutral tweets incorrectly. In this case, the proposed technique will correct the sentiment polarity by selecting the sentiment that is corresponding to second maximum value of the three prediction probabilities (P_+, P_-, and P_o). Thereby, the technique predicts correctly neutral sentiment polarity since neutral polarity leads usually to the second maximum of prediction probabilities. While, positive and negative tweets lead alternately to the lowest or largest prediction probability since these polarities are clearer than neutral polarity which may confuse even the experts.

It deserves attention that S3VMOVR outperforms all semi-supervised learning methods in terms of macro-average F1-score. S3VMOVR provides competitive results but it is not robust since their provided results are sensitive to setting randomly a parameter used for achieving Quasi-Newton optimization. While, applying our proposed technique does not need to set randomly any parameter and converges always to the same results when using same ratio of labeled data. Moreover, using QN-S3VM needs to set two additional parameters (Lamda and LamdaU).

8.8 Conclusion

We evaluated numerous machine learning techniques and proposed new methods for improving the performance of target-dependent sentiment classification. We also investigated in efficiency of employing semi-supervised techniques and proposed a new technique for applying target-dependent sentiment classification by using partially labeled data.

Based on our statistical analysis, we conclude that using different techniques provide different statistical results. Thus, we reject the third hypothesis in our dissertation which estimates that there is no statistical difference among the proposed techniques. Moreover, experiment results provided when evaluating different semi-supervised learning teachings show that models of label propagation and spreading provide low confidential results while semi-supervised K-means provides medium confidential results. Whereas, self-training, QN-S3VM, and our proposed technique provide high confidential results. The experiment results show also that semi-supervised K-means provides the worst performance in comparison with other semi-supervised learning techniques.

The proposed technique outperforms existing semi-supervised learning techniques when solving problem of target-dependent sentiment classification. Moreover, using our proposed semi-supervised learning technique performs competitive results in comparison with other related supervised learning techniques that are proposed in the state of the art. We also conclude that using semi-supervised methods outperform unsupervised leaning techniques. For example, semi-supervised K-means outperforms K-means method.

We also conclude that semi-supervised K-means method performs combative results in comparison with unsupervised learning methods (K-means and Birch). Thus, extending this work by employing semi-supervised K-means for developing a new technique may improve the performance of target-dependent sentiment classification. We also suggest selecting Cosine, Euclidean, and City Block distance measures when conducting the experimental work since using these measures provide high performance.

CHAPTER 9

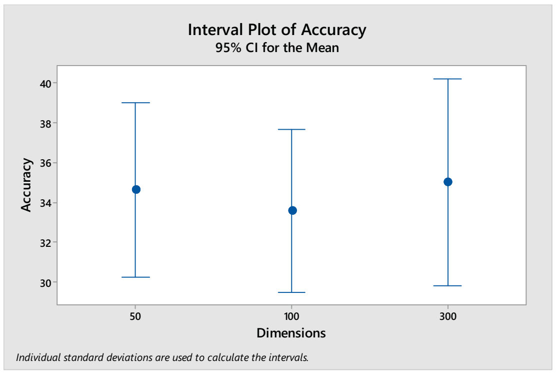
EXPERIMENT RESULTS: IMPACT OF DIMENSION REDUCTION WITH TARGET-DEPENDENT SENTIMENT ANALYSIS

Various experiment results are shown in this chapter for illustrating performance of applying PCA and LDA for reducing number of dimensions in DatasetA. Next sections describe impact of using dimension reduction when applying data clustering. Performance of combining LDA with other supervised learning classifiers is shown also in the sequel.

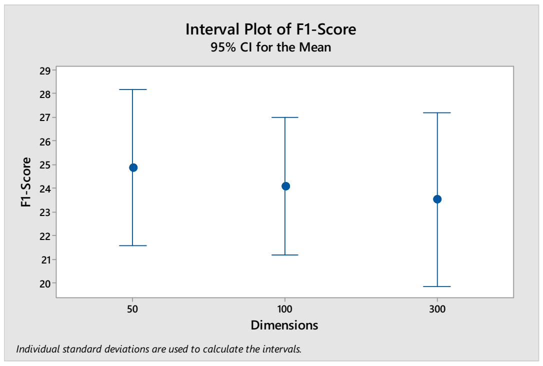
9.1 K-means with PCA

Since K-means method works better with low dimensional data, we checked performance of applying K-means method after reducing number of dimensions by using PCA. PCA tries to make feature selection to reduce number of dimensions. We run this experiment 105 times and reported only non redundant values. We applied K-means method firstly to the training data and then the resulted centroids are used to classify testing data by assigning each data point to the closest centroid. We achieved this experiment by using Cosine distance measure since it provides good results based on our experiment results as shown in Section 8.5.2.3.

Figure 9.1 shows interval plot of 95% confidence for the mean of classification accuracies and F1-scores. The figure illustrates clearly that changing number of dimensions with PCA has no effect since each mean value is located inside the confidence interval of the others. We set number of dimensions to 50, 100, and 300. We could not make number of reduced features more than 692, since the maximum number of tweets included in testing data is 692. The maximum reported performance was 50.1 and 34.5% for classification accuracy and macro-average F1-score respectively when number of dimensions equals 50 and 300. For more details, refer to Table III.1 in Appendix III to see all numerical values that are provided when conducting this experiment.



(a)



(b)

Figure 9.1: Effect of using K-means with PCA: a) Accuracy, b) F1-score.

9.2 Semi-supervised K-Means with PCA

We evaluated performance of reducing number of dimensions when applying semi-supervised K-means. We set number of dimensions to 300 and 600 dimensions and changed ratio of labeled data. Ratio of labeled data is changed from 0.01 to 0.63 with increasing step that is equal to 0.02. At each ratio of labeled data we run the experiment four times. The maximum achieved accuracy was 50.1% with ratio that is equal to 0.01

when selecting number of dimensions equals 300. The same maximum accuracy is reported also when setting number of dimensions to 600 with ratio equals 0.11.

Figure 9.2 illustrates results of applying semi-supervised K-means with PCA when setting number of dimensions to 300. We noticed clearly that there is a change in the accuracy when modifying ratio of labeled data. However, the achieved accuracies are very low. For more details, refer to Table III.2 in Appendix III to see all numerical values that are provided when conducting this experiment.

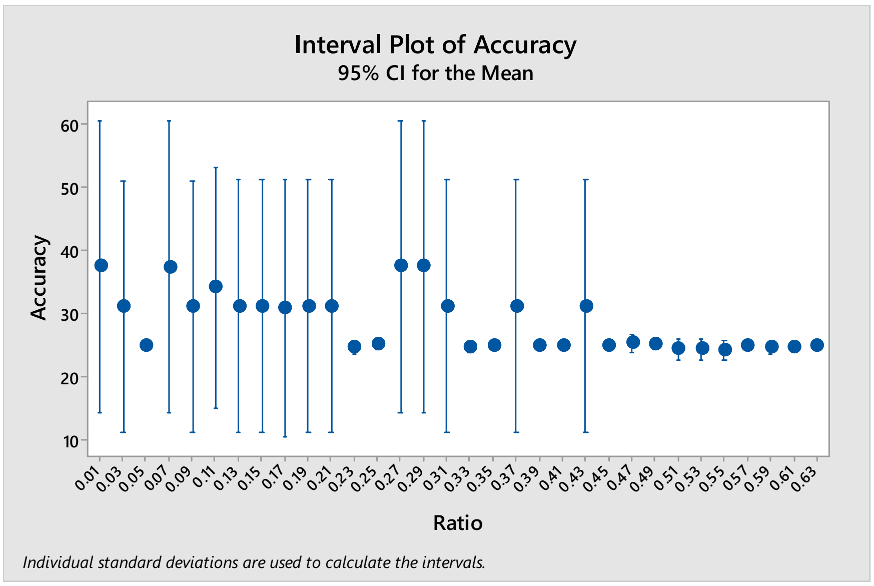


Figure 9.2: Effect of changing ratio of labeled data when applying semi-supervised K-means with PCA (dims#=300).

Figure 9.3 illustrates results of applying semi-supervised K-means with PCA when setting number of dimensions to 600. We noticed clearly that there is a change in classification accuracy when modifying ratio of labeled data. However, the achieved classification accuracies are very low. For more details, refer to Table III.2 in Appendix III to see all numerical values that are provided when conducting this experiment. Experiment results show that changing number of dimensions between 300 and 600 has no effect since each mean value is located inside the confidence interval of the other. The best accuracy was achieved with ratios 0.01 and 0.11 when setting number of dimensions to 300 and 600 respectively.

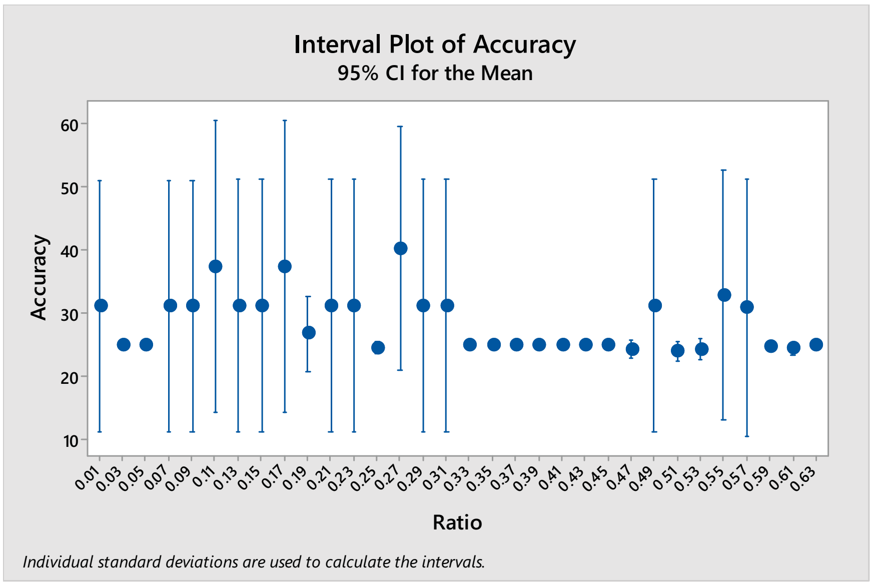


Figure 9.3: Effect of changing ratio of labeled data when applying semi-supervised K-means with PCA (dims#=600).

9.3 Supervised Classification with K-means and Dimension Reduction

This section discusses different experiments that are conducted to show performance of our solution proposed for exploiting dimension reduction. The proposed solution uses semi-supervised K-means method with LDA for improving classification accuracy as described in Section 5.5.

9.3.1 Using Different Distance Measures

We evaluated performance of using different distance measures when classifying testing data after learning the proposed model by using training data as shown in Table 9.1. The maximum reported performance was 92.1% and 91.8% for classification accuracy and macro-average F1-score respectively. This maximum performance is achieved when using City Block distance measure. It is noteworthy that City Block distance measure provided the best macro-average F1-score as described in Section 8.5.2.3.

Table 9.1: Effect of using different distance measures when applying K-means with LDA.

Distance Measure	**Accuracy**	**Macro-F1**
Norm	91.5	91.0
Braycurtis	90.3	89.9
Canberra	88.3	88.3
Chebyshev	91.8	91.3
City block	92.1	91.8
Correlation	65.9	51.5
Cosine	89.3	89.0
Euclidean	91.5	91.0
Square Euclidean	91.5	91.0

9.3.2 Using Different Clustering Scenarios

To show efficiency of our proposed solution, we applied it to different data parts of the dataset. Next subsections describe all experiments and report a summary of results provided by using City Block distance measure. We use four scenarios for showing the performance. In the first scenario, we applied both clustering (learning) and classifying (testing) phases to training data. While in the second scenario, we use only testing data for both clustering (learning) and classifying (testing) phases. In the third scenario, we apply firstly clustering on testing data and then classify training data. The last scenario is a traditional behavior used for learning any model. In this scenario, we apply firstly clustering phase to training data and then classify testing data. The next subsections show results of applying each scenario individually.

9.3.2.1 Clustering and Classify Training Data

When conducting this experiment, we use only training data for both clustering (learning) and classifying (testing) phases. We run this experiment 51 times and reported all results in Table III.3 in Appendix III. The maximum achieved performance was 91.2% and 91.0% for classification accuracy and macro-average F1-score respectively. Table 9.2 shows a summary of the provided results along with confidence interval of 95% confidence.

Table 9.2: Summary of clustering and classifying training data when applying K-means with LDA.

	Accuracy	Macro-F1
Max	91.2	91.0
Min	4.3	4.0
AVRG	88.2	88.1
CI High	92.4	92.1
CI Low	84.0	84.0

9.3.2.2 Clustering and Classifying Testing Data

When conducting this experiment, we use only testing data for both clustering (learning) and classifying (testing) phases. We run this experiment 51 times and reported all results in Table III.4 in Appendix III. The maximum achieved result was 92.6% and 92.3% for classification accuracy and macro-average F1-score respectively. Table 9.3 shows a summary of the provided results along with confidence interval of 95% confidence.

Table 9.3: Summary of clustering and classifying testing data when applying K-means with LDA.

	Accuracy	F1-score
Max	92.6	92.3
Min	26.9	34.1
AVRG	91.3	91.1
CI High	93.9	93.4
CI Low	88.8	88.8

9.3.2.3 Clustering Testing Data and Classify Training Data

When conducting this experiment, we use testing data for clustering (learning) and then we use training data for classifying phase. We run this experiment 51 times and reported all results in Table III.5 in Appendix III. The maximum achieved performance was 90.8%

and 90.4% for classification accuracy and macro-average F1-score respectively. Table 9.4 shows a summary of the provided results along with confidence interval of 95% confidence.

Table 9.4: Summary of clustering testing data and classifying training data when applying K-means with LDA.

	Accuracy	F1-score
Max	90.8	90.4
Min	6.5	6.3
AVRG	85.3	85.3
CI High	90.7	90.3
CI Low	80.0	80.4

9.3.2.4 Clustering Training Data and Classify Testing Data

When conducting this experiment, we use training data for clustering (learning) and then we use testing data for classifying phase. We run this experiment 96 times and reported all results in Table III.6 in Appendix III. We run this experiment more than 51 (which used with previous experiments) since we applied here the traditional scenario that is designed for learning any model used in literature. The maximum achieved performance was 92.1% and 91.8% for classification accuracy and macro-average F1-score respectively. Table 9.5 shows a summary of the provided results along with confidence interval of 95% confidence.

Table 9.5: Summary of clustering training data and classifying testing data when applying K-means with LDA.

	Accuracy	F1-score
Max	92.1	91.8
Min	25.9	32.6
AVRG	89.3	89.4
CI High	92.0	91.8
CI Low	86.7	87.0

9.4 Combined Supervised Learning Technique

We proposed a technique that combines LAD with supervised learning classifier for improving the performance. We applied LDA for reducing number of feature attributes. Then, we classified the output of LDA by using linear logistic regression. We selected logistic regression method for classifying output of LDA because it provided the best performance in comparison with other supervised learning methods as shown in Table 7.13. The value of *C* parameter which is used for building logistic regression model equals 0.09. The performance of this proposed solution is competitive which is 91.5% and 91.0% for classification accuracy and macro-average F1-score respectively.

We also checked efficacy of applying other classifier to the reduced dimensional data. The combined technique included neural networks for classifying the output of LDA. Using deep learning here provided higher performance. The best achieved result was 91.6% and 91.2% for classification accuracy and macro-average F1-score respectively. The neural network model includes three layers with the best values used when setting the parameters as shown in Table 7.12.

9.5 Discussion

We noticed clearly that dimension reduction makes an obvious change in shapes of sentimental classes (positive, negative, and neutral) included in the used dataset. Using PCA for dimension reduction does not make effective improvement. While, using LDA provides significant improvement in classifying the sentimental classes. After applying LDA for reducing number of dimensions to only 2 dimensions, the shapes of sentimental classes become well separated and can be separated easily by using data clustering or supervised classifier. We used this idea to propose new solutions for improving the performance of target-dependent sentiment classification.

LDA outperforms PCA when reducing number of dimensions because it uses linear learning model with labeled data for finding the best values that represents the new dimensions of each data point. Thus, LDA outperforms PCA which uses only unlabeled data for reducing number of dimensions. Moreover, applying PCA for dimension reduction is not well with the used dataset because it removes important details that represent the tweets.

Using different ratio of labeled data when applying semi-supervised K-means with PCA will change classification accuracy, but all achieved results are worse. While, using LDA with semi-supervised K-means provides competitive results. Our proposed solution which is based on using semi-supervised K-means with LDA provides usually high accuracy but sometimes it converges to worse results. Thus, we can claim that the proposed solution is a metaheuristic technique.

The results converge usually to good solutions because the centriods of K-means are initialized randomly from their correct corresponding classes. However, the location of centriods may be fall sometimes in a complex region which leads to worse clustering solution. We compared performance of our proposed solutions with previous related supervised learning techniques in the state of the art. Table 9.6 describes briefly all compared techniques, while Table 9.7 shows classification accuracy and macro-average F1-scores that are provided by these techniques. For simplicity the reading, we repeated again listing all methods included in Chapter 8 and used for making the comparisons.

Table 9.6: Description of all compared methods.

Method	**Description**
SSWE	Sentiment-specific word embedding model [76].
SVM-indep	SVM classifier uses only target-independent features.
SVM-dep	SVM classifier uses target-independent features concatenated with target-dependent features provided by Jiang et al. [98].
RecursiveNN	Standard recursive neural network with target-dependent dependency tree [93].
AdaRNN-w/oE	Adaptive recursive neural network (RNN) [93].
AdaRNN-w/E	Adaptive recursive neural network (RNN) [93].
AdaRNN-comb	Adaptive recursive neural network (RNN) [93].
Target-dep	SVM classifier uses rich target-independent and target-dependent features [96].
Target-dep+	SVM classifier uses rich target-independent, target-dependent, and sentiment lexicon features [96].
LSTM	Long short-term memory model (recurrent neural network) uses Glove vector. It classifies target-dependent sentiment based on target independent strategy [127].
TD-LSTM	Target-Dependent LSTM [127].
TC-LSTM	Target-Connection LSTM [127].
Bi-GRU	Bi-directional gated recurrent unit for target-dependent sentiment classification [131]
Compind_K-means (Max)	Maximum accuracy which is achieved by using proposed technique that combines LDA with K-means
Compind_K-means (Avg)	Average accuracy which is achieved by using proposed technique that combines LDA with K-means
Compind_LR	Proposed technique that combines LDA with linear regression
Compind_NN	Proposed technique that combines LDA with neural networks

Based on the experiment results, we conclude that the proposed solutions outperform all previous related works. The performance is increased significantly in terms of both classification accuracy and macro-average F1-score. The proposed technique Compind_K-means increases the classification accuracy by about 20% in comparison with accuracy of prominent supervised learning method proposed in the state of the art.

Table 9.7: Comparing different techniques for target-dependent sentiment classification.

Method, year	**Setting**	**Acc**	**F1**
SSWE, 2014		62.4	60.5
SVM-indep, 2011		62.7	60.2
SVM-dep, 2011		63.4	63.3
RecursiveNN, 2014		63.0	62.8
AdaRNN-w/oE, 2014		64.9	64.4
AdaRNN-w/E, 2014		65.8	65.5
AdaRNN-comb, 2014		66.3	65.9
Target-dep, 2015		69.7	68.0
Target-dep+, 2015		71.1	69.9
LSTM, 2016		66.5	64.7
TD-LSTM, 2016		70.8	69.0
TC-LSTM, 2016		71.5	69.5
Bi-GRU, 2018		72.3	70.5
Compind_K-means (Max) ♦		92.1	91.8
Compind_K-means (Avg) ♦		89.3	89.4
Compind_LR ♦	*C*=0.09	91.5	91.0
Compind_NN ♦	Hidden Layers =3, Act='relu', Solv='adam', Eps=0.9	91.6	91.2

♦ Proposed solutions

9.6 Conclusion

We checked effect of using dimension reduction by using PCA and LDA. The experiment results show clearly that LDA outperform PCA when applying dimension reduction. As a result of this, using LDA provided significant improvement in classification accuracy. Thus, we used LDA to develop new solutions for improving the performance of target-dependent sentiment classification. One of proposed solution is a metaheuristic technique that combines unsupervised method (K-means) with LDA. This proposed solution is sensitive to initializing centriods and distance measures. The other proposed solution combines LDA with a supervised learning classifier such as linear logistic regression and neural networks.

Based on the experiment results, we conclude that the proposed solutions outperform significantly all previous related works. The proposed solutions increased the classification accuracy by about 20% over prominent supervised learning method proposed in the state of the art. Thus, we conclude that reducing the used feature attributes will increase significantly accuracy of target-dependent sentiment classification. It is clear also that the proposed technique cannot detect correctly all sentiment polarities in the dataset. However, the proposed solution provides high accuracy in comparison with other related works. Thus, we approve the second hypothesis in our dissertation which estimates that there is no super classifier that can identify correctly all sentiment polarity expressed in micro-blogs.

CHAPTER 10

OPEN DOMAIN TARGETED SENTIMENT ANALYSIS

This chapter describes our proposed context-based analysis system (which meets research objective RO2 in our dissertation) that deals with open domain targeted sentiment among a set of micro-blogs. The chapter presents also new supervised and semi-supervised learning techniques (which meet research objectives RO3 and RO4 in our dissertation) proposed for improving the performance of open domain targeted sentiment classification.

10.1 Context-Based Targeted Sentiment Analysis System

This section describes our proposed context-based analysis system that deals with open domain targeted sentiment among a set of micro-blogs (such as tweets). The next subsection describes our approach in designing the proposed system. The second subsection describes all details required for implementing the system.

10.1.1 The Approach

In this work, we propose a context-based analysis system. This system is capable of detecting targets and most common topics (context) that are discussed among a set of micro-blogs and detecting sentiment polarities toward the topics. To the best of our knowledge, existing systems in the state of the art deal with detecting topics and sentiment polarities expressed in each micro-blog individually [143]. Whereas, our proposed system deals with detecting both topics and sentiment polarities expressed in a set of micro-blogs. Some comparisons between the proposed system and other existing systems are described in the sequel.

Most of existing systems employ context-based analysis for generating features attributes [144] that can be used to improve performance of sentiment analysis systems. While, our objective in this direction is finding the context (topics) discussed among a set of micro-blogs. Additionally, some existing systems use documents (granularity level is document) for detecting topics that are covered in a specific domain such as hotel reviews [145]. While, our goal is developing an open domain analysis system that can detect any topic (such as any named person or organization) among a set of micro-blogs (granularity level is sentence).

Existing systems may use Hashtags entities to facilitate detecting the common topics [146] since the micro-blogs are already grouped by Hashtags entities. Additionally, existing systems may use conversations written by the same user (user level) [147] to facilitate detecting sentiment polarity since each user express usually the same sentiment direction. Our proposed system deals with a more changing situation since it does not use additional information such as Hashtags and conversations written by the same user. Based on the previous discussion, we can claim that our proposed system is the first context-based analysis system that deals with open domain targeted sentiment analysis among a set of micro-blogs. The next subsection describes all details required for implementing the proposed system.

10.1.2 The Architecture

The proposed context-based analysis system can be accomplished by passing through different stages as shown in Figure 10.1. It starts by collecting micro-blogs and building a set of micro-blogs. After collecting micro-blogs, the next step is preprocessing the collected micro-blogs by removing unrelated contents and filtering the text. The next stage is identifying the most common targets in the set of micro-blogs. After that, the system detects topics that are related to the targets and grouping micro-blogs that are belong to the same topic. The final step includes applying classification techniques to classify sentiment expressed in each micro-blog into positive, negative, or neutral. The next subsections describe all details that are included in these stages.

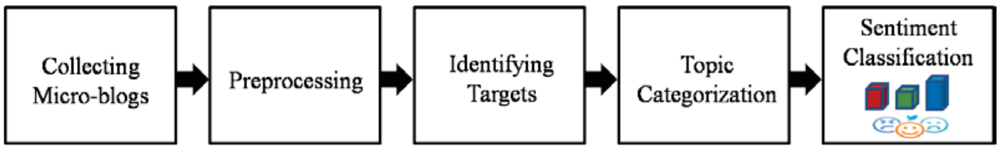


Figure 10.1: Architecture of proposed context-based targeted sentiment analysis system.

10.1.2.1 Collecting Micro-blogs

In this step, various micro-blogs should be collected from different sources to build a set of micro-blogs. This collected set will be used as input to the context-based analysis system. We need to collect a huge amount of micro-blogs to guarantee that the system can find more common targets described in a suitable number of micro-blogs.

10.1.2.2 Preprocessing

The preprocessing step is used to clean text from undesired contents such as user names, pictures, hash-tags, and URLs. This phase includes also filtering the text by removing punctuation, non-letters, short vowels, etc. Additionally, we normalize words by combining words that have different surface forms.

10.1.2.3 Identifying Targets

In this stage, the system detects the most common targets described in a set of micro-blogs. To find targets, we use natural language processing to identify name entities in the micro-blogs. To achieve this task, we use external part of speech (POS) tagger. We should select an accurate POS tagger to improve the performance of the proposed system.

Finding accurate POS tagger for dealing with non English micro-blogs (such as Arabic micro-blogs) is not an easy job because its accuracy is still limited. For example, dealing with Arabic language is still an open research problem because there are many challenges when developing Arabic POS tagger. For example, same Arabic micro-blog may include different dialects.

After extracting name entities from the micro-blogs, we identify the targets from the extracted name entities. We use the tag labels that are provided by using POS tagger for identifying the targets. For example, we select tag labels that stand for proper noun and noun. To make task of detecting targets more accurate, we calculate frequency of phrases that are labeled as proper noun and noun. Calculating frequency helps also in finding the most common targets among the set of micro-blogs.

10.1.2.4 Topic Categorization

After detecting targets, the system groups the micro-blogs based on the detected targets. This step helps in finding micro-blogs that are related to the same target. Then, the system detects topics (and subtopics) that are discussed among each group of micro-blogs discussed same target. Figure 10.2 describes details required for detecting main topics. While, Figure 10.3 describes details required for identifying subtopics. Step 4 describes process of feature engineering which is based on using word2vec embeddings. The output of step 4 is a vector of numerical values which is referred to as a data point.

```
Algorithm of detecting main topics in context-based analysis system:
Inputs: a group of micro-blogs that belong to same target
Output: main topics
1)  Select noun entities from each micro-blog
2)  Count frequency of each noun entity among all tweets
3)  Group micro-blogs based on the most common noun entities (main topics)
```

Figure 10.2: Algorithm of detecting main topics in context-based analysis system.

```
Algorithm of predicting subtopics in context-based analysis system:
Inputs: a group of micro-blogs that belong to same main topic
Output: subtopics
1)  Select noun and adjective entities from each micro-blog
2)  Count frequency of each noun and adjective entity among all tweets
3)  Find most common and important word in each tweet and select the closest word to the main topic.
4)  Convert selected words to word2vec embeddings
5)  Cluster selected word2vec embeddings by using hierarchical clustering method.
6)  Group micro-blogs based on the subtopics
```

Figure 10.3: Algorithm of identifying subtopics in context-based analysis system.

We use hierarchical clustering method for clustering the data since using traditional clustering algorithms such as K-means is not suitable in this case. K-means works better with well separated clusters while our task may include complex clusters. Additionally, K-means cannot cluster groups that have too different sizes. While in our problem, we need to classify even a cluster that includes only one data point. Thus, using hierarchical clustering method is the best choice in this case. To find the optimum number of clusters, we need to use an evaluation measure such as Elbow or Silhouette. The system continues in detecting topics and subtopic until number of frequencies becomes below the threshold.

10.1.2.5 Sentiment Classification

This is the final stage in the proposed context-based targeted sentiment analysis system. There are many alternative methods could be used to classify sentiment polarity expressed in the micro-blogs. The outcome of this step expresses sentiment polarities as one of three options: positive, negative, or neutral. We use here target-dependent sentiment classification since the system already detected targets in pervious stages. However, we can use open domain targeted sentiment classification. In this case, we need to link the detected targets resulted from open domain targeted sentiment classification with the selected targets. It is noteworthy that using target-dependent sentiment classification provides more accurate results with less complexity in implementation.

It is clear that we need to train the classifier by using labeled micro-blogs. We can use supervised learning classifier if we have a large number of labeled micro-blogs. Otherwise, we can use our proposed semi-supervised learning technique for training the classifier with partially labeled micro-blogs. It is important to clarify that using unsupervised learning methods for detecting sentiment polarities will enable us to use only unlabelled micro-blogs. However, the classification accuracy will be inaccurate and we will not be able to validate experiment results.

10.2 Proposed Solutions for Improving Open Domain Targeted Sentiment Classification

Three new solutions are proposed for improving the performance of open domain targeted sentiment classification. The first one is based on combining discrete features with multiple word embeddings. The second solution is based on employing semi-supervised learning by generating feature attributes dynamically. The last proposed solution combines supervised learning with dynamic generation of feature attributes. All details required for implementing the three solutions are included in the sequel.

10.2.1 The Approach

Our goal in this research direction is based on developing sentiment classification system that does not use an external analyzer and can be applied easily to any language (language independent). Thus, we did not use here any NLP tools when building our proposed techniques. Additionally, we did not use sentiment lexicons as exploited in traditional methods. Thereby, developing open domain targeted sentiment system with these restrictions is more changeable.

10.2.2 Feature Engineering

Recent studies use broadly distributed word representations to map text into low dimensional vectors. We depend on this method for extracting features that are used in our work since this method is flexible and can be applied easily to any language. We used specifically a famous form of word embedding called word2vec. Different word2vec embeddings are used from three sources to decrease effect of unseen word2vec embeddings (out-of-vocabulary words). Many unseen word2vec embeddings are revealed because micro-blogs include slang words that could not be represented by word2vec embeddings when using pre-trained word embeddings.

We propose a feature engineering method based on merging more than one word2vec vector that are gathered from different sources. The proposed technique concatenates more than one word2vec vector and normalizes them to generate longer vector that includes more feature attributes. Normalization process makes the feature attributes (numerical values) fall in the same range. Using normalized feature vector improves the performance and helps in merging word2vec embeddings with discrete features. We normalize word2vec vector by applying next formula which makes all numerical values located in the range between -1 and 1.

$$X_{new} = 2\left(\frac{X - X_{min}}{X_{max} - X_{min}} - \frac{1}{2}\right) \qquad (10.1)$$

10.2.3 Supervised Learning of Combined Discrete Features and Multiple Word Embeddings

Using word embeddings improves significantly the performance of open domain targeted sentiment classification. The problem of employing word embeddings in social media analysis systems is revealed when finding word embeddings that represent all words included in the micro-blog. Logically, it is impossible to find word embeddings that represent each word in micro-blog since bloggers usually use slang words. In the ideal case, we can find word embeddings that represent each word included in micro-blog when training the machine learning model. While, we cannot find word embeddings that represents all words included in testing data since we cannot know all words that may be used by bloggers in real life situation.

Of course, existence of unseen word embeddings limits the performance. To decrease effect of unseen words, we proposed a solution based on merging pre-trained word embeddings that are collected from different resources. Thereby, probability of missing word embeddings when representing all words will be decreased. To improve the performance, we also concatenated word embeddings with discrete feature attributes.

The proposed solution uses SVM HMM to take into consideration the relations between words included in each micro-blog. We selected this machine learning method because this research problem can be solved by employing a sequence labeling method. While, it is improper to use traditional classifiers such as SVM. Based on our knowledge, our research is the first work that employs SVM HMM for improving the performance of open domain targeted sentiment classification.

Another reason for choosing SVM HMM comes from its ability to accept numerical (continuous) features, categorical (discrete) features and a combination of them. Moreover, different studies showed efficiency of SVM HMM in comparison with other methods such as CRF [148]. All details of training the proposed technique are illustrated in Figure 10.4.

We use optimization method to find optimum value of *C* parameter. The optimization process is applied by increasing value of *C* parameter gradually. At each selected value of *C* parameter, we test performance of the model by using the development set and calculate "zero/one-error" measure. We selected the best value of *C* parameter that is provided the lowest value of "zero/one-error" measure.

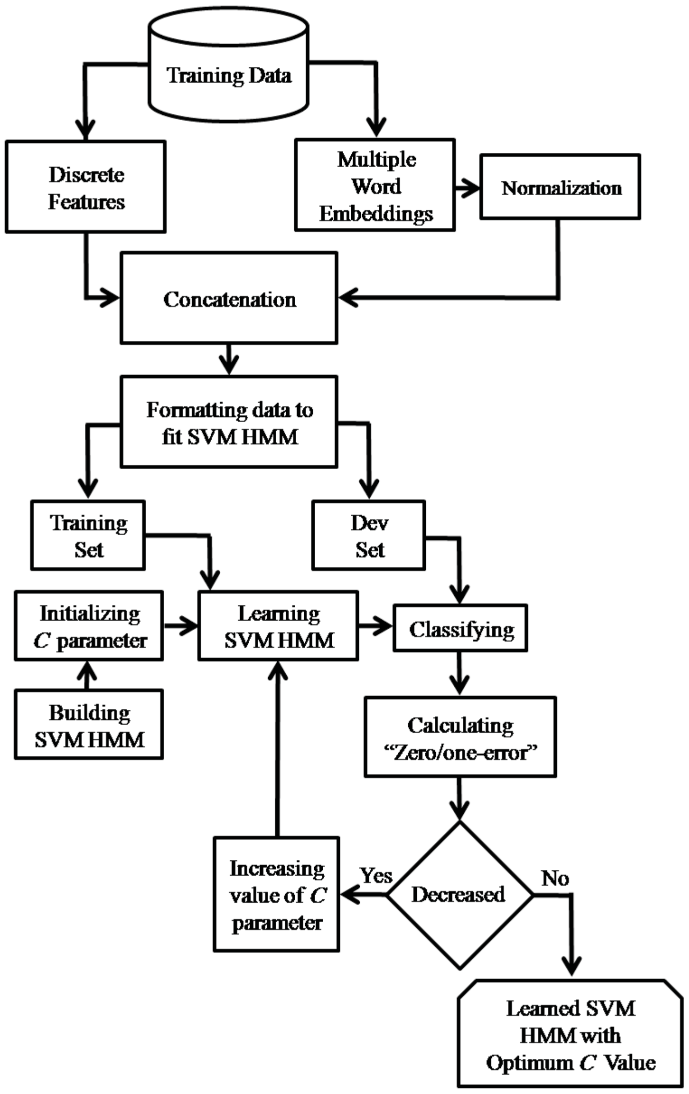


Figure 10.4: Flowchart of training a model that combines discrete features with multiple word embeddings.

Of course, using testing set instead of using development set will provide better values of *C* parameter. While, we use development set in this optimization process to make our proposed solution more realistic. In real problem, we cannot see testing data while we can use development data (which is a part of training data) for testing. When classifying a new unseen micro-blog, we use the trained SVM HMM which is learned by using the best value of *C* parameter. To check the performance, the proposed technique is applied to the testing data and the evaluation measures are calculated for name entity recognition (NER) and sentiment analysis (SA). We used precision, recall, and F1-score because they are used broadly in the literature.

All details of testing the proposed technique are illustrated in Figure 10.5. We collect word2vec embeddings that represents each word in testing data by using the same sources which are selected for training phase. Then, we concatenate multiple word2vec embeddings with the discrete features as illustrated in the figure. Finally, we format the data for fitting the suitable form that is used by SVM HMM (as used in training phase).

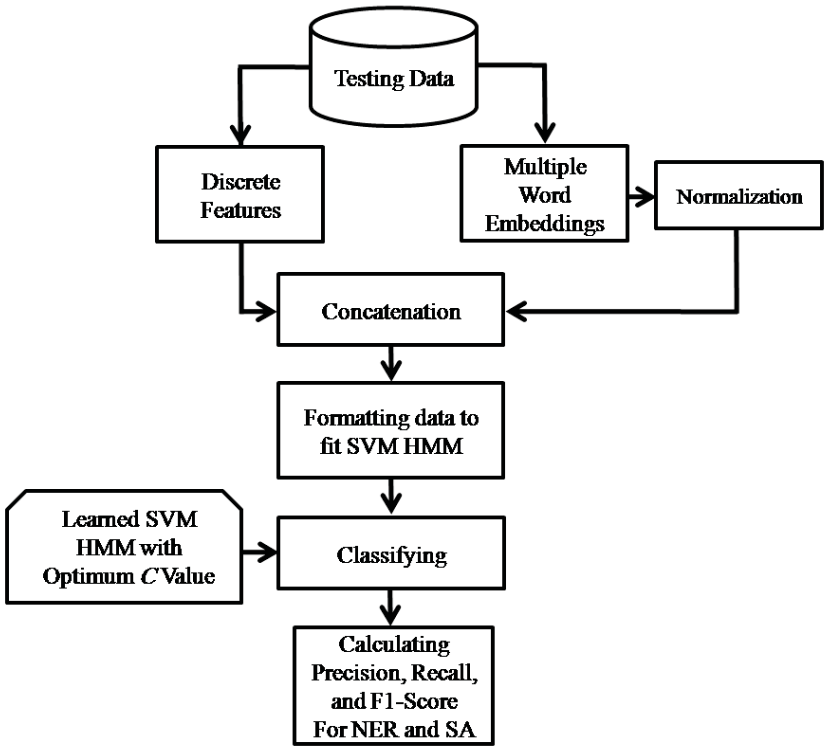

Testing Data
Discrete Features
Multiple Word Embeddings
Normalization
Concatenation
Formatting data to fit SVM HMM
Learned SVM HMM with Optimum *C* Value
Classifying
Calculating Precision, Recall, and F1-Score For NER and SA

Figure 10.5: Flowchart of testing model for combining discrete features with multiple word embeddings.

10.2.4 Semi-supervised Learning with Dynamic Generation of Feature Attributes

A new technique is proposed for employing semi-supervised learning in open domain targeted sentiment classification by using partially labeled data. Based on our knowledge, our solution is the first work that employs semi-supervised learning technique for open domain targeted sentiment classification. The proposed technique is based on improving the performance by generating more attributes in the horizontal level that represents data points. Thereby, the technique adds more attributes to each feature vector that represents each word included in micro-blogs. The proposed solution is based specifically on the level of feature attributes because open domain targeted sentiment classification deals with word level instead of micro-blog level.

Using traditional semi-supervised learning techniques is not suitable for open domain targeted sentiment classification because these techniques ignore the relations between each sequence of words in micro-blogs. Thus, using SVM HMM is more suitable for this research direction in comparison with other techniques that deal with micro-blog level such as self-learning and co-training.

The proposed solution is inspired by Qi et al. [149]. However, we developed a new method for generating feature attributes. Qi et al. use supervised classifier for generating the new feature attributes. Additionally, their solution counts number of sequences that have been classified. Moreover, their proposed equation that is used for generating a new feature attribute calculates the total number of sequences that includes selected word in all unlabeled dataset.

Qi et al. proposed also an extension by clustering detected labels that are resulted from the classifier. Then, they use the cluster ids as additional feature attribute. Thereby, they need to find the optimum number of clusters by using optimization method. As a result of this, their method consumes more time when generating each feature attribute. While, our proposed method is simpler and decreases time consuming when generating feature attributes. Figure 10.6 describes the main idea of the proposed technique while all details are included in Figure 10.7.

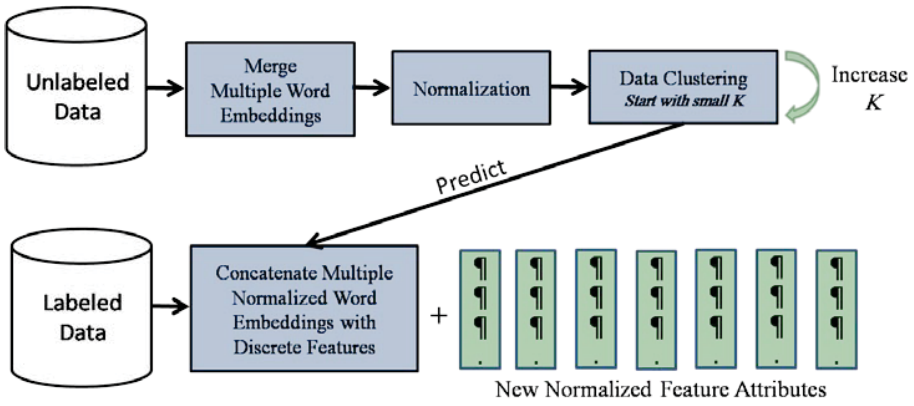

Unlabeled Data
Merge Multiple Word Embeddings
Normalization
Data Clustering Start with small K
Increase K
Predict
Labeled Data
Concatenate Multiple Normalized Word Embeddings with Discrete Features
+
New Normalized Feature Attributes

Figure 10.6: Semi-supervised learning technique for open domain targeted sentiment classification.

```
Algorithm of new semi-supervised learning technique for open domain targeted sentiment classification:
Inputs: Label ratio, training set (trainSet), Development set (DevSet), testing set (TestSet)
Output: precision, recall, and F1-score of classifying testing data
1)  Split trainSet into labeled data (trainSetLab) equals ratio value and the rest as unlabeled data
    (trainSetUnLab)
2)  Build SVM HMM model and train it by using trainSetLab data with an initial small value of C
    parameter
3)  Calculate zero/one-error of classifying DevSet
4)  Increase value of C parameter and repeat steps 2 and 3 until zero/one-error does not decrease.
5)  Check performance of SVM HMM model by using optimum value of C parameter.
6)  Find only numerical values in each vector of trainSetUnLab data and store them in trainUnLabArray
7)  Cluster the trainUnLabArray by using K-means with initial value of number of clusters (ClusterNum).
8)  For each word in trainSetLab determine cluster ID (ClusterID) which the word belongs to.
9)  Normalize values of all ClusterID to form ClusterIDNorm for each word in trainSetLab data.
10) Concatenate ClusterIDNorm as new feature attribute to the feature vector of each word in trainSetLab
    to form trainSetLab+.
11) Retrain the SVM HMM model by using trainSetLab+.
12) Increase value of ClusterNum and iterate steps 5 to 10 until stopping criterion is met.
13) Classify TestSet data by using the best SVM HMM model and output results.
```

Figure 10.7: Algorithm of semi-supervised learning technique for open domain targeted sentiment.

The optimization process, which is conducted by using steps 2 and 3, is the same optimization method which is illustrated in Figure 10.4 for finding optimum value of *C* parameter. The normalization process in step 9 is calculated by dividing each cluster id (*ClusterID)* by the total number of clusters (*ClusterNum*). Thus, the values of normalized cluster ids (*ClusterIDNorm*) are fallen in the range (0, 1]. The stopping criterion in step 12 can be conducted by using different ways. In this work, we applied a stopping criterion that checks whether the performance of learned SVM HMM (step 5) is improved significantly after each incremental increase in *ClusterNum* value.

10.2.5 Supervised Learning with Dynamic Generation of Feature Attributes

This solution is based on the proposed technique that is described in Figure 10.7. While, we use here all training set (*trainSet*) as labeled data instead of splitting it into labeled and unlabelled data when training the SVM HMM model. We propose this solution for evaluating the performance of employing supervised learning method with generating feature attributes dynamically.

To save memory and make this technique faster, we selected by default half amount (almost 50%) of training set for conducting clustering process when generating feature attributes. However, using larger amount of training set will improve the performance of data clustering since more words will be clustered correctly. The outcome is selected based on finding the best performance achieved when applying incremental generation of feature attributes. Thereby, if the generated feature attributes do not improve the performance, then the technique will select the outcome immediately from the basic supervised learning classifier.

CHAPTER 11

EXPERIMENT RESULTS:

OPEN DOMAIN TARGETED SENTIMENT ANALYSIS

This chapter presents experiment work that is developed to show efficacy of our solutions for improving the performance of open domain targeted sentiment classification. Various discussions are included also for analyzing numerous experiment results.

11.1 Using Cluster IDs as Feature Attribute

This section describes our work for improving the performance of open domain targeted sentiment models that are proposed by Mitchell et al. [124]. These models have been introduced as the first approach for open domain sentiment classification. We improved the performance by adding another feature attribute to the dataset. Since we could not use numerical feature attributes with CRF, a data clustering method is applied to the dataset and the cluster ids (integer values) are used as additional feature attribute.

To achieve our goal, we firstly collected word2vec embeddings that are representing each word in the used dataset by using pre-trained word2vec embeddings provided by Zhang et al. [125]. Then, we clustered the data of word2vec embeddings that represent all entities in tweets. Finally, cluster id is concatenated with the discrete feature attributes that are used by Mitchell et al. [124]. Thereby, the added feature attribute represents to which cluster the corresponding word is belong.

We applied this technique to the 2nd fold of dataset which is available in implementation code provided by Mitchell et al. [124]. We used K-means method for clustering all word2vec embeddings. Number of these word2vec embeddings (includes both training and testing data) is 35681 vectors. After adding cluster IDs as feature attribute to the used dataset, we checked the improving in performance by training and testing all models proposed by Mitchell et al. [124]. We conducted this experiment by modifying implementation code provided by Mitchell et al. [124] which is available publically[9]. Table 11.1 describes all tested models while Table 11.2 and Table 11.3 show results provided when conducting the experiments by using cluster granularity equals 0.1. Table 11.3 does not include base models since they provide same results in comparison with their corresponding models when applying NER. Moreover, the corresponding results of SA could not be calculated for base models.

Table 11.1: Description of all evaluated models.

Model	Description
Joint_CRF_Base	Baseline joint model which uses volitional entity labels that are specified by Mitchell et al. [124] and assign no sentiment directed towards the entity.
Joint_CRF	Joint model proposed by Mitchell et al. [124]
Joint_Clusters_Base	Adding clusters ids as feature attribute to Joint_CRF_Base model
Joint_Clusters	Adding clusters ids as feature attribute to Joint_CRF model.
Pipeline_CRF_Base	Baseline pipeline model which uses volitional entity labels that are specified by Mitchell et al. [124] and assign no sentiment directed towards the entity.
Pipeline_CRF	Pipeline model proposed by Mitchell et al. [124]
Pipeline_Clusters_Base	Adding clusters ids as feature attribute to Pipeline_CRF_Base
Pipeline_Clusters	Adding clusters ids as feature attribute to Pipeline_CRF
Collapsed_CRF_Base	Baseline collapsed model which uses volitional entity labels that are specified by Mitchell et al. [124] and assign no sentiment directed towards the entity.
Collapsed_CRF	Collapsed model proposed by Mitchell et al. [124]
Collapsed_Clusters_Base	Adding clusters ids as feature attribute to Collapsed_CRF_Base
Collapsed_Clusters	Adding clusters ids as feature attribute to Collapsed_CRF

[9] http://www.m-mitchell.com/code/index.html

Table 11.2: Results of evaluating models (part 1/2).

Model		Acc-all	Acc-Bsent
Joint	Joint_CRF_Base	87.25	32.69
	Joint_CRF	87.18	32.05
	Joint_Clusters_Base	**90.18**	**33.83**
	Joint_Clusters	89.89	31.84
Pipeline	Pipeline_CRF_Base	87.73	32.01
	Pipeline_CRF	87.73	32.01
	Pipeline_Clusters_Base	**90.3**	**37.38**
	Pipeline_Clusters	90.06	35
Collapsed	Collapsed_CRF_Base	89.77	30
	Collapsed_CRF	89.77	30
	Collapsed_Clusters_Base	**90.44**	**32.41**
	Collapsed_Clusters	90.44	31.66

Table 11.3: Results of evaluating models (part 2/2).

Model	NER			SA		
	Precision	Recall	Specificity	Precision	Recall	Specificity
Joint_CRF	52.11	70.76	91.78	41	46.59	75
Joint_Clusters	69.17	54.94	96.8	35.48	37.5	74.58
Pipeline_CRF	53.08	65.82	92.66	46.91	43.18	81.78
Pipeline_Clusters	67.76	63.58	96.05	43.43	48.86	76.27
Collapsed_CRF	64.8	52.81	96.29	51.43	9	86.29
Collapsed_Clusters	71.82	54.41	97.2	46.77	15.18	75.19

11.2 Supervised Learning of Combined Discrete Features and Multiple Word Embeddings

In this proposed solution, we use sequence tagging with structural support vector which is referred to as SVM HMM[10]. To make our comparison with previous related works more accurate and fair enough, we used the same code that is provided by Li and Lu [130] for calculating evaluation measures. To be able to use the same public DatasetB utilized by pervious related works, we reformatted the feature vectors to fit our proposed techniques. We converted the data form which is provided by Zhang et al. [125] to fit format of sequence tagging with structural support vector machines[11] (SVM HMM). We prepared

[10] https://www.cs.cornell.edu/people/tj/svm_light/svm_hmm.html

[11] https://www.cs.cornell.edu/people/tj/svm_light/svm_hmm.html

the data to represent collapsed labels (B-negative, B-neutral, B-positive, I-negative, I-neutral, I-positive, and O).

As a result of this work, we prepared numerous datasets as described briefly in Table 11.4. We used only discrete features for checking performance of using these features alone. We refer to this resulted dataset as “Discrete_Data”. We also prepared data that includes only features attributes of pre-trained word2vec embeddings provided by Zhang et al. [125]. We refer to this data as “Word2VecZhang” which include feature vector of size 100 attributes. We normalized the “Word2VecZhang” and called it “Word2VecZhangNorm”. We prepared as well a dataset that combines both discrete and normalized word2vec embeddings to check its efficiency in improving the performance. We refer to this merged dataset as “Discrete_Word2VecZhangNorm”. Figure 11.1 shows a data point of this resulted dataset.

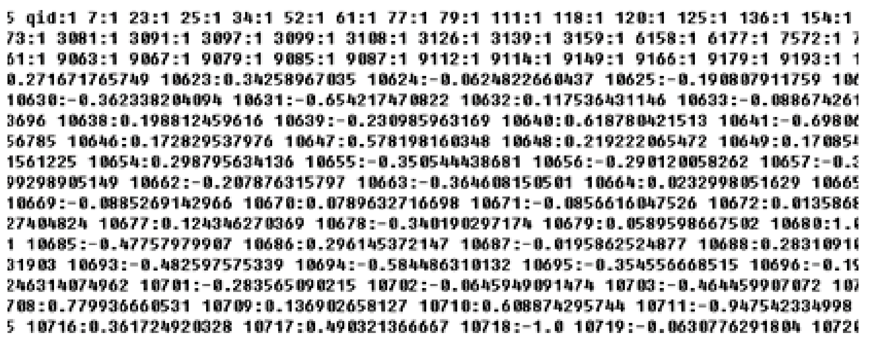

```
5 qid:1 7:1 23:1 25:1 34:1 52:1 61:1 77:1 79:1 111:1 118:1 120:1 125:1 136:1 154:1
73:1 3081:1 3091:1 3097:1 3099:1 3108:1 3126:1 3139:1 3159:1 6158:1 6177:1 7572:1 7
61:1 9063:1 9067:1 9079:1 9085:1 9087:1 9112:1 9114:1 9149:1 9166:1 9179:1 9193:1 1
0.271671765749 10623:0.34258967035 10624:-0.0624822660437 10625:-0.190807911759 106
10630:-0.362338204094 10631:-0.654217470822 10632:0.117536431146 10633:-0.088674261
3696 10638:0.198812459616 10639:-0.230985963169 10640:0.618780421513 10641:-0.6980
56785 10646:0.172829537976 10647:0.578198160348 10648:0.219222065472 10649:0.17085
1561225 10654:0.298795634136 10655:-0.350544438681 10656:-0.290120058262 10657:-0.3
99298905149 10662:-0.207876315797 10663:-0.364608150501 10664:0.0232998051629 10665
10669:-0.0885269142966 10670:0.0789632716698 10671:-0.0856616047526 10672:0.013586
27404824 10677:0.124346270369 10678:-0.340190297174 10679:0.0589598667502 10680:1.0
1 10685:-0.47757979907 10686:0.296145372147 10687:-0.0195862524877 10688:0.283109
31903 10693:-0.482597575339 10694:-0.584486310132 10695:-0.354556668515 10696:-0.1
246314074962 10701:-0.283565090215 10702:-0.0645949091474 10703:-0.464459907072 107
708:0.779936660531 10709:0.136902658127 10710:0.608874295744 10711:-0.947542334998
5 10716:0.361724920328 10717:0.490321366667 10718:-1.0 10719:-0.0630776291804 1072
```

Figure 11.1: Formatting discrete feature attributes and word2vec embeddings for applying SVM HMM.

Additionally, we prepared data that includes pre-trained wor2vec embeddings provided by Al-Rfou et al. [150] and used by Li and Lu [130]. These wor2vec embeddings are available online and can be downloaded freely. Each vector of these word2vec embeddings contains 64 values. The resulted dataset is called “Word2VecPolyglot” and its normalized version is called “Word2VecPolyglotNorm”. We merged

"Word2VecPolyglotNorm" and "Word2VecZhangNorm" to build data that includes both representations of word2vec embeddings. The combined version is called "Word2VecBothPolyglot&ZhangNorm" and the dataset which includes additionally discrete features is called "DiscW2VPolyglot&ZhangNorm".

Table 11.4: Summary of all sets prepared from DatasetB for open domain targeted sentiment.

Dataset	Description
Discrete_Data	Includes only discrete features that are used by Mitchell et al. [124]
Word2VecZhang	Includes only word2vec embeddings features that are used by Zhang et al. [125]
Word2VecZhangNorm	Normalized version of "Word2VecZhang" dataset
Discrete_Word2VecZhangNorm	Combines both "Discrete_Data" and "Word2VecZhangNorm" datasets
Word2VecPolyglot	Includes wor2vec embeddings which are used by Li and Lu [130]
Word2VecPolyglotNorm	Normalized version of "Word2VecPolyglot" dataset
Word2VecBothPolyglot&ZhangNorm	Combines both "Word2VecPolyglotNorm" and "Word2VecZhangNorm" datasets
DiscW2VPolyglot&ZhangNorm	Combines both "Discrete_Data" and "Word2VecBothPolyglot&ZhangNorm" datasets
Word2VecBojanowski	Includes wor2vec embeddings used by Bojanowski et al. [151]
Word2VecBojanowskiNorm	Normalized version of "Word2VecBojanowski" dataset
Discrete_Word2VecBojanowskiNorm	Combines both "Discrete_Data" and "Word2VecBojanowskiNorm"
W2VPolyglotZhangBojanowskiNorm	Combines "Word2VecZhangNorm", "Word2VecPolyglotNorm", and "Word2VecBojanowskiNorm"
DW2VPolyglotZhangBojanowskiNor	Combines both "Discrete_Data" and "W2VPolyglotZhangBojanowskiNorm"

Moreover, we prepared another form of data that includes a third source of pre-trained word2vec embeddings called fastText [151]. This representation of word2vec embeddings has dimension size equals 300 attributes and it is also available online[12]. The resulted dataset is called "Word2VecBojanowski" and the normalized version is called "Word2VecBojanowskiNorm". We merged also these normalized word2vec embeddings with the discrete features and called it as "Discrete_Word2VecBojanowskiNorm". We merged the three forms of word2vec embeddings in one dataset called

[12] https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

“W2VPolyglotZhangBojanowskiNorm”. When combining the discrete features to “W2VPolyglotZhangBojanowskiNorm”, the resulted dataset is called “DW2VPolyglotZhangBojanowskiNor”.

We apply an optimization task for selecting best value of *C* parameter when using SVM HMM. It is important to clarify that we did not optimize Epsilon parameter since using its default value is enough to converge to high performance when changing values of *C* parameter. We selected the best value of *C* parameter that provides lowest "zero/one-error" when classifying development set. The evaluation measure "zero/one-error" is one of results that are provided by the used tool when building SVM HMM model.

We trained the SVM HMM model by using different values of *C* parameter in the range between 1 and 550 with an increasing step that is equal to 10. With each selected *C* value we learned the SVM HMM model by using training data and calculated "zero/one-error" by classifying development data. We use the best *C* value for classifying the testing data and calculating evaluation measures (precision, recall, and F1-score). It is noteworthy that using testing data instated of development data will provide better values of *C* parameter. However, we use development set rather than testing data for providing real results. We applied SVM HMM to the 2nd fold of all prepared set collected from English tweets. We reported all results when using each dataset described above as shown in Table 11.5. The maximum values in this table are highlighted as bold font. The experiment results show that there are 324 data points that match criteria of open domain targeted sentiment classification. These data points specify number of words that are targeted as topics and have sentiment polarities.

Table 11.5: Summary of best result when applying SVM HMM to the 2nd fold of prepared datasets.

Dataset	Err	*C*	Pred #	NER			SA		
				P	R	F1	P	R	F1
Discrete_Data	80.66	111	161	69.57	34.57	46.19	55.9	27.78	37.11
Word2VecZhang	91.51	101	99	57.58	17.59	26.95	43.43	13.27	20.33
Word2VecZhangNorm	91.98	101	118	50.85	18.52	27.15	37.29	13.58	19.91
Discrete_Word2VecZhangNorm	75.47	81	231	64.5	45.99	53.69	48.48	34.57	40.36
Word2VecPolyglot	82.55	41	165	67.88	34.57	45.81	51.52	26.23	34.76
Word2VecPolyglotNorm	82.55	41	178	65.73	36.11	46.61	50.56	27.78	35.86
Word2VecBothPolyglot&ZhangNorm	79.25	41	192	66.15	39.2	49.22	51.56	30.56	38.37
DiscW2VPolyglot&ZhangNorm	73.11	31	226	71.68	50	58.91	54.87	38.27	45.09
Word2VecBojanowski	75	71	220	65.91	44.75	53.31	49.09	33.33	39.71
Word2VecBojanowskiNorm	75	81	220	68.64	46.6	55.51	51.82	35.19	41.91
Discrete_Word2VecBojanowskiNorm	74.06	41	237	73.84	54.01	62.39	54.85	40.12	46.35
W2VPolyglotZhangBojanowskiNorm	73.58	31	220	69.55	47.22	56.25	51.36	34.88	41.54
DW2VPolyglotZhangBojanowskiNor	**70.75**	**21**	**242**	**74.38**	**55.56**	**63.6**	**56.61**	**42.28**	**48.41**

Since DW2VPolyglotZhangBojanowskiNor dataset provides the best results (lowest error) as shown in Table 11.5, we applied SVM HMM to all folds of this dataset. All results that are generated when using both English and Spanish data are reported in Table 11.6. This experiment uses optimization method to find best value of *C* parameter which provides the lowest value of "zero/one-error" (Err). We changed value of *C* parameter from 1 into 550 with increase step equals 10. The table includes also number of observed data points (obs) and number of data points (Pred) that are detected correctly. The results include three evaluations measures: precision (P), recall (R), and F1-score (F1) when applying both name entity recognition (NER) and sentiment analysis (SA). The maximum values of classification accuracy and F1-score among all folds are highlighted by using bold and underlined font.

Table 11.6: Results of applying SVM HMM to prepared dataset included discrete and three sources of word2vec embeddings.

Lang	**Fold**	**Err**	***C***	**Obs #**	**Pred #**	**NER**			**SA**		
						P	**R**	**F1**	**P**	**R**	**F1**
	1	69.34	101	347	311	69.45	62.25	**65.65**	49.52	44.38	46.81
	2	70.75	21	324	242	74.38	55.56	63.6	56.61	42.28	48.41
	3	68.87	51	346	274	67.15	53.18	59.35	48.18	38.15	42.58
	4	73.11	51	318	253	67.59	53.77	59.89	49.41	39.31	43.78
	5	69.34	61	340	259	67.18	51.18	58.1	48.65	37.06	42.07
Eng	6	68.87	31	319	243	72.43	55.17	62.63	51.85	39.5	44.84
	7	67.92	31	309	218	70.64	49.84	58.44	50.0	35.28	41.37
	8	69.34	21	320	233	74.68	54.37	62.93	60.09	43.75	**50.63**
	9	69.34	61	346	295	69.15	58.96	63.65	45.76	39.02	42.12
	10	69.81	31	319	232	68.1	49.53	57.35	48.71	35.42	41.02
	Avg	**69.67**	**46**	**329**	**256**	**70.08**	**54.38**	**61.16**	**50.88**	**39.42**	**44.36**
	1	64.87	81	677	556	77.16	63.37	69.59	50.54	41.51	45.58
	2	64.36	121	656	563	74.96	64.33	69.24	46.36	39.79	42.82
	3	62.42	151	676	524	75.38	58.43	65.83	50.19	38.91	43.83
	4	65.52	121	641	538	79.0	66.3	72.09	52.23	43.84	47.67
	5	64.58	111	669	545	81.28	66.22	**72.98**	51.56	42.0	46.29
Span	6	64.66	121	663	556	74.1	62.14	67.6	48.38	40.57	44.13
	7	65.44	141	651	533	76.17	62.37	68.58	47.28	38.71	42.57
	8	65.3	111	681	592	73.82	64.17	68.66	46.62	40.53	43.36
	9	62.2	141	661	581	71.77	63.09	67.15	44.75	39.33	41.87
	10	66.81	51	675	545	78.17	63.11	69.84	53.58	43.26	**47.87**
	Avg	**64.62**	**115**	**665**	**553**	**76.18**	**63.35**	**69.16**	**49.15**	**40.85**	**44.60**

11.3 Semi Supervised Learning

This section shows experiment results that are provided by applying different semi-supervised learning techniques for open domain targeted sentiment classification. The next subsection shows efficacy of applying label propagation. The other following subsection shows experiment results that are provided when applying our proposed solution that uses partially labeled data.

11.3.1 Label Propagation

We developed an experiment to evaluate efficacy of applying label propagation method. We used only feature attributes of word2vec embeddings for training and testing label propagation method since this method uses only numerical data for finding nearest neighbors. Thereby, the used data include only numerical vectors that represent each word in the dataset.

We used "W2VPolyglotZhangBojanowskiNorm" dataset for conducting this experiment since it includes all pre-trained word2vec embeddings that are collected from the three resources. We selected different values for setting kNN (k nearest neighbor) parameter. We changed as well the ratio of used labeled data that is collected from the training set. All results are reported in Table 11.7. We noticed clearly that this method is not suitable for solving our research problem because it does not consider the relation between words (tokens) in the same tweet.

Table 11.7: Summary of best result provided when applying label propagation.

Ratio %	**kNN**	**Pred #**	**NER**			**SA**		
			P	**R**	**F1**	**P**	**R**	**F1**
11	3	4205	4.68	60.8	8.7	0.48	6.17	0.88
51	3	4205	4.68	60.8	8.7	0.48	6.17	0.88
31	81	47	72.34	10.49	18.33	57.45	8.33	14.56
51	81	47	72.34	10.49	18.33	57.45	8.33	14.56
71	81	47	72.34	10.49	18.33	57.45	8.33	14.56
31	150	39	87.18	10.49	18.73	69.23	8.33	14.88
51	150	39	87.18	10.49	18.73	69.23	8.33	14.88
31	200	39	**87.18**	**10.49**	**18.73**	**69.23**	**8.33**	**14.88**
31,51	250,300	0	0	0	0	0	0	0

11.3.2 Semi-supervised Learning with Dynamic Generation of Feature Attributes

We developed an experiment to show performance of our proposed semi-supervised learning solution. To make our comparison with previous related works more accurate and fair enough, we used the same code that is provided by Li and Lu [130] for calculating the evaluation measures. We use all results reported by Li and Lu to make our comparisons and show efficiency of our proposed solution.

We changed ratio of labeled data into 25%, 50%, and 75% of training data. At each selected ratio of labeled data, we applied both supervised SVM HMM and our proposed semi-supervised learning technique. We reported the experiment results to make the comparison easier and show the improvement in the performance at each ratio of labeled data. We apply as well a simple optimization method when using each ratio of labeled data for finding the best value of *C* parameter that provides the lowest value of "zero/one-error" (Err). The optimization method includes changing value of *C* parameter from 1 into 550 with increase step equals 10. We applied the proposed technique to all folds of DW2VPolyglotZhangBojanowskiNor dataset.

All results provided by using both English and Spanish are reported in Table 11.8. The table includes also number of observed data points (obs) and number of data points (Pred) that are detected correctly. The table shows the average values provided when using the 10 folds. The maximum values in this table are highlighted by using bold font. For more details, refer to tables from Table IV.1 to Table IV.6 in Appendix IV to see all numerical values that are resulted from these experiments.

Table 11.8: Average performance of applying semi-supervised learning with dynamic generation of feature attributes.

Lang	**Model**	**Ratio**	**NER**			**SA**		
			P	**R**	**F1**	**P**	**R**	**F1**
Eng	Supervised	25	64.65	48.57	55.08	45.76	34.19	38.85
	Semi-supervised		64.20	50.18	55.84	45.92	35.77	39.86
	Supervised	50	66.46	51.92	58.21	47.86	37.31	41.88
	Semi-supervised		66.43	53.46	59.13	48.34	38.81	42.97
	Supervised	75	68.93	51.86	59.15	50.56	38.00	43.36
	Semi-supervised		**68.21**	**53.10**	**59.65**	**50.86**	**39.57**	**44.46**
Span	Supervised	25	68.86	61.12	64.72	40.04	35.53	37.63
	Semi-supervised		67.48	62.97	65.05	39.90	37.21	38.45
	Supervised	50	73.73	61.18	66.84	45.42	37.66	41.15
	Semi-supervised		71.48	64.02	67.39	44.15	39.51	41.61
	Supervised	75	74.66	62.34	67.93	47.06	39.27	42.80
	Semi-supervised		**74.31**	**63.32**	**68.27**	**46.93**	**39.98**	**43.12**

11.4 Supervised Learning with Dynamic Generation of Feature Attributes

We developed experiments to evaluate efficacy of merging supervised SVM HMM with our proposed method of generating feature attributes dynamically. We applied this combined supervised learning technique to all folds of DW2VPolyglotZhangBojanowskiNor dataset. With each fold we run optimization method for finding the optimum value of *C* parameter by finding the lowest value of "zero/one-error" (Err). We changed value of *C* parameter from 1 into 550 with increase step equals 10. When clustering data which is used for generating feature attributes dynamically, we used a ratio of labeled data that is equal to 51% of training set.

All results achieved by applying these experiments to both English and Spanish data are reported in Table 11.9. The maximum values of accuracy and F1-score that are generated when evaluating sentiment analysis are highlighted by using bold and underlined font. While, the average values of all results provided when using all folds are highlighted by using only bold font.

11.4.1 Using Additional Dataset

To show performance of using our proposed technique in other environments, we applied it to the dataset (DatasetC) which is collected by Zhang et al. [128]. We used evaluation measure acc-all for reporting results of this experiment. Our proposed supervised learning technique provides acc-all equals 91.63% while the lowest zero/one-error achieved by SVM HMM is 81.18% when *C*=111. For making the results more accurate, we used all training set during process of date clustering to involve all words.

Table 11.9: Results of applying supervised learning with dynamic generation of feature attributes.

Lang	**Fold**	**Err**	***C***	**Obs #**	**Pred #**	**NER**			**SA**		
						P	**R**	**F1**	**P**	**R**	**F1**
	1	69.34	101	347	316	68.67	62.54	**65.46**	49.37	44.96	47.06
	2	70.75	21	324	254	72.83	57.1	64.01	55.91	43.83	49.13
	3	68.87	51	346	254	69.29	50.87	58.67	50.0	36.71	42.33
	4	73.11	51	318	268	65.67	55.35	60.07	48.51	40.88	44.37
	5	69.34	61	340	260	66.54	50.88	57.67	50.0	38.24	43.33
Eng	6	68.87	31	319	271	67.9	57.68	62.37	49.82	42.32	45.76
	7	67.92	31	309	219	71.23	50.49	59.09	50.68	35.92	42.05
	8	69.34	21	320	229	76.42	54.69	63.75	60.26	43.13	**50.27**
	9	69.34	61	346	288	70.83	58.96	64.35	47.57	39.6	43.22
	10	69.81	31	319	225	71.11	50.16	58.82	52.89	37.3	43.75
	Avg	**69.67**	**46**	**329**	**258**	**70.05**	**54.87**	**61.43**	**51.50**	**40.29**	**45.13**
	1	64.87	81	677	576	76.04	64.7	69.91	50.17	42.69	46.13
	2	64.36	121	656	564	76.06	65.4	70.33	47.52	40.85	43.93
	3	62.42	151	676	571	74.61	63.02	68.32	48.34	40.83	44.27
	4	65.52	121	641	538	79.0	66.3	72.09	52.23	43.84	47.67
	5	64.58	111	669	604	79.3	71.6	**75.26**	51.49	46.49	**48.86**
Span	6	64.66	121	663	556	74.1	62.14	67.6	48.38	40.57	44.13
	7	65.44	141	651	533	76.17	62.37	68.58	47.28	38.71	42.57
	8	65.3	111	681	658	70.36	67.99	69.16	45.44	43.91	44.66
	9	62.2	141	661	665	66.62	67.02	66.82	42.11	42.36	42.23
	10	66.81	51	675	594	76.77	67.56	71.87	51.85	45.63	48.54
	Avg	**64.62**	**115**	**665**	**586**	**74.90**	**65.81**	**69.99**	**48.48**	**42.59**	**45.30**

11.5 Context-Based Targeted Sentiment Analysis System

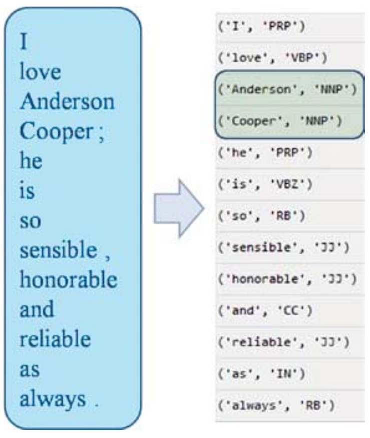
This section shows the performance of our proposed context-based analysis system. We applied the proposed system to two datasets. One of these datasets includes English tweets while the other one includes Arabic tweets. Next subsections present some case studies for showing the performance of the proposed system.

11.5.1 Using English Micro-blogs

We selected the training set included in DatasetA (6248 tweets) for showing efficiency of the proposed system when using English language. We used this dataset to validate the performance of our proposed system since this dataset is fully labeled. We also used

NLTK package[13] for applying POS English tagger. Figure 11.2 shows an example of detecting targets in the tweet by using POS tagger. We noticed that the targets are labeled as proper nouns when applying the POS tagger. Figure 11.3 shows frequency of targets in the selected set of English tweets. This figure illustrates that our system detect targets that consists of more than one entity. For simplicity, we use symbol "-" to preview entities of same target in one line.

To validate the results, we compared number of detected targets with number of actual labeled targets. we noticed that frequency of detected targets is close to actual labeled targets. For example, frequency of detected target "Barack Obama" is 223 while the actual frequency is 222. Another example shows that frequency of detected target "Jimmy Carter" is 98 while the actual number is 101. The difference in comparisons comes from nature of some tweets which include more than one target. Thereby, our system counts all targets in the same tweet. While, the dataset includes annotation for one target per each tweet.

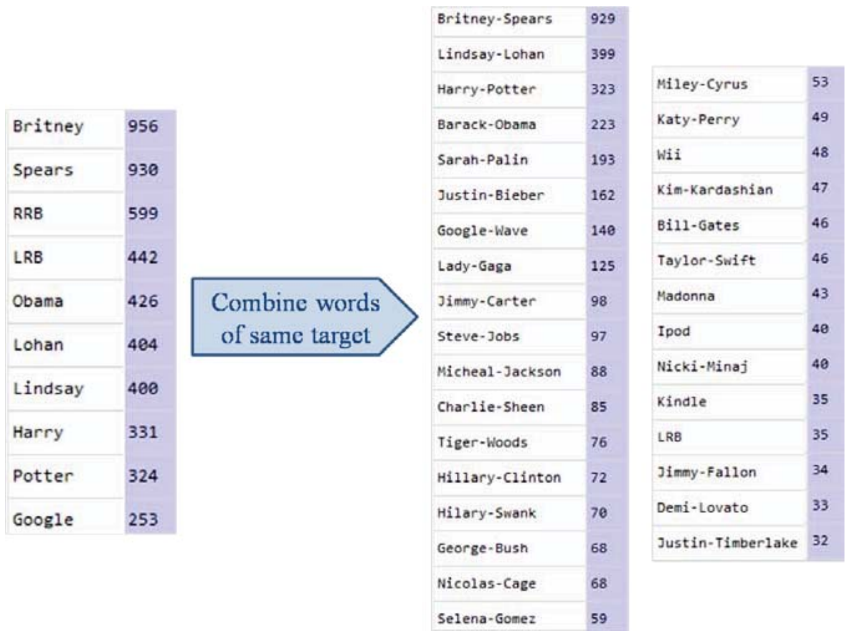


I love Anderson Cooper ; he is so sensible , honorable and reliable as always .

```
('I', 'PRP')
('love', 'VBP')
('Anderson', 'NNP')
('Cooper', 'NNP')
('he', 'PRP')
('is', 'VBZ')
('so', 'RB')
('sensible', 'JJ')
('honorable', 'JJ')
('and', 'CC')
('reliable', 'JJ')
('as', 'IN')
('always', 'RB')
```

Figure 11.2: Detecting target in the tweet by using POS tagger.

[13] https://www.nltk.org/


Combine words of same target

Britney	956
Spears	930
RRB	599
LRB	442
Obama	426
Lohan	404
Lindsay	400
Harry	331
Potter	324
Google	253

Britney-Spears	929
Lindsay-Lohan	399
Harry-Potter	323
Barack-Obama	223
Sarah-Palin	193
Justin-Bieber	162
Google-Wave	140
Lady-Gaga	125
Jimmy-Carter	98
Steve-Jobs	97
Micheal-Jackson	88
Charlie-Sheen	85
Tiger-Woods	76
Hillary-Clinton	72
Hilary-Swank	70
George-Bush	68
Nicolas-Cage	68
Selena-Gomez	59

Miley-Cyrus	53
Katy-Perry	49
Wii	48
Kim-Kardashian	47
Bill-Gates	46
Taylor-Swift	46
Madonna	43
Ipod	40
Nicki-Minaj	40
Kindle	35
LRB	35
Jimmy-Fallon	34
Demi-Lovato	33
Justin-Timberlake	32

Figure 11.3: Frequency of targets in the set of English tweets.

Figure 11.4 shows sentiment analysis of two targets selected as a case study. We selected this case study because there are some shared topics that describe both targets. Additionally, frequency of these targets is large which helps in conducting experimental work. Our system is able to detect topics by selecting nouns from the tweet as shown in Figure 11.5. Frequencies of main topics described in the case study are shown in Figure 11.6. Figure 11.7 shows results of applying sentiment analysis to the main topics.

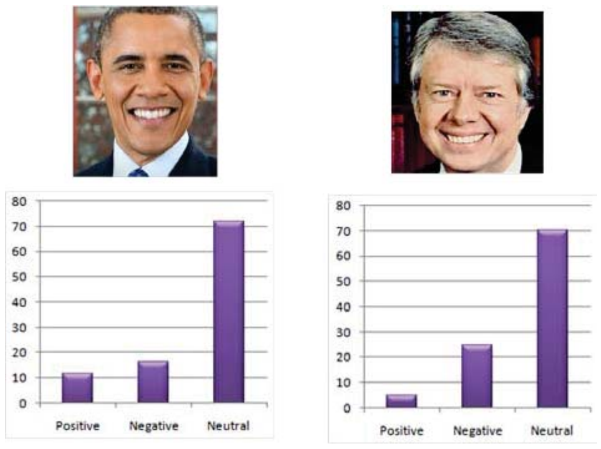
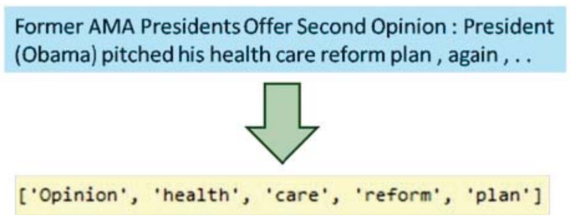


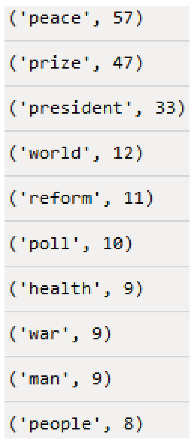
Figure 11.4: Sentiment analysis of case study that includes two targets.



Former AMA Presidents Offer Second Opinion : President (Obama) pitched his health care reform plan , again , . .

```
['Opinion', 'health', 'care', 'reform', 'plan']
```

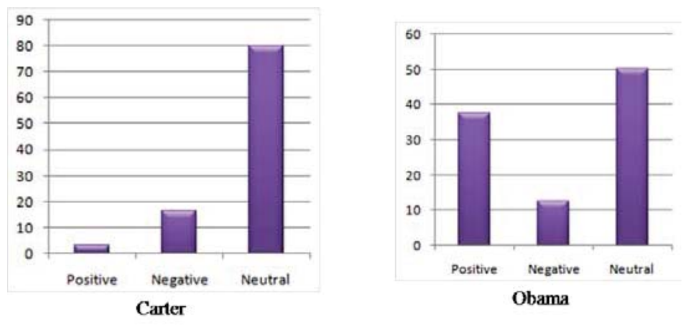
Figure 11.5: Detecting main topics in the micro-blog.



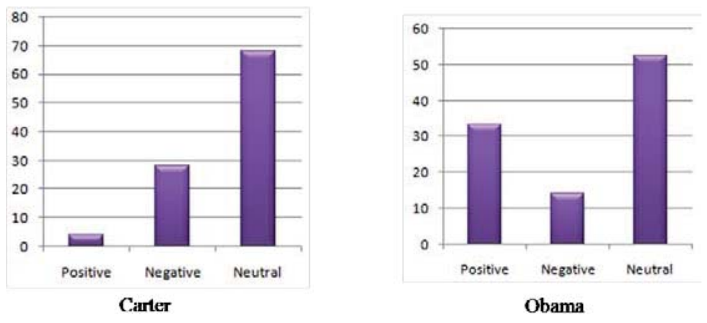
```
('peace', 57)
('prize', 47)
('president', 33)
('world', 12)
('reform', 11)
('poll', 10)
('health', 9)
('war', 9)
('man', 9)
('people', 8)
```

Figure 11.6: Frequency of main topics in the micro-blogs of case study.

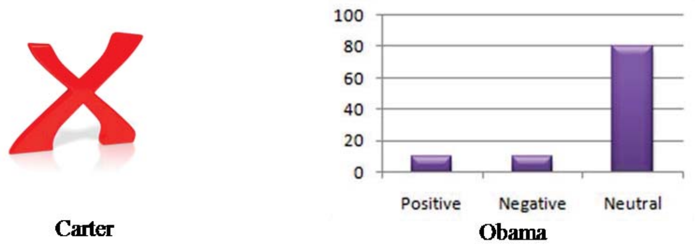
The proposed system is able also to detect subtopics as illustrated in Figure 11.8. Figure 11.9 illustrates results of applying hierarchal data clustering for detecting more subtopics. Figure 11.9 (b) shows how we used Elbow measure for detecting the optimum number of clusters to make a cut in the resulted dendrogram shown in Figure 11.9 (a). Figure 11.10 shows results of applying sentiment analysis to subtopics discussed in the case study. Figure 11.11 and Figure 11.12 show results of analysis two more targets described in the selected set of English tweets. The system did not provide sentiment polarities for the "Bill Gates" target because number of topics that describe this target are smaller the selected threshold.



a) Sentiment analysis of “Peace” Topic



b) Sentiment analysis of “Prize” Topic



c) Sentiment analysis of “Reform” Topic

Figure 11.7: Sentiment analysis of main topics in the micro-blogs of case study.

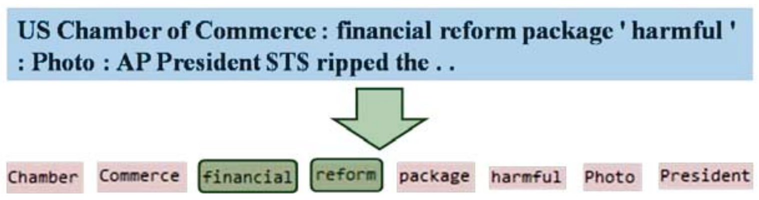
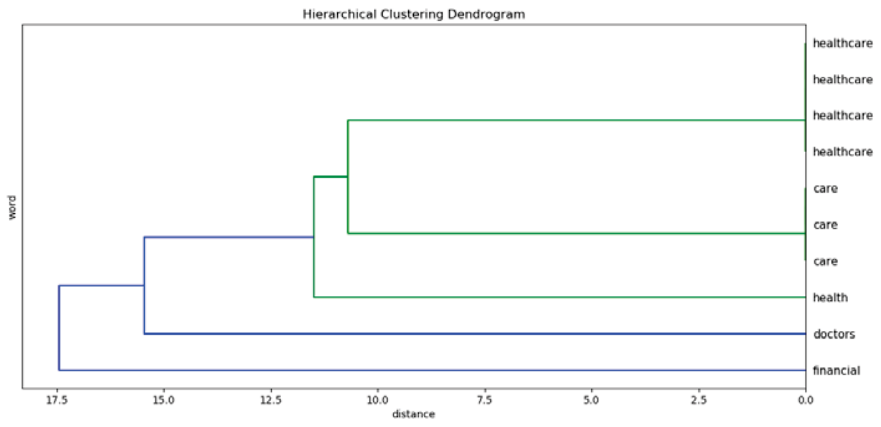
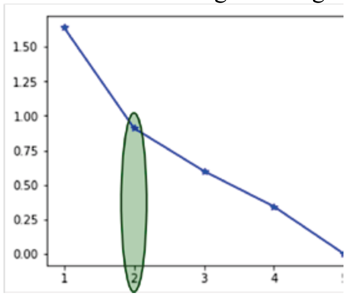

US Chamber of Commerce : financial reform package ' harmful ' : Photo : AP President STS ripped the . .
Chamber
Commerce
financial
reform
package
harmful
Photo
President

Figure 11.8: Detecting subtopics in the micro-blog.



a) Hierarchal Clustering Dendrogram.



b) Elbow measure.

Figure 11.9: Hierarchal Clustering for detecting subtopics in the micro-blogs of case study.

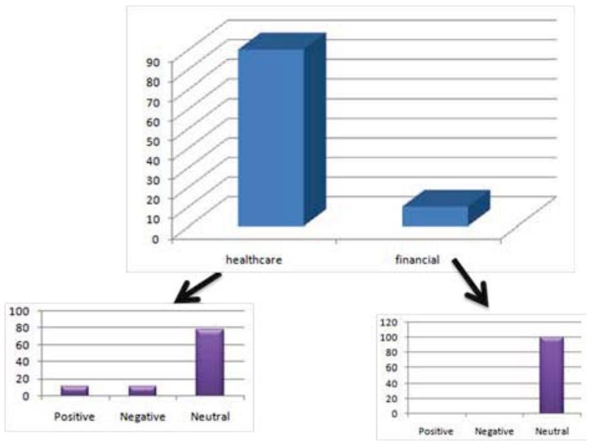
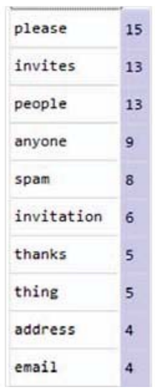


Figure 11.10: Sentiment analysis of subtopics in the micro-blogs of case study.



please	15
invites	13
people	13
anyone	9
spam	8
invitation	6
thanks	5
thing	5
address	4
email	4

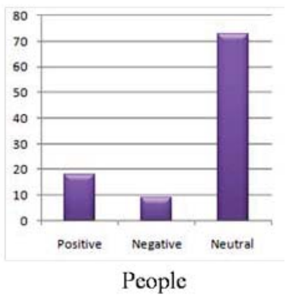
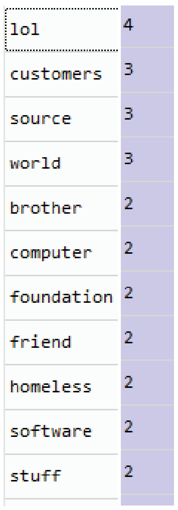


Figure 11.11: Results of analysis "Google Wave" target.



lol	4
customers	3
source	3
world	3
brother	2
computer	2
foundation	2
friend	2
homeless	2
software	2
stuff	2

Figure 11.12: Results of analysis "Bill Gates" target.

11.5.2 Using Arabic Micro-blogs

We also selected the training set (1999 tweets) included in DatasetD for showing efficiency of the proposed system when using Arabic language. Since tagging Arabic words is more difficult, we used a specific package[14] for applying POS Arabic tagger. Table 11.10 shows most common targets that are detected by using our proposed system. Since target "مصر" is the most common one, we selected it as a case study.

[14] https://github.com/EmilStenstrom/rippletagger/blob/master/README.md

Table 11.11 shows the main topics that are discussed in micro-blogs included target "مصر". In preprocessing phase, we normalized words that have different surface forms and replaced them by one form. For example, we replaced word "شعب" by word "الشعب". This normalization process helps in counting all relevant words.

Table 11.10: Most common detected targets in Arabic tweets.

Target	Frequency	Target	Frequency
مصر	106	رئيس	19
ايران	55	الرياض	17
حلب	55	نجران	16
الشعب	46	البلد	28
العراق	36	فلسطين	15
اليمن	35	الوطن	13
الاخوان	31	الامارات	13
مريم	31	العرب	13
زويل	24	المسلمين	12
العالم	24	السعوديه	12
الكويت	24	السلام	11
الهلال	22	الخليج	11
اسرائيل	19		

Table 11.11: Most common topics detected in the case study of Arabic tweets.

Topic	Frequency	Topic	Frequency
الشعب	8	حكم	2
الشباب	4	الفساد	2
فلسطين	3	الواقع	2
العالم	3	المال	2
الاخوان	2	مستقبل	2
الامارات	2	الاجانب	1
العسكر	2		

Table 11.12 shows results of detecting subtopics among tweets included the topic "الشعب" in this case study. Left column shows the detected subtopics while the right column illustrates whether the corresponding subtopic is detected correctly. It is clear that our proposed system could not detect correctly all subtopics. Section 11.6 discusses some explanations for this result. We did not show results of clustering subtopics since frequency of each subtopic is small. We also could not apply sentiment classification since the dataset does not include sentiment labels.

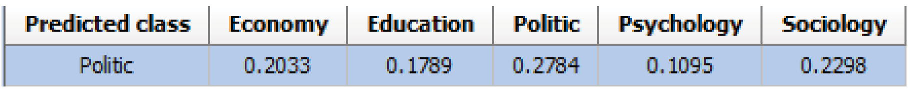
Table 11.12: Examples of detected subtopics described in a case study of Arabic tweets.

Detected Subtopic	Actual Subtopic
ليل	**الريس**
الاخوان	الاخوان
بلد	**الامارات**
مجرمين	مجرمين
فقر	فقر
الصمت	الصمت
حرام	**الاجانب**

11.5.3 Performance of Existing Text Analysis System

We checked efficacy of existing document analysis system in detecting topics among a set micro-blogs. We used software called WordStat document classifier[15] v1.0 (Provalis Research) to conduct our experiment. This tool is used for applying text categorization and document classification. It uses Naive Bayes classifier for classifying the document. To conduct this experiment, we input a set of English tweets as one document to see how this tool detects the topics.

The results show that this system could not detect the targets described in the set of tweets. This tool expects that the whole document covers only one target. While, our proposed system detects all targets in the first level of topic categorization. Additionally, this tool classifies words to classes of specific domains as shown in Figure 11.13. While, our proposed system deals with an open domain aspect.

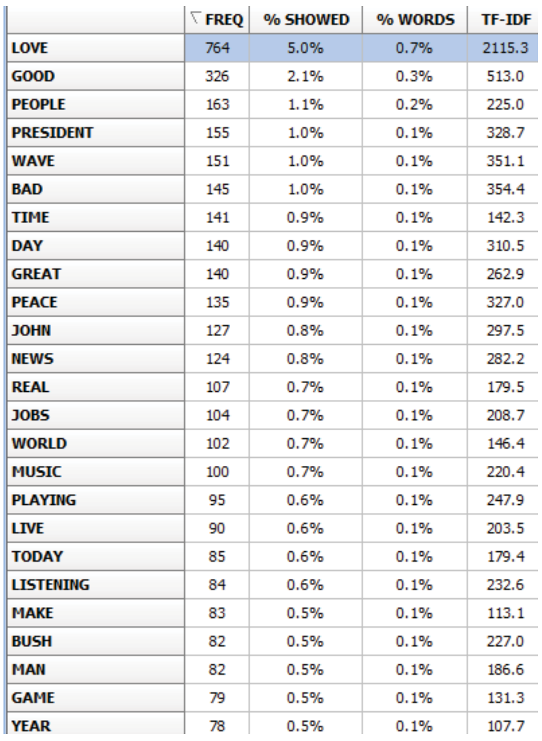


Predicted class	Economy	Education	Politic	Psychology	Sociology
Politic	0.2033	0.1789	0.2784	0.1095	0.2298

Figure 11.13: Results of applying WordStat document classifier for detecting topics.

[15] https://provalisresearch.com/downloads/trial-versions/

Figure 11.14 shows some of common words that are detected by using WordStat tool. It is clear that this tool skipped many name entities such as "Barack Obama" and calculated frequency for the rest of words. Thereby, this tool could not identify the targets. Additionally, it could not identify subtopics after detecting the main topics. Regarding sentiment analysis, this tool identifies the whole document as one sentiment. Thereby, it could not find sentiment polarities towards all described topics.



	FREQ	% SHOWED	% WORDS	TF-IDF
LOVE	764	5.0%	0.7%	2115.3
GOOD	326	2.1%	0.3%	513.0
PEOPLE	163	1.1%	0.2%	225.0
PRESIDENT	155	1.0%	0.1%	328.7
WAVE	151	1.0%	0.1%	351.1
BAD	145	1.0%	0.1%	354.4
TIME	141	0.9%	0.1%	142.3
DAY	140	0.9%	0.1%	310.5
GREAT	140	0.9%	0.1%	262.9
PEACE	135	0.9%	0.1%	327.0
JOHN	127	0.8%	0.1%	297.5
NEWS	124	0.8%	0.1%	282.2
REAL	107	0.7%	0.1%	179.5
JOBS	104	0.7%	0.1%	208.7
WORLD	102	0.7%	0.1%	146.4
MUSIC	100	0.7%	0.1%	220.4
PLAYING	95	0.6%	0.1%	247.9
LIVE	90	0.6%	0.1%	203.5
TODAY	85	0.6%	0.1%	179.4
LISTENING	84	0.6%	0.1%	232.6
MAKE	83	0.5%	0.1%	113.1
BUSH	82	0.5%	0.1%	227.0
MAN	82	0.5%	0.1%	186.6
GAME	79	0.5%	0.1%	131.3
YEAR	78	0.5%	0.1%	107.7

Figure 11.14: Results of applying WordStat document classifier for analysis a set of tweets.

11.6 Discussion

Based on the experiment results, we noticed that using cluster ids as additional feature attributes improves significantly the performance of open domain targeted sentiment classification. We noticed clearly that Collapsed_Clusters_Base model outperforms all other models with respect to Acc-all measure. While, Pipeline_Clusters_Base model outperforms all other models with respect to Acc-Bsent measure. This means that collapsed models are the best in general. If our interest focuses on accuracy of name entity recognition, then our choice should be pipeline models. We also noticed that results of Acc-Bsent is too low (does not exceed 40%) since it is difficult to classify correctly the beginning of targeted entities.

Based on results reported in Table 11.5, we noticed clearly that using three sources of word2vec embeddings decreases effect of unseen words. After using these three sources most of words have at least one word2vec representation. We noticed as well that Bojanowski word2vec embeddings outperforms the other two word2vec embeddings. While, concatenate all word2vec embeddings with discrete features provides the best results.

To summarize our work, we reported all results that are achieved by our proposed solution in comparison with previous related works. We reported the average of all values that are achieved by using all folds however using some specific folds provide better results. All main results that are achieved for open domain targeted sentiment classification are reported in Table 11.13. The maximum achieved results are highlighted by using bold font. We noticed clearly that SVM HMM provides competitive results. Applying SVM HMM by using discrete features with multiple word2vec embeddings outperforms all previous related works. We noticed as well that using some specific folds provide better results as shown in Table 11.6.

Table 11.13: Main results of open domain targeted sentiment classification.

Model, year	English						Spanish					
	Entity Recognition			Sentiment Analysis			Entity Recognition			Sentiment Analysis		
	P.	**R.**	**F1**	**P.**	**R.**	**F1**	**P.**	**R.**	**F1**	**P.**	**R.**	**F1**
CRF-P [124],13	65.74	47.59	55.18	46.8	33.87	39.27	71.29	58.26	64.11	43.8	35.8	39.4
CRF-C[124],13	54.0	42.69	47.66	38.4	30.38	33.9	62.2	52.08	56.66	39.39	32.96	35.87
CRF-J [124],13	59.45	43.78	50.32	41.77	30.8	35.38	66.05	52.55	58.51	41.54	33.05	36.79
NN-P [125],15	60.69	51.63	55.67	43.71	37.12	40.06	70.77	62.0	65.76	46.55	40.57	43.04
NN-C [125],15	64.16	44.98	52.58	48.35	32.84	38.36	73.51	53.3	61.71	49.85	34.53	40.0
NN-J [125],15	61.47	49.28	54.59	44.62	35.84	39.67	71.32	61.11	65.74	46.67	39.99	43.02
SS [130], 17	63.18	51.67	56.83	44.57	36.48	40.11	71.49	61.92	66.36	46.06	39.89	42.75
SS(+w)[130],17	66.35	56.59	61.08	47.3	40.36	43.55	73.13	64.34	68.45	47.14	41.48	44.13
SS(+P)[130],17	65.14	55.32	59.83	45.96	39.04	42.21	71.55	62.72	66.84	45.92	40.25	42.89
SS(se)[130],17	63.93	54.53	58.85	44.49	37.93	40.94	70.17	64.15	67.02	44.12	40.34	42.14
SVMHMM+FE	70.08	54.38	61.16	50.88	39.42	44.36	76.18	63.35	69.16	49.15	40.85	44.60
Semi-Su-DFG	68.21	53.10	59.65	50.86	39.57	44.46	74.31	63.32	68.27	46.93	39.98	43.12
Super-DFG	**70.05**	**54.87**	**61.43**	**51.50**	**40.29**	**45.13**	**74.90**	**65.81**	**69.99**	**48.48**	**42.59**	**45.30**

Using our proposed semi-supervised learning solution (Semi-Su-DFG) provides competitive results with less number of labeled data. The performance of this solution is close to performance of prominent related work. Thus, it is a good choice for using our proposed solution when there is a lack of labeled data. Moreover, our proposed supervised learning solution (Super-DFG) with dynamic generation of feature attributes outperforms all models that are proposed so far. To the best of our knowledge, these maximum results are not reported before with any related work for open domain targeted sentiment classification. To show clearly the performance of our proposed solution Super-DFG, a summary of the provided F1-scores along with confidence interval of 95% confidence are shown in Table 11.14.

Table 11.14: Summary of applying supervised learning with dynamic generation of feature attributes.

	English F1-score		Spanish F1-score	
	NER	**SA**	**NER**	**SA**
Max	65.46	50.27	75.26	48.86
Min	57.67	42.05	66.82	42.23
AVRG	61.43	45.13	69.99	45.30
CI High	63.47	47.17	71.80	47.00
CI Low	59.38	43.09	68.19	43.59

We should spot the light to using context-based targeted analysis system with the dataset of Arabic tweets. We noticed clearly that the system could not identify all subtopics correctly. The first reason belongs to the nature of Arabic subtopics that may have many meanings. For example, word “التحرير” may be a name of place or describes the freedom. Another challenge comes from that some words are not related to main topic. Such as, word “شعب” may describe other target such as “الإمارات” which is not the same target in our case study (“مصر”).

Additionally, Arabic tweets contain various slang words that may affect performance of detecting subtopics such as “عشان”. These words cannot be tagged by POS Arabic tagger. Moreover, we cannot find word2vec embeddings that represent these words. By default, our proposed system removes these words in the preprocessing phase. However, some of these slang words may describe important topic in the Arabic tweet such as “الريس”.

11.7 Conclusion

We conclude that integrating discrete features with word2vec embeddings improves the performance of open domain targeted sentiment classification when using CRF instead of neutral network (NN) which used by Zhang et al. [125]. Moreover, adding word2vec embeddings as additional feature will improve the performance immediately without using additional feature layer in NN as used by Zhang et al. [125]. Zhang et al. [125] proposed a new technique by adding the word2vec embeddings to a separated layer when building neutral network model. While, we showed efficiency of using word2vec embeddings by just concatenating them with discrete feature attributes.

We showed in this chapter that applying SVM HMM by using discrete features with multiple word2vec embeddings outperforms all previous related works. Additionally, using our proposed semi-supervised learning solution provides competitive results with less number of labeled data. The performance of this solution is close to performance of prominent related work.

Thus, we accept the first hypothesis in our dissertation which estimates that semi-supervised technique improves the performance of open domain targeted sentiment classification. Additionally, it is a good choice for using our proposed solution when there is a lack of labeled data or preparing it needs a costly process.

Additionally, we showed that our proposed supervised learning solution with dynamic generation of feature attributes outperforms all models proposed so far. To the best of our knowledge, this proposed solution achieved results that are not reported before with any related work for open domain targeted sentiment classification. Our findings show as well that applying NER followed by target-dependent sentiment classification provides better performance in comparison with detecting both target and sentiment polarity in one shot. Thus, the first scenario in more accurate for open domain targeted sentiment classification.

Moreover, it is worth to clarify that our proposed context-based targeted analysis system works well with English tweets. However, it needs some improvements for dealing with other languages such as Arabic. Using non-English tweets adds more challenges which should be treated individually for improving the performance of our proposed solution.

CHAPTER 12

CONCLUSION AND FUTURE WORK

In this research work, we addressed two recent research problems; namely, target-dependent and open domain targeted sentiment classification. We evaluated the performance of applying numerous supervised, unsupervised and semi-supervised learning methods to both problems. New semi-supervised learning techniques are proposed for both target-dependent and open domain targeted sentiment classification by using partially labeled data. Moreover, new supervised learning techniques are proposed for improving the performance.

This chapter provides conclusions regarding the findings of the dissertation; it also discusses some suggestions for future work. Moreover, the chapter presents some threats to the validity of our findings.

12.1 Conclusion

This dissertation presents a comprehensive review on sentiment analysis in social media. A survey on target-dependent sentiment analysis is carried out also with summarized results. The survey revealed some gaps to be addressed in future research and illustrates that there are still many limitations in previous research works. Additionally, we carried out comparisons between different techniques applied to the same dataset. As a result of this, two comparison frameworks are built to validate our solutions proposed for both target-dependent and open domain targeted sentiment classification.

Performance of applying many supervised learning techniques has been evaluated and new solutions are proposed for improving the performance of target-dependent sentiment classification. The proposed solutions provided a significant increase in classification accuracy equals about 20% in comparison with accuracy of prominent method proposed in the literature. Additionally, we have addressed the difficulty of preparing labeled data from social media by proposing a new semi-supervised learning technique that uses partially labeled data.

Additionally, efficiency of using deep learning techniques has been addressed for improving the performance of target-dependent sentiment classification. We have compiled all previous works that employed deep learning techniques for both target-dependent and open domain targeted sentiment classification. We evaluated as well the efficiency of applying neural networks and deep conventional neural networks on target-dependent sentiment classification.

Moreover, two new solutions are proposed for improving the performance of open domain targeted sentiment classification. The first solution is a supervised learning technique while that other one is a semi-supervised learning technique. The best improvement in performance reported an increase by more than 4%. This increase in performance seems small but it adds a significant contribution since this research direction includes different tasks and still an open research problem. A new system has been developed also for context-based target-dependent sentiment analysis. The proposed system detects context patterns among a set of micro-blogs by detecting targets and identifying sentiment polarities towards categorized topics that describe the detected targets.

There are two scenarios for implementing open domain targeted sentiment classification. The first scenario consists of two tasks: detecting targets and then identifying sentiment polarities towards the targets. The second scenario is based on detecting targets and identifying sentiment polarities towards them in one shot. Based on our analysis and experiment work, we conclude that the first scenario provides better results in comparison with the second one.

Numerous experiments are developed in this research work to show efficacy of our proposed solutions. All experimental results show that the proposed techniques outperform all previous related works. The performance is improved when applying the proposed techniques in comparison with other prominent work.

12.2 Threats to Validity

In our target-dependent sentiment classification experiments, we used a very popular dataset used in the literature. The dataset includes unbalanced distribution of sentimental classes. We believe that applying our techniques to other datasets with different balancing schemes may result in different performance. We could not evaluate this effect because there was no other public datasets avilable in this direction. Additionally, performance of K-means method which used in our solutions is based mainly on initializing the centriods. While, initializing the centriods is based basically on the form of data and the distribution of data points. Thereby, applying the proposed techniques to other datasets may result in different performance.

When applying context-based analysis system, we remove slang words during the filtering phase. If some of those slang words are not caught during the filtering phase, they will be processed at the lower level by word2vec embeddings. In such cases, the system will not be able to find word2vec embeddings that represent all corresponding slangs. Existence of unseen (missed) word2vec embeddings decreases performance of any machine leaning technique as well as our proposed technique. To decrease effect of this problem, we developed a method for merging word2vec embeddings from three sources [125] [130] [151]. However, there are still some unseen words that do not have word2vec embeddings in the three sources.

12.3 Limitations

When applying context-based analysis system, we remove slang words during the filtering phase. Thereby, using slang words may result in a change in the accuracy of detecting targets and topics among a set of micro-blogs. Additionally, the proposed techniques remove all emoticons when filtering micro-blogs. However, employing these icons may improve the performance. The proposed techniques remove also some acronyms such as “RT” and “#”. “RT” is an acronym for a “re-tweet” while the hash-tag “#” is used to organize tweets. Thereby, using these acronyms may add more information that helps in improving the performance.

Performance of the proposed context-based analysis system is affected by accuracy of the used part of speech (POS) tagger. Thus, we need to evaluate accuracy of POS tagger especially when applying the system to non English micro-blogs such as Arabic micro-blogs. Based on our experiment work, finding accurate Arabic POS tagger is not an easy job because its performance is still limited. This task is an open research problem because there are many challenges when developing Arabic POS tagger. For example, same Arabic micro-blog may include different dialects. Additionally, we noticed clearly that the system could not identify all subtopics correctly with Arabic tweets. The reason belongs to the nature of Arabic subtopics that may have many meanings.

12.4 Future Work

Guided by the gaps identified by our literature review, and the findings and limitations of our research, we identify future work related to target dependent sentiment analysis and open-domain targeted sentiment analysis as we discuss in the sequel.

12.4.1 Target-Dependent Sentiment Analysis

This work can be extended in different directions. It is worth to investigate optimization methods such as genetic algorithms for finding the global optimum values of parameters that are used for building semi-supervised learning models in general. It will be efficient work when running more experiments to find effect of changing more than one parameter independently.

Additionally, it is worth a try to detect the best ratio of labeled data by using 5-fold cross validation applied to training data before calculating classification accuracy. It may be also a good improvement if there is a check of differences between runs while changing place of labeled data selected from training data. Moreover, we should try to use different weight for each micro-blog and check the performance when applying balanced mode.

Another research direction may improve the performance by combining more than one semi-supervised techniques such as merging our proposed solution with QN-S3VM. In the same manner, extending label propagation method may improve the performance by using lower ratio of labeled data. It is also worth a try to employ semi-supervised learning with deep learning for improving the performance.

This work can be extended also by testing performance of using other semi-supervised learning techniques. It would be interesting to develop methods for detecting minimal number of micro-blogs that are required to be labeled for providing the best performance. Such micro-blogs should form a representative sample adequate enough to classify the overall input data. Moreover, future work may investigate developing cluster-based technique for partitioning input micro-blogs and selecting specific ones that provide high performance.

Based on our experiment results, we noticed that SVM as supervised learning and K-means as unsupervised learning provide the best results. Thus, it is worth a try to merge them for developing models under umbrella of semi-supervised learning. Additionally, using other methods rather than PCA and LDA for dimension reduction may improve performance of target-dependent sentiment classification. We also should check efficiency of applying other data clustering methods instead of using K-means. Using more multicast strategies and developing new strategies may also improve the performance.

12.4.2 Open Domain Targeted Sentiment Analysis

It is clear that using word embeddings provides significant improvement in the performance. Thus, it is interesting to check efficiency of employing additional forms of word embeddings in this research direction such as using global vectors for word representation (GloVe). Additionally, it may be efficient to develop new mechanisms for generating feature attributes automatically. Moreover, developing a new optimization solution for finding optimum value needed to setting parameters (such as *C* parameter with SVM HMM) may be an important direction. It would be of interest also to employ more sequence labeling methods for improving the performance.

It is noteworthy that study deeply effect of unseen word2vec embeddings (out-of-vocabulary words) on the performance is a very important direction. Developing a new method for decreasing the effect may be a promising research direction. To develop such method, we may employ char2vec [152] embeddings instead of word2vec embeddings for finding all unseen words.

When evaluating performance of context-based analysis system, we could not show all experiment results since there is a huge data. Thus, we selected some case studies to show efficacy of the proposed system. It would be of interest to evaluate more case studies in future work. Several extensions may be achieved as well for improving performance of using non-English micro-blogs (such as Arabic) with our proposed context-based targeted analysis system. Checking efficiency of more POS taggers may improve the performance. It is important to check performance of converting slang words to standard Arabic words [153] since some slang words may include important meanings. Developing a sophisticated POS tagger that deals with both slang and standard Arabic words is expected to be a promising research direction.

References

[1] B. Pang, and L. Lee , "Opinion mining and sentiment analysis," *Foundations and* Trends in Information Retrieval, vol. 2, no. 1-2, pp. 1-135, 2008.

[2] A. Shoufan, and S. Al-Ameri "Natural Language Processing for Dialectical Arabic: A Survey," *in Second Workshop on Arabic Natural Language Processing*, July 26-31, 2015, pp. 36-48.

[3] M. Tsytsarau, and T. Palpanas, "Survey on mining subjective data on the web," *Data Mining and Knowledge Discovery*, vol. 24 no. 3, pp. 478- 514, 2012.

[4] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? Sentiment classification using machine learning techniques," *in the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2002, pp. 79-86.

[5] J. Wiebe, T. Wilson, and M. Bell, "Identifying collocations for recognizing opinions," *in ACL'01 Association for Computational Linguistics Workshop on Collocation: Computational Extraction, Analysis, and Exploitation*, 2001, pp. 24-31.

[6] G. Somprasertsri, and P. Lalitrojwong, "Mining Feature- Opinion in Online Custumer Reviews for Opinion Summarization," *Journal of Universal Computer Science*, vol. 16, pp. 938-955, 2010.

[7] A. Zubiaga, M. Liakata, R. Procter, K. Bontcheva, and P. Tolmie, "Towards Detecting Rumours in Social Media," *in AAAI Workshop on AI for Cities*, 2015, pp. 380-390.

[8] K. Wu, S. Yang, and Kenny Q. Zhu, "False Rumors Detection on Sina Weibo by Propagation Structures," *in the 31st International Conference on Data Engineering*, 2015, pp. 651-662.

[9] R. Dehkharghani, B. YANIKOGLU, Y. SAYGIN, and K. Oflazer, "Sentiment Analysis in Turkish at Different Granularity Levels," *Natural Language Engineering*, pp. 1-25, 2016.

[10] K. Veselovska, J. Hajic, Jr. and Jana Sindlerova, "Creating Annotated Resources for Polarity Classification in Czech," *in KONVENS (PATHOS workshop)*, Vienna, September 21, 2012.

[11] I. V. Tetko, D. J. Livingstone, and A. I. Luik, “Neural network studies. 1. Comparison of Overfitting and Overtraining,” *Journal of Chemical Information and Computer Sciences*, vol. 35, no. 5, pp. 826–833, 1995.

[12] S. Abudalfa, and M. Ahmed, “Survey on Target Dependent Sentiment Analysis of Micro-Blogs in Social Media,” *in 9th IEEE GCC Conference & Exhibition*, Manama, Bahrain, 7-10 May 2017.

[13] S. Abudalfa, and M. Ahmed, “Comparative Study on Efficiency of Using Supervised Learning Techniques for Target-Dependent Sentiment Polarity Classification in Social Media,” *International Journal of Computing and Digital Systems (IJCDS)*, vol. 7, no. 3, 2018.

[14] S. Abudalfa, and M. Ahmed, “Deep Learning for Target-Dependent Sentiment Classification in Social Media,” *Smart Cities Symposium*, Bahrain, 2018.

[15] S. O. Alhumoud, M. I. Altuwaijri, T. M. Albuhairi, and W. M. Alohaideb, “Survey on Arabic Sentiment Analysis in Twitter,” *International Journal of Social, Behavioral, Educational, Economic and Management Engineering*, vol. 9, no.1, 2015.

[16] S. Ahmed, and G. Qadah. "Key Issues in Conducting Sentiment Analysis on Arabic Social Media Text," *in 9th International Conference on Innovations in Information Technology (IIT)*, 2013, pp. 72-77.

[17] N. Abdulla, N. Ahmed, M. Shehab, and M. Al-Ayyoub, “Arabic Sentiment Analysis: Lexicon-Based and Corpus-Based,” *in IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, 2013, pp. 1-6.

[18] M. Saleh, *Sentiment Analysis in Arabic: Opinion Polarity Detection*, PhD thesis, UNIVERSIDAD DE JAÉN, 2013.

[19] S. El-Beltagy, and A. Ali, “Open Issues in the Sentiment Analysis of Arabic Social Media: A Case Study,” *in 9th International Conference on Innovation in Information Technology (IIT)*, 2013, pp.215-220.

[20] L. Rokach, and O. Maimon, *Data mining with decision trees: theory and applications*, World Scientific Pub Co Inc, 2008.

[21] H. Zhang, “The optimality of Naive Bayes,” *in FLAIRS*, 2004.

[22] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer, Section 4.3, p.106-119, 2008.

[23] J. L. Bentley, “Multidimensional binary search trees used for associative searching,” *Communications of the ACM*, vol. 18, no. 9, pp. 509-517, Sep. 1975.

[24] S. Abudalfa, and M. Mikki, “A Dynamic Linkage Clustering using KD-Tree,” *International Arab Journal of Information Technology (IAJIT)*, vol. 10, no. 3, May 2013.

[25] R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu, “Diagnosis of multiple cancer types by shrunken centroids of gene expression,” *in the National Academy of Sciences of the United States of America*, vol. 99, no. 10, 2002, pp. 6567-6572.

[26] Y. Tsuruoka, J. Tsujii, and S. Ananiadou, “Stochastic Gradient Descent Training for L1-regularized Log-linear Models with Cumulative Penalty,” *in the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, Suntec, Singapore, 2-7 August 2009, pp. 477–485.

[27] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, “Online Passive-Aggressive Algorithms,” *Journal of Machine Learning Research*, vol. 7, pp. 551–585, 2006.

[28] C. Scholkopf, J. C. Burges, and A. J. Smola, *Advances in Kernel Methods: Support Vector Learning*, MIT Press, 1999.

[29] Y. LeCun, L. Bottou, G. Orr, and K. Müller, “Efficient BackProp,” *Neural Networks: Tricks of the Trade*, 1998.

[30] P. Sermanet and Y. LeCun “Traffic Sign Recognition with Multi-Scale Convolutional Networks,” *in the International Joint Conference on Neural Networks (IJCNN)*, 2011.

[31] R. Socher, C. Lin, A. Ng, and C. Manning. “Parsing Natural Scenes and Natural Language with Recursive Neural Networks,” *in ICML*, 2011.

[32] M. Schuster, K. Paliwal, “Bidirectional Recurrent Neural Networks,” *IEEE Transactions on Signal Processing,* 1997.

[33] S. Hochreiter, J. Schmidhuber, “Long Short-Term Memory,” Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[34] K. Cho, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation,” *in the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[35] J. Pennington, R. Socher, and C. D. Manning, “GloVe: Global Vectors for Word Representation,” *in Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[36] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *in International Conference on Learning Representations (ICLR)*, 2013.

[37] R. Al-Rfou, B. Perozzi, and S. Skiena, “Polyglot: Distributed Word Representations for Multilingual NLP,” *in 17th Conference on Computational Natural Language Learning*, 2013, pp. 183—192.

[38] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, “Enriching Word Vectors with Subword Information,” *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[39] B. Liu, *SENTIMENT ANAYSIS: Mining Opinions, Sentiments, and Emotions*, Cambridge University Press, 2015.

[40] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for large databases,” *in the ACM SIGMOD Intl. Conference on Management of Data (SIGMOD)*, 1996, pp. 103-114.

[41] C. Ding, and X. He, "K-means Clustering via Principal Component Analysis," *in the 21st International Conference on Machine Learning*, Banff, Canada, July 2004, pp. 225–232.

[42] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-supervised Learning*, The MIT Press, London, England, 2006.

[43] J. Turian, L. Ratinov, and Y. Bengio, “Word representations: A simple and general method for semi-supervised learning,” *in the 48th Annual Meeting of the Association for Computational Linguistics*, Uppsala, Sweden, 11-16 July 2010, pp. 384–394.

[44] X. Zhu, and Z. Ghahramani, “Learning from labeled and unlabeled data with label propagation,” *Technical Report*, Carnegie Mellon University, 2002.

[45] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf, “Learning with local and global consistency,” *Advances in Neural Information Processing Systems 16*, 2004.

[46] S. Basu, A. Banerjee, and R. Mooney, “Semi-supervised Clustering by Seeding,” *in the 19th International Conference on Machine Learning (ICML-2002)*, Sydney, Australia, July 2002, pp. 19-26.

[47] X. Wang, C. Wang, and J. Shen, “Semi–supervised K-Means Clustering by Optimizing Initial Cluster Centers,” *Web Information Systems and Mining*, vol. 6988 of the series Lecture Notes in Computer Science, pp. 178-187, 2011.

[48] S. Ravi, “Semi-supervised Learning in Support Vector Machines,” *Project Report*, Princeton University, 2014.

[49] K. Nigam, A. McCallum, and T. Mitchell, *Semi-supervised Text Classification Using EM*, The MIT Press, London, England, ch. 3, pp. 33-55, 2006.

[50] F. Gieseke, A. Airolab, T. Pahikkalab, and O. Kramera, “Fast and Simple Gradient-Based Optimization for Semi-supervised Support Vector Machines,” *Technical Report*, 2013.

[51] F. Gieseke, A. Airola, T. Pahikkala, and O. Kramer, “Sparse Quasi-Newton Optimization for Semi-supervised Support Vector Machines,” *Technical Report*, 2012.

[52] X. Zhu, “Semi-supervised Learning Literature Survey”, *Computer Sciences Technical Report 1530*, University of Wisconsin–Madison, 2005.

[53] M. Aly, “Survey on Multiclass Classification Methods,” *Technical Report*, California Institute of Technology, November 2005.

[54] A. Miranda, Y.-A. Borgne, and G. Bontempi, “New Routes from Minimal Approximation Error to Principal Components,” *Neural Processing Letters, Springer*, vol. 27, no. 3, June, 2008.

[55] H. Abdi, "Discriminant correspondence analysis," *in N.J. Salkind (Ed.): Encyclopedia of Measurement and Statistic*, Thousand Oaks (CA), Sage, 2007, pp. 270–275.

[56] E. Sang, and F. Meulder, "Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition," *in the 7th Conference on Natural Language Learning (CONLL '03)*, vol. 4, Stroudsburg, PA, 2003, pp. 142-147.

[57] L. Ratinov, and D. Roth, "Design Challenges and Misconceptions in Named Entity Recognition," *in the 13th Conference on Computational Natural Language Learning (CoNLL '09)*, Stroudsburg, PA, 2009, pp. 147-155.

[58] N. Nguyen, and Y. Guo, “Comparisons of Sequence Labeling Algorithms and Extensions,” *in the 24th International Conference on Machine Learning*, Corvallis, OR, 2007, pp 681–688.

[59] Y. Altun, I. Tsochantaridis, and T. Hofmann, “Hidden Markov Support Vector Machines,” *in the 20th International Conference on Machine Learning (ICML-2003)*, Washington DC, 2003, pp. 3–10.

[60] A. Abbasi, H. Chen, and A. Salem, “Sentiment Analysis in Multiple Languages: Features selection for Opinion Classification in Web Forums,” *ACM Transactions on Information Systems (TOIS)*, vol. 26, no. 3, pp. 1-34, 2008.

[61] D. de Kok, and H. Brouwer, “N-gram,” *Creative Commons Attribution 3.0*, 2010.

[62] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau, “Sentiment Analysis of Twitter Data,” *in the Workshop on Languages in Social Media, ser. LSM ’11, Association for Computational Linguistics*, 2011, pp. 30–38.

[63] L. Sayfullina, *Reducing Sparsity in Sentiment Analysis Data using Novel Dimensionality Reduction Approaches*, Master Thesis, Aalto University, October 2014.

[64] M. Abdul-Mageed, M. T. Diab, and M. Korayem, “Subjectivity and Sentiment Analysis of Modern Standard Arabic,” *in the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011, pp. 587–591.

[65] D. Davidov, O. Tsur, and A. Rappoport, “Enhanced Sentiment Learning Using Twitter Hashtags and Smileys,” *in the 23rd International Conference on Computational Linguistics: Posters*, 2010, pp. 241-249.

[66] H. Saif, Y. He, and H. Alani, “Semantic Sentiment Analysis of Twitter,” *in the 11th International Semantic Web Confernce (ISWC’12)*, 2012, pp. 508-524.

[67] M. Elhawary, and M. Elfeky, “Mining Arabic Business Reviews,” *in International Conference in Data Mining Workshops (ICDMW)*, 2010, pp. 1108-1113.

[68] E. Refaee, and V. Rieser, “Subjectivity and Sentiment Analysis of Arabic Twitter Feeds with Limited Resources,” *in 9th International Conference on Language Resources and Evaluation,* 2014.

[69] G. Vinodhini, and RM. Chandrasekaran, “Sentiment Analysis and Opinion Mining: A Survey,” *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, no. 6, June 2012.

[70] P. Gonçalves, and M. Araújo, “Comparing and Combining Sentiment Analysis Methods,” t*he ACM Conference on Social Netowrks (COSN’13),* Oct. 2013.

[71] L. Zhang, R. Ghosh, M. Dekhil, M. Hsu, and B. Liu, *Combining Lexicon-based and Learning-based Methods for Twitter Sentiment Analysis*, Hewlett-Packard Development Company, L.P., 2011.

[72] R. de Groot, *Data Mining for Tweet Sentiment Classification*, Master thesis, Utrecht University, July 23, 2012.

[73] A. Go, R. Bhayani, and L. Huang, "Twitter Sentiment Classification using Distant Supervision," *CS224N Project Report*, Stanford, 2009.

[74] M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, and A. Kappas, "Sentiment strength detection in short informal text," *Journal of the American Society for Information Science and Technology*, vol. 61, no. 12, 2544- 2558, 2010.

[75] A. Kumar, and T. Mary Sebastian, "Sentiment analysis on twitter," *IJCSI International Journal of Computer Science Issues*, vol. 9, no. 3, pp. 372-378, 2012.

[76] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment-specific word embedding for Twitter sentiment classification," *in the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014, pp. 1555-1565.

[77] M. Yang, W. Tu, Z. Lu, W. Yin, and K.-P. Chow, "LCCT: A Semi-supervised Model for Sentiment Classification," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015, pp. 546–555.

[78] A. Bifet, and E. Frank, "Sentiment Knowledge Discovery in Twitter Streaming Data," *in the 13th International Conference on Discovery Science*, Springer Berlin Heidelberg, 2010, pp. 1-15.

[79] B. Liu, *Sentiment Analysis and Opinion Mining*, Morgan & Claypool Publishers, May 2012.

[80] Y. Mejova, *Sentiment analysis within and across social media streams*, PhD thesis, University of Iowa, 2012.

[81] M. Wu, *Using Clustering and Sentiment Analysis on Twitter,* Master thesis, Texas A&M University-Corpus Christi, 2014.

[82] R. Biagioni, *The SenticNet Sentiment Lexicon: Exploring Semantic Richness in Multi-Word Concepts*, Master thesis, March 2015.

[83] K. Bloom, *Sentiment Analysis Based on Appraisal Theory and Functional Local Grammars*, PhD thesis, 2011.

[84] A. Mourad, and K. Darwish, “Subjectivity and Sentiment Analysis of Modern Standard Arabic and Arabic Microblogs,” *in the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, June 2013, pp. 55–64.

[85] M. Mayo, “A Clustering Analysis of Tweet Length and its Relation to Sentiment,” 2014.

[86] M. Ibrahim, and N. Salim, ”Opinion Analysis For Twitter and Arabic Tweets: A Systematic Literature Review,” *Journal of Theoretical and Applied Information Technology*, vol. 56 no. 2, 20th October 2013.

[87] M. A. Haque, T. Rahman, “Sentiment Analysis by Using Fuzzy Logic,” *International Journal of Computer Science, Engineering and Information Technology (IJCSEIT)*, vol. 4, no. 1, 2014.

[88] D. N. Prata, K. P. Soares, M. A. Silva, D. Q. Trevisan, and P. Letouze, “Social Data Analysis of Brazilian’s Mood from Twitter,” *International Journal of Social Science and Humanity*, vol. 6, no. 3, March 2016.

[89] Z. Zhao, P. Resnick, and Q. Mei, “Enquiring Minds: Early Detection of Rumors in Social Media from Enquiry Posts,” *in International World Wide Web Conference Committee (IW3C2)*, May 2015.

[90] J. Ma, W. Gao, Z. Wei, Y. Lu, and K. Wong, “Detect Rumors Using Time Series of Social Context Information on Microblogging Websites,” *in CIKM’15*, ACM, October 19–23, 2015.

[91] E. Seoa, P. Mohapatrab, and T. Abdelzahera, “Identifying Rumors and Their Sources in Social Networks,” *in SPIE Defense, Security, and Sensing , International Society for Optics and Photonics*, May, 2012, pp. 83891I-83891I.

[92] S. Vosoughi, *Automatic Detection and Verification of Rumors on Twitter*, PhD thesis, Massachusetts Institute of Technology, 2015.

[93] L. Dong, F. Wei, C. Tan, D. Tang, M. Zhou, and K. Xu, “Adaptive Recursive Neural Network for Target-dependent Twitter Sentiment Classification,” *in the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers)*, June 2014, pp. 49–54.

[94] C. Quan, and F. Ren, “Target Based Review Classification for Fine-Grained Sentiment Analysis,” *International Journal of Innovative Computing, Information and Control*, vol. 10, no. 1, 2014.

[95] Z. Zhang, and M. Lan, “ECNU: Extracting Effective Features from Multiple Sequential Sentences for Target-dependent Sentiment Analysis in Reviews,” *in 9th International Workshop on Semantic Evaluation (SemEval 2015)*, June 4-5, 2015, pp. 736–741.

[96] D. Vo, and Y. Zhang, “Target-Dependent Twitter Sentiment Classification with Rich Automatic Features,” *in the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, 2015, pp. 1347–1353.

[97] X. Hu, J. Tang, H. Gao, and H. Liu, “Unsupervised Sentiment Analysis with Emotional Signals,” *in International World Wide Web Conference Committee (IW3C2)*, May 2013, pages 607–618.

[98] L. Jiang, M. Yu, M. Zhou, X. Liu, and T. Zhao, “Target-dependent Twitter Sentiment Classification,” *in 49th Annual Meeting of the Association for Computational Linguistics*, June 2011, pp. 151-160.

[99] S. M. Mohammad, S. Kiritchenko, and X. Zhu, “NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets,” *in the 7th international workshop on Semantic Evaluation Exercises (SemEval-2013)*, 2013.

[100] R. Bosch, *Sentiment Analysis: Incremental learning to build domain models*, Master thesis, Universitat Pompeu Fabra, 2013.

[101] E. Duarte, *Sentiment Analysis on Twitter for the Portuguese Language*, Master Thesis , Universidade Nova de Lisboa, 2013.

[102] X. Fang, and J. Zhan, “Sentiment analysis using product review data,” *Journal of Big Data*, vol. 2, no. 1, pp. 1-14, 2015.

[103] J. Smailovic, *Sentiment Analysis in Streams of Microblogging Posts*, PhD thesis, Jožef Stefan International Postgraduate School, Ljubljana, Slovenia, November 2014.

[104] R. Suchdev, P. Kotkar, R. Ravindran, and S. Swamy, “Twitter Sentiment Analysis using Machine Learning and Knowledge-based Approach,” *International Journal of Computer Applications*, vol. 103, no.4, pp. 0975 – 8887, October 2014.

[105] S. Kiritchenko, X. Zhu, and S. M. Mohammad, “Sentiment Analysis of Short Informal Texts,” *Journal of Artificial Intelligence Research*, pp. 723-762, 2014.

[106] S. Narr, M. Hulfenhaus, and S. Albayrak, “Language-Independent Twitter Sentiment Analysis,” *in Knowledge Discovery and Machine Learning (KDML)*, LWA, 2012, pp. 12–14.

[107] G. Gebremeskel, *Sentiment Analysis of Twitter Posts About News*, Master thesis, University of Malta, 2011.

[108] T. Gunther, *Sentiment Analysis of Microblogs*, Master Thesis, University of Gothenburg, June 2013.

[109] G. Valkanas, I. Katakis, and D. Gunopulos, "Mining Twitter Data with Resource Constraints," *in the 2014 IEEE/WIC/ACM International Conference on Web Intelligence*, August 2014.

[110] R. Soni, and K. J. Mathai, "Improved Twitter Sentiment Prediction through 'Cluster-then-Predict Model'," *IJCSN International Journal of Computer Science and Network*, vol. 4, no. 4, 2015.

[111] J. Zhao, L. Dong, J. Wu, and K. Xu, "MoodLens: An Emoticon-Based Sentiment Analysis System for Chinese Tweets," *in 18th ACM SIGKDD International Confernece on Knowledge Discovery and Data Mining*, 2012, pp. 1528-1531.

[112] A. Cimino, S. Cresci, F. Dell'Orletta, and M. Tesconi, "Linguistically–motivated and Lexicon Features for Sentiment Analysis of Italian Tweets," *in 4th evaluation campaign of Natural Language Processing and Speech tools for Italian,* 2014, pp. 81-86.

[113] R. Batool, A. M. Khattak, J. Maqbool, and S. Lee, "Precise Tweet Classification and Sentiment Analysis," *in IEEE/ACIS 12th International Conference on Computer and Information Science (ICIS)*, June, 2013, pp. 461-466.

[114] A. Bakliwal, P. Arora, S. Madhappan, N. Kapre, M. Singh, and V. Varma, "Mining Sentiments from Tweets," *in 3rd Workshop on Sentiment and Subjectivity Analysis in Conjunction with 50th annual meeting of Association for Computational Linguistics*, 2012.

[115] R. M. Duwairi, R. Marji, N. Sha'ban, and S. Rushaidat, "Sentiment Analysis in Arabic Tweets," *in 5th international conference on information and communication systems*, April 2014, 2014, pp. 1–6.

[116] M. Abdul-Mageed, M. Diab, and S. Kübler, "SAMAR: Subjectivity and sentiment analysis for Arabic social media," *Computer Speech and Language*, vol. 28, no. 1, pp. 20-37, 2014.

[117] H. S. Ibrahim, S. M. Abdou, and M. Gheith, "Sentiment Analysis For Modern Standard Arabic and Colloquial," *International Journal on Natural Language Computing (IJNLC)*, vol. 4, no.2, April 2015.

[118] N. El-Makky, K. Nagi, A. El-Ebshihy, E. Apady, O. Hafez, S. Mostafa, and S. Ibrahim, “Sentiment Analysis of Colloquial Arabic Tweets,” *in ASE BigData/SocialInformatics/PASSAT/BioMedCom Conference*, Harvard University, December 14-16, 2014.

[119] M. N. Al-Kabi, and A. H. Gigieh, “Opinion Mining and Analysis for Arabic Language,” *International Journal of Advanced Computer Science and Applications*, vol. 5, no. 5, pp. 181-195, 2014.

[120] A. Hamouda, and F. El-taher, “Sentiment Analyzer for Arabic Comments System,” *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 4, no.3, 2013.

[121] A. M. Shoukry, *Arabic Sentence Level Sentiment Analysis*, Master Thesis, the American University in Cairo, Spring 2013.

[122] S. Ahmed, *Sentiment Mining of Arabic Twitter Data*, Master thesis, American University of Sharjah, January 2014.

[123] M. Yang, W. Tu, Z. Lu, W. Yin, and K.-P. Chow, "LCCT: A Semi-supervised Model for Sentiment Classification," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015, pp. 546–555.

[124] M. Mitchell, J. Aguilar, T. Wilson, and B. Durme, “Open Domain Targeted Sentiment,” *in the 2013 Conference on Empirical Methods in Natural Language Processing*, October 2013, pp. 1643-1654.

[125] M. Zhang, Y. Zhang, and D. Vo, “Neural Networks for Open Domain Targeted Sentiment,” *in the 2015 Conference on Empirical Methods in Natural Language Processing*, Sep. 2015, pp. 612–621.

[126] Y. Wang, H. Gao, and S. Geng "A Target-Dependent Sentiment Analysis Method for Micro-blog Streams," in *18th Asia-Pacific Web Conference, Web Technologies and Applications, APWeb 2016, Part{II}*, September 2016, pp. 30-42.

[127] D. Tang, B. Qin, X. Feng, and T. Liu, “Effective LSTMs for Target-Dependent Sentiment Classification,” *in the 26th International Conference on Computational Linguistics: Technical Papers*, December 2016, pp. 3298–3307.

[128] M. Zhang, Y. Zhang, and D. Vo, “Gated Neural Networks for Targeted Sentiment Analysis,” *in the 30th AAAI Conference on Artificial Intelligence*, February 2016, pp. 3087-3093.

[129] N. Farra, and K. McKeown, “SMARTies: Sentiment Models for Arabic Target Entities,” *European Chapter of the Association for Computational Linguistics*, Jan 2017.

[130] H. Li, and W. Lu, “Learning Latent Sentiment Scopes for Entity-Level Sentiment Analysis,” *in 31st AAAI Conference on Artificial Intelligence (AAAI-17)*, 2017, pp. 3482- 3489.

[131] M. Jabreel and A. Moreno, “Target-dependent Sentiment Analysis of Tweets using a Bi-directional Gated Recurrent Unit,” *Smart Innovation, Systems and Technologies*, SIST, vol. 85, 2018.

[132] K. Ravi and V. Ravi, “A survey on Opinion Mining and Sentiment Analysis: Tasks, Approaches and Applications,” *Knowledge-Based Systems* vol. 89, pp. 14–46, 2015.

[133] B. Fang, Y. Li, H. Zhang, and J.C.-W. Chan, “Semi-supervised Deep Learning Classification for Hyperspectral Image Based on Dual-Strategy Sample Selection,” *Remote Sensing*, vol. 10, no. 4, 574, 2018.

[134] C. Metz, “Basic principles of ROC analysis,” *Semin Nucl Med.*, vol. 8, no. 4, pp. 283–98, Oct 1978.

[135] S. Parambath, N. Usunier, and Y. Grandvalet, “Optimizing F-measures by Cost-Sensitive Classification,” *in Neural Information Processing Systems (NIPS)*, 2014, pp. 2123-2131.

[136] A. El-Kilany, A. Azzam, and S. El-Beltagy, “Using Deep Neural Networks for Extracting Sentiment Targets in Arabic Tweets,” *Studies in Computational Intelligence*, 2017.

[137] P. Juszczak, D. Tax; R. Dui, “Feature scaling in support vector data descriptions,” *in 8th Annu. Conf. Adv. School Comput. Imaging*, 2002, pp. 25-30.

[138] S. Ahire, “A Survey of Sentiment Lexicons,” IIT Bombay, 2015.

[139] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” *in Neural Information Processing Systems (NIPS)*, 2013.

[140] S. Abudalfa, and M. Mikki, “K-Means Algorithm with a Novel Distance Measure,” *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 21, no. 6, pp. 1665-1684, Oct 2013.

[141] A. Gulli and S. Pal, *Deep Learning with Keras*, Packt Publishing Ltd, 2017.

[142] D. Arthur, and S. Vassilvitskii "k-means++: the advantages of careful seeding," *in eighteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics Philadelphia*, PA, USA, 2007, pp. 1027-1035.

[143] B. Yang and C. Cardie "Context-aware Learning for Sentence-level Sentiment Analysis with Posterior Regularization," *in the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014, pp. 325–335.

[144] G. Katz, N. Ofek, and B. Shapira, "ConSent: Context-based sentiment analysis,", *Knowledge-Based Systems*, vol. 84, pp. 162–178, 2015.

[145] S. Sharma, S. Chakraverty, A. Sharma, J. Kaur, "A context-based algorithm for sentiment analysis," *International Journal of Computational Vision and Robotics (IJCVR)*, vol. 7, no. 5, 2017.

[146] A. Vanzo, D. Croce, R. Basili ,"A context-based model for Sentiment Analysis in Twitter," *in the 25th International Conference on Computational Linguistics: Technical Papers*, 2014, pp. 2345–2354.

[147] A. Vanzo, G. Castellucci, D. Croce, and R. Basili, "A context based model for Sentiment Analysis in Twitter for the Italian Language," *in the First Italian Conference on Computational Linguistics CLiC-it & the Fourth International Workshop EVALITA*, 2014, pp. 379–383.

[148] S. Sathiya Keerthi and S. Sundararajan "CRF versus SVM-Struct for Sequence Labeling," *Technical Report*, Yahoo! Research, 2007.

[149] Y. Qi, P. Kuksa, R. Collobert, K. Sadamasa, K. Kavukcuoglu and J. Weston, "Semi-supervised Sequence Labeling with Self-Learned Features," *in 9th IEEE International Conference on Data Mining (ICDM 09)*, 2009, pp. 428–437.

[150] R. Al-Rfou, B. Perozzi, and S. Skiena, "Polyglot: Distributed Word Representations for Multilingual NLP," *in 17th Conference on Computational Natural Language Learning*, 2013, pp. 183—192.

[151] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, "Enriching Word Vectors with Subword Information," arXiv preprint arXiv:1607.04606, 2017.

[152] K. Cao and M. Rei, "A Joint Model for Word Embedding and Word Morphology," arXiv:1606.02601v, cs.CL, 2016.

[153] K. Shaalan, H. M. Abo Bakr, and I. Ziedan, "Transferring Egyptian Colloquial Dialect into Modern," *in the International Conference on Recent Advances in Natural Language Processing (RANLP)*, Borovets, Bulgaria, 2007, pp. 525-529.

Appendix I

EXPERIMENT RESULTS: SUPERVISED LEARNING FOR TARGET-DEPENDENT SENTIMENT ANALYSIS

This appendix includes all details of experiment results that are illustrated in Chapter 7 for target dependent sentiment analysis.

Table I.1: Regularized linear model with different settings.

Loss	**Penalty**	**Accuracy**	**Macro-F1**
hinge	None	63.6	62.4
hinge	None	66.3	64.3
hinge	None	67.5	63.5
hinge	None	61.6	61.5
hinge	None	67.5	65.9
hinge	None	66.2	64.2
hinge	None	66.3	62.1
hinge	None	63.4	62.0
hinge	None	64.2	62.7
hinge	None	64.2	62.3
hinge	None	65.5	61.9
hinge	None	60.7	59.4
MAX		**67.5**	**65.9**
MIN		**60.7**	**59.4**
AVERAGE		**64.8**	**62.7**
CI HIGH		**66.4**	**63.9**
CI LOW		**63.3**	**61.5**

Table I.2: Results of applying passive aggressive classifier.

Classifying Training Data		Classifying Testing Data		C
Accuracy	**F1-score**	**Accuracy**	**F1-score**	
54.6	35.2	54.5	36.2	10^{-5}
62.7	53.7	61.3	52.9	3×10^{-5}
64.9	58.3	64.9	59.0	5×10^{-5}
64.8	57.7	64.2	57.4	7×10^{-5}
66.7	61.0	65.3	59.7	9×10^{-5}
68.3	64.4	67.2	63.7	0.0001
71.3	69.9	68.1	66.6	0.0003
70.6	66.0	66.3	60.8	0.0005
73.3	71.4	**69.7**	**67.3**	**0.0007**
74.2	72.8	68.2	66.8	0.0009
74.1	71.2	68.2	64.5	0.001
58.1	59.1	53.0	53.3	0.003
67.3	60.0	64.2	56.5	0.005
54.6	53.8	50.1	49.6	0.007
68.2	60.5	64.3	55.8	0.009
62.5	54.9	56.9	50.4	0.01
75.1	73.5	66.9	64.9	0.03
44.0	42.4	42.9	42.3	0.05
63.4	63.8	54.2	54.5	0.07
53.8	54.7	49.4	50.3	0.09
71.4	66.8	64.9	59.3	0.1
74.3	71.6	65.6	61.9	0.3
72.5	71.2	65.2	64.0	0.5
70.4	64.4	67.1	59.7	0.7
71.5	69.8	63.4	60.9	0.9
71.1	67.5	65.9	63.6	1.0
69.0	69.1	62.0	62.1	3.0
51.2	51.7	46.0	45.9	5.0
65.1	65.2	56.9	56.9	7.0
53.9	54.9	49.1	49.5	9.0

Table I.3: Linear SVC with OvR multiclass strategy.

Classifying Training Data		**Classifying Testing Data**		**C**
Accuracy	**F1-score**	**Accuracy**	**F1-score**	
54.1	33.6	53.9	34.6	10^{-5}
61.5	50.9	60.7	50.8	3×10^{-5}
64.8	57.4	63.6	56.4	5×10^{-5}
66.1	60.0	65.0	59.1	7×10^{-5}
67.2	61.8	65.9	60.5	9×10^{-5}
67.5	62.4	66.8	61.7	0.0001
71.0	67.7	69.4	65.8	0.0003
72.5	69.6	69.7	66.6	0.0005
73.6	70.9	70.4	67.7	0.0007
74.2	71.7	69.8	67.0	0.0009
74.6	72.1	69.9	67.1	0.001
78.0	76.1	**70.5**	**68.1**	**0.003**
79.8	78.2	69.7	67.2	0.005
80.8	79.3	69.4	67.0	0.007
81.5	80.0	68.9	66.5	0.009
81.9	80.5	68.6	66.3	0.01
86.1	85.3	67.3	65.3	0.03
87.9	87.3	66.3	64.2	0.05
89.2	88.7	65.3	63.1	0.07
90.0	89.6	64.7	62.5	0.09
90.3	89.9	64.7	62.6	0.1
93.9	93.7	62.0	59.7	0.3
95.1	94.9	61.0	58.9	0.5
95.4	95.3	60.8	58.4	0.7
95.8	95.7	61.1	58.9	0.9
95.4	95.3	61.0	58.8	1.0
91.4	91.0	61.3	57.8	3.0
95.9	95.8	60.3	58.1	5.0
91.8	91.5	60.4	57.1	7.0
96.0	95.9	59.4	57.7	9.0

Table I.4: Linear SVC with OvO multiclass strategy.

Classifying Training Data		**Classifying Testing Data**		**C**
Accuracy	**F1-score**	**Accuracy**	**F1-score**	
52.9	30.1	52.0	29.9	10^{-5}
59.8	47.2	59.5	48.2	3×10^{-5}
63.1	54.4	62.3	54.3	5×10^{-5}
65.1	58.2	64.0	57.4	7×10^{-5}
66.4	60.4	64.7	59.0	9×10^{-5}
66.8	61.1	64.7	59.1	0.0001
70.7	67.1	68.2	64.3	0.0003
72.6	69.7	70.2	67.1	0.0005
73.6	71.0	69.9	67.0	0.0007
74.5	72.1	**70.7**	**67.9**	**0.0009**
74.7	72.3	70.5	67.7	0.001
77.8	75.9	70.2	68.1	0.003
80.1	78.5	69.5	67.3	0.005
81.6	80.2	69.1	66.9	0.007
82.6	81.3	68.8	66.6	0.009
83.0	81.7	68.9	66.8	0.01
87.4	86.7	66.2	64.1	0.03
90.0	89.6	66.3	64.4	0.05
91.8	91.5	64.7	62.7	0.07
92.9	92.6	64.0	62.0	0.09
93.3	93.1	64.0	62.0	0.1
97.1	97.1	61.7	59.7	0.3
98.4	98.4	60.1	58.2	0.5
98.9	98.9	60.4	58.7	0.7
99.0	99.0	60.3	58.8	0.9
98.6	98.6	60.5	58.3	1.0
98.8	98.7	59.8	58.3	3.0
99.3	99.3	61.7	59.9	5.0
99.5	99.5	60.0	58.4	7.0
99.1	99.1	60.4	58.0	9.0

Table I.5: Unbalanced C-SVC linear kernel.

Classifying Training Data		**Classifying Testing Data**		**C**
Accuracy	**F1-score**	**Accuracy**	**F1-score**	
50.0	22.2	50.0	22.2	10^{-5}
50.1	22.4	50.0	22.2	3×10^{-5}
51.5	26.3	50.1	24.6	5×10^{-5}
51.8	27.1	50.6	25.7	7×10^{-5}
52.5	29.1	51.7	28.4	9×10^{-5}
52.7	29.8	51.6	28.8	0.0001
61.2	50.8	61.4	52.7	0.0003
64.9	57.6	64.2	57.4	0.0005
67.1	61.5	65.9	60.4	0.0007
68.2	63.4	67.3	62.5	0.0009
68.6	64.0	67.9	63.2	0.001
72.9	70.0	69.5	65.9	0.003
74.9	72.5	70.1	66.9	0.005
76.3	74.2	70.2	67.3	0.007
77.0	75.0	69.9	67.4	0.009
77.4	75.4	**70.2**	**67.7**	**0.01**
82.1	80.8	68.9	66.6	0.03
83.9	82.9	67.6	65.8	0.05
85.6	84.8	66.5	64.6	0.07
86.5	85.8	65.9	64.1	0.09
87.1	86.4	65.6	63.8	0.1
92.2	92.0	63.3	61.5	0.3
94.5	94.4	62.1	60.3	0.5
95.7	95.6	60.7	58.9	0.7
96.6	96.6	59.7	57.9	0.9
96.9	96.9	59.2	57.5	1.0
99.6	99.6	56.8	55.1	3.0
99.9	99.9	57.2	55.4	5.0
99.9	99.9	56.8	55.0	7.0
100.0	100.0	56.8	55.1	9.0

Table I.6: Balanced C-SVC linear kernel.

Classifying Training Data		Classifying Testing Data		C
Accuracy	**F1-score**	**Accuracy**	**F1-score**	
50.0	22.2	50.0	22.2	10^{-5}
60.1	56.0	59.5	55.1	3×10^{-5}
60.7	58.6	59.5	57.0	5×10^{-5}
61.2	59.6	60.5	58.2	7×10^{-5}
62.3	60.9	62.9	60.9	9×10^{-5}
62.8	61.4	63.4	61.5	0.0001
66.0	65.1	65.8	64.5	0.0003
66.7	66.0	65.2	64.1	0.0005
67.4	66.8	64.6	63.6	0.0007
67.9	67.3	64.2	63.2	0.0009
68.1	67.5	64.3	63.3	0.001
70.9	70.5	65.5	64.8	0.003
72.4	72.0	65.3	64.5	0.005
73.9	73.5	66.0	65.2	0.007
75.1	74.7	**67.6**	**66.9**	**0.009**
75.5	75.2	67.5	66.8	0.01
80.2	80.0	65.2	64.4	0.03
82.6	82.4	64.2	63.2	0.05
84.5	84.4	64.0	63.0	0.07
85.6	85.5	63.2	62.1	0.09
86.1	86.0	62.9	61.7	0.1
91.1	91.2	62.0	61.0	0.3
93.5	93.5	59.8	58.4	0.5
94.9	95.0	59.8	58.2	0.7
95.8	95.8	59.5	57.9	0.9
96.2	96.2	59.0	57.3	1.0
99.2	99.2	57.1	55.4	3.0
99.8	99.8	57.7	56.0	5.0
99.9	99.9	57.4	55.7	7.0
99.9	99.9	57.1	55.4	9.0

Table I.7: Neural networks with the best achieved settings.

Run #	Accuracy	macro-average F1-score
1	69.8	67.2
2	68.6	65.3
3	69.4	65.8
4	70.7	68.1
5	70.7	68.6
6	70.7	68.0
7	69.5	67.0
8	69.8	67.5
9	69.5	66.7
10	70.4	67.6
11	68.5	66.0
12	70.8	68.3
MAX	**70.8**	**68.6**
MIN	**68.5**	**65.3**
AVRG	**69.9**	**67.2**
CI High	**70.4**	**67.9**
CI Low	**69.4**	**66.5**

Appendix II

EXPERIMENT RESULTS: SEMI-SUPERVISED LEARNING FOR TARGET-DEPENDENT SENTIMENT ANALYSIS

This appendix includes all details of experiment results that are illustrated in chapter 8 for target dependent sentiment analysis.

Table II.1: Effect of changing labeling ratio when using label propagation with RBF kernel and Gamma=0.07.

Ratio	R1	R2	R3	R4	Max	Average
0.01	60.5	60.3	49.3	50.0	60.5	56.1
0.03	60.1	59.7	48.1	49.7	60.1	55.5
0.05	60.5	59.5	49.0	48.0	60.5	55.5
0.07	60.8	59.2	47.8	47.5	60.8	55.3
0.09	60.8	59.0	47.4	48.4	60.8	55.3
0.11	60.8	59.5	46.0	46.0	60.8	54.6
0.13	60.8	59.1	49.4	47.1	60.8	55.5
0.15	60.5	58.7	48.3	47.0	60.5	55.0
0.17	60.3	58.7	48.4	48.0	60.3	55.1
0.19	59.5	59.1	46.1	48.4	59.5	54.5
0.21	59.8	58.7	46.8	45.7	59.8	54.2
0.23	60.0	58.8	47.1	49.1	60.0	55.0
0.25	59.8	59.1	47.8	48.3	59.8	55.0
0.27	60.3	59.1	46.2	47.0	60.3	54.6
0.29	60.3	59.2	48.0	46.1	60.3	54.8
0.31	60.4	59.2	49.0	46.8	60.4	55.2
0.33	60.0	59.2	47.0	45.7	60.0	54.4
0.35	60.1	59.2	49.4	50.9	60.1	56.0
0.37	60.1	59.1	46.5	47.8	60.1	54.7
0.39	59.7	59.1	46.2	48.4	59.7	54.6
0.41	59.8	60.0	47.3	46.5	60.0	54.7
0.43	59.4	59.7	47.3	46.1	59.7	54.4
0.45	60.0	59.5	47.4	47.0	60.0	54.8
0.47	60.3	59.2	46.0	46.2	60.3	54.4
0.49	59.7	59.0	46.7	46.5	59.7	54.3
0.51	59.8	59.0	44.5	47.3	59.8	54.1
0.53	59.1	59.2	47.3	46.1	59.2	54.2
0.55	59.4	59.2	42.5	45.7	59.4	53.2
0.57	58.8	59.1	46.7	44.9	59.1	53.7
0.59	58.5	59.1	48.1	45.2	59.1	54.0
0.61	56.2	58.2	47.4	44.4	58.2	52.9
0.63	55.9	58.5	48.6	49.3	58.5	54.2
0.65	56.1	58.5	46.1	48.0	58.5	53.4
0.67	54.8	58.1	44.7	46.2	58.1	52.4
0.69	54.6	57.1	48.7	48.7	57.1	53.2
0.71	55.5	56.6	45.4	46.2	56.6	52.1
0.73	55.1	57.1	46.2	44.7	57.1	52.0
0.75	54.2	56.5	48.1	46.4	56.5	52.3
0.77	54.2	56.8	42.9	45.7	56.8	51.3
0.79	54.5	56.9	45.4	49.4	56.9	52.6
0.81	53.5	56.5	48.0	45.4	56.5	52.0
0.83	53.6	56.2	46.0	48.0	56.2	52.0
0.85	53.9	55.2	41.2	43.8	55.2	49.9
0.87	53.3	55.8	42.9	43.1	55.8	50.2
0.89	52.9	52.3	44.2	44.9	52.9	49.5
0.91	52.5	53.6	45.4	43.2	53.6	49.7
0.93	51.9	53.3	35.7	41.2	53.3	47.1
0.95	48.7	50.3	45.2	40.8	50.3	47.1
0.97	49.0	47.8	40.8	34.7	49.0	44.2
0.99	45.8	44.4	36.1	38.9	45.8	42.2

Table II.2: Effect of changing number of neighbors with label propagation model.

Neighbors #	**Accuracy**	**macro-average F1-score**
1	**61.7**	**62.7**
3	56.4	51.2
5	52.5	47.0
7	53.4	46.0
9	54.7	45.3
11	54.6	44.9
13	54.5	45.1
15	55.0	44.3
17	54.7	43.0
19	54.4	42.2
21	54.5	41.8
23	54.2	40.9
25	53.9	40.1
27	53.9	40.1
29	54.0	40.1
31	53.8	39.8
33	54.3	38.8
35	54.1	38.3
37	53.9	37.4
39	53.0	35.3
41	53.0	35.3
43	53.0	35.3
45	53.0	35.3
47	53.0	35.3
49	53.0	35.3
51	52.9	35.0
53	52.5	34.0
55	52.5	34.0

Table II.3: Effect of changing ratio of labeled data when applying label propagation with kNN kernel (*k*=21).

Ratio	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**	**R11**	**R12**	**Max**
0.01	53.61	53.61	50.43	50.14	50.00	49.42	50.00	50.58	48.84	50.43	49.71	50.00	53.61
0.03	53.76	53.61	48.84	51.01	48.99	48.41	50.43	48.12	48.27	49.71	50.00	50.14	53.76
0.05	53.61	53.18	49.57	48.41	50.00	47.98	50.14	48.99	48.55	50.29	45.95	47.69	53.61
0.07	53.76	53.76	49.42	48.27	50.00	50.00	47.98	46.53	49.71	49.42	47.83	50.00	53.76
0.09	53.90	53.47	49.71	50.72	48.70	49.42	48.70	50.58	50.29	50.00	48.84	47.83	53.90
0.11	53.90	53.47	50.29	50.58	49.57	49.57	49.28	50.00	48.70	49.86	49.71	49.28	53.90
0.13	53.76	53.32	49.42	48.84	50.14	49.86	49.71	48.41	50.00	48.84	48.41	49.28	53.76
0.15	53.90	53.76	50.29	50.00	48.55	48.70	48.55	47.40	50.29	48.27	50.00	50.00	53.90
0.17	54.19	53.76	47.83	48.84	50.14	49.13	50.00	49.71	49.71	49.28	50.00	49.71	54.19
0.19	54.19	54.19	48.27	47.54	50.43	47.25	49.13	48.84	49.71	49.57	48.99	47.98	54.19
0.21	54.19	54.05	50.00	49.42	49.28	48.12	50.58	49.42	47.98	49.13	48.12	48.84	54.19
0.23	53.90	54.19	50.00	50.14	49.86	49.71	49.13	48.70	49.42	51.16	50.87	48.55	54.19
0.25	53.76	54.62	51.01	48.55	47.98	51.59	50.87	47.83	49.57	50.29	47.11	49.86	54.62
0.27	54.62	53.90	48.99	50.00	49.42	48.84	49.57	47.54	48.27	48.84	48.99	49.57	54.62
0.29	53.90	53.76	47.69	49.13	47.25	48.55	48.41	48.70	48.27	46.68	50.00	49.57	53.90
0.31	53.90	54.34	49.71	48.84	48.99	48.99	47.69	47.11	49.42	50.14	49.86	50.29	54.34
0.33	53.61	54.34	49.42	47.98	49.13	48.27	49.28	50.14	46.53	46.53	50.58	49.71	54.34
0.35	53.61	54.34	50.14	47.83	50.00	49.86	50.58	49.13	50.72	49.13	50.43	48.70	54.34
0.37	53.47	54.77	48.84	48.99	50.14	50.00	51.45	47.54	49.57	47.54	47.83	48.27	54.77
0.39	52.89	54.48	47.83	46.24	49.57	48.12	48.41	47.98	50.43	48.84	49.13	49.71	54.48
0.41	52.89	54.91	51.45	49.13	49.86	48.70	49.13	47.54	51.01	46.82	49.13	47.40	54.91
0.43	53.47	54.62	49.13	50.00	48.12	46.53	48.84	51.01	46.68	47.69	47.11	51.16	54.62
0.45	53.18	54.62	48.99	50.00	46.82	44.65	45.38	47.25	49.28	46.82	48.41	48.41	54.62
0.47	53.03	53.90	47.98	46.82	45.66	50.14	48.12	50.00	45.66	48.99	48.84	47.11	53.90
0.49	53.90	53.61	50.00	45.66	49.86	45.66	49.42	50.43	47.98	47.11	49.28	48.12	53.90
0.51	53.90	53.76	47.98	46.39	47.25	48.12	47.40	50.14	46.53	50.58	46.82	46.68	53.90
0.53	53.90	53.90	47.40	45.23	44.65	50.00	49.42	48.41	48.41	46.68	49.57	45.81	53.90
0.55	53.90	54.19	48.84	48.84	47.40	47.40	48.99	47.40	48.55	49.71	50.72	47.54	54.19
0.57	53.90	53.76	49.42	48.41	47.54	47.40	43.06	47.54	46.68	46.68	47.69	49.13	53.90
0.59	53.90	54.05	47.69	49.57	45.81	48.70	47.40	47.40	46.82	47.69	47.54	44.94	54.05
0.61	51.30	53.61	48.27	48.70	49.86	48.84	45.81	49.71	47.11	43.79	46.82	46.68	53.61
0.63	51.01	53.32	48.99	44.51	47.83	50.58	49.13	48.99	48.99	46.97	46.82	47.40	53.32
Max	**54.62**	**54.91**	**51.45**	**51.01**	**50.43**	**51.59**	**51.45**	**51.01**	**51.01**	**51.16**	**50.87**	**51.16**	**54.91**
Ratio	**0.27**	**0.41**	**0.41**	**0.03**	**0.19**	**0.25**	**0.37**	**0.43**	**0.41**	**0.23**	**0.23**	**0.43**	**0.41**

Table II.4: Effect of changing ratio of labeled data when applying label propagation with kNN kernel (*k*=1).

Ratio	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**	**R11**	**R12**	**Max**
0.01	56.36	56.07	39.45	39.60	35.98	37.57	35.98	37.57	35.98	37.57	35.98	37.57	56.36
0.03	55.64	55.64	36.85	34.83	35.69	37.72	35.69	37.72	35.69	37.72	35.69	37.72	55.64
0.05	55.49	54.62	36.42	40.46	36.13	37.43	36.13	37.43	36.13	37.43	36.13	37.43	55.49
0.07	55.49	53.90	36.71	35.98	39.74	40.03	39.74	40.03	39.74	40.03	39.74	40.03	55.49
0.09	55.78	53.18	35.40	37.86	40.46	36.85	40.46	36.85	40.46	36.85	40.46	36.85	55.78
0.11	55.64	51.88	34.83	33.96	37.72	36.27	37.72	36.27	37.72	36.27	37.72	36.27	55.64
0.13	54.77	50.58	31.94	37.14	33.82	38.15	33.82	38.15	33.82	38.15	33.82	38.15	54.77
0.15	53.76	50.29	37.28	36.99	38.44	35.69	38.44	35.69	38.44	35.69	38.44	35.69	53.76
0.17	53.03	49.28	36.99	34.97	37.72	33.96	37.72	33.96	37.72	33.96	37.72	33.96	53.03
0.19	52.89	48.99	36.42	35.40	34.54	34.97	34.54	34.97	34.54	34.97	34.54	34.97	52.89
0.21	52.02	48.84	35.55	33.82	37.14	34.25	37.14	34.25	37.14	34.25	37.14	34.25	52.02
0.23	51.88	48.12	31.65	35.55	34.97	32.51	34.97	32.51	34.97	32.51	34.97	32.51	51.88
0.25	51.45	47.69	32.80	33.96	35.98	34.39	35.98	34.39	35.98	34.39	35.98	34.39	51.45
0.27	50.87	46.97	33.53	34.97	33.53	34.68	33.53	34.68	33.53	34.68	33.53	34.68	50.87
0.29	50.72	47.25	30.64	33.38	33.53	36.27	33.53	36.27	33.53	36.27	33.53	36.27	50.72
0.31	50.14	46.82	31.94	32.51	33.53	34.68	33.53	34.68	33.53	34.68	33.53	34.68	50.14
0.33	48.99	46.24	32.66	33.24	33.38	36.71	33.38	36.71	33.38	36.71	33.38	36.71	48.99
0.35	47.98	45.81	35.40	31.94	29.62	32.95	29.62	32.95	29.62	32.95	29.62	32.95	47.98
0.37	47.11	44.94	33.82	33.96	30.92	36.13	30.92	36.13	30.92	36.13	30.92	36.13	47.11
0.39	46.39	44.51	31.36	31.79	31.79	36.42	31.79	36.42	31.79	36.42	31.79	36.42	46.39
0.41	44.51	43.79	35.12	32.08	31.36	32.37	31.36	32.37	31.36	32.37	31.36	32.37	44.51
0.43	43.35	42.92	33.96	31.65	31.07	33.82	31.07	33.82	31.07	33.82	31.07	33.82	43.35
0.45	42.77	41.91	31.21	31.36	32.23	29.48	32.23	29.48	32.23	29.48	32.23	29.48	42.77
0.47	41.47	41.47	31.79	31.07	29.91	29.48	29.91	29.48	29.91	29.48	29.91	29.48	41.47
0.49	41.33	40.46	29.34	32.66	35.12	32.08	35.12	32.08	35.12	32.08	35.12	32.08	41.33
0.51	41.04	40.17	28.61	30.64	30.64	28.61	30.64	28.61	30.64	28.61	30.64	28.61	41.04
0.53	40.03	40.03	28.76	29.62	29.19	29.34	29.19	29.34	29.19	29.34	29.19	29.34	40.03
0.55	39.60	38.73	31.36	29.62	31.36	32.37	31.36	32.37	31.36	32.37	31.36	32.37	39.60
0.57	38.58	38.15	29.34	32.51	29.34	30.20	29.34	30.20	29.34	30.20	29.34	30.20	38.58
0.59	37.72	36.99	31.07	27.02	31.79	31.50	31.79	31.50	31.79	31.50	31.79	31.50	37.72
0.61	36.99	35.12	28.18	28.18	29.34	26.59	29.34	26.59	29.34	26.59	29.34	26.59	36.99
0.63	36.56	34.39	29.62	31.07	30.78	29.91	30.78	29.91	30.78	29.91	30.78	29.91	36.56
Max	**56.36**	**56.07**	**39.45**	**40.46**	**40.46**	**40.03**	**40.46**	**40.03**	**40.46**	**40.03**	**40.46**	**40.03**	**56.36**
Ratio	**0.01**	**0.01**	**0.01**	**0.05**	**0.09**	**0.07**	**0.09**	**0.07**	**0.09**	**0.07**	**0.09**	**0.07**	**0.01**

Table II.5: Effect of changing Gamma when applying label propagation with RBF kernel.

Gam	Accu	F1	Gam	Accu	F1	Gam	Accu	F1
0.01	50.1	22.6	0.85	58.2	55.4	1.69	57.8	54.8
0.03	52.2	35.1	0.87	58.2	55.4	1.71	57.8	54.8
0.05	56.1	46.6	0.89	58.2	55.4	1.73	57.8	54.8
0.07	59.0	52.4	0.91	58.2	55.3	1.75	57.8	54.8
0.09	59.0	53.5	0.93	58.1	55.2	1.77	57.8	54.8
0.11	58.8	53.9	0.95	57.9	55.0	1.79	57.8	54.8
0.13	**59.0**	**54.2**	0.97	57.9	55.0	1.81	57.8	54.8
0.15	58.1	53.7	0.99	57.9	55.0	1.83	57.8	54.8
0.17	57.8	53.6	1.01	57.9	55.0	1.85	57.8	54.8
0.19	57.5	53.5	1.03	57.9	55.0	1.87	57.8	54.8
0.21	57.4	53.5	1.05	57.9	55.0	1.89	57.8	54.8
0.23	57.5	53.8	1.07	57.9	55.0	1.91	57.8	54.8
0.25	57.7	53.9	1.09	57.9	55.0	1.93	57.8	54.8
0.27	57.8	54.3	1.11	57.9	55.0	1.95	57.8	54.8
0.29	57.5	54.1	1.13	57.9	55.0	1.97	57.8	54.8
0.31	57.2	53.9	1.15	57.9	55.0	1.99	57.8	54.8
0.33	57.7	54.4	1.17	57.9	55.0	2.01	57.8	54.8
0.35	57.7	54.4	1.19	57.9	55.0	2.03	57.8	54.8
0.37	57.9	54.9	1.21	57.9	55.0	2.05	57.8	54.8
0.39	58.2	55.2	1.23	57.9	55.0	2.07	57.8	54.8
0.41	58.2	55.2	1.25	57.9	55.0	2.09	57.8	54.8
0.43	58.2	55.2	1.27	57.8	54.8	2.11	57.8	54.8
0.45	58.1	55.1	1.29	57.9	55.0	2.13	57.8	54.8
0.47	58.2	55.2	1.31	57.9	55.0	2.15	57.8	54.8
0.49	58.4	55.4	1.33	57.9	55.0	2.17	57.8	54.8
0.51	58.4	55.4	1.35	57.9	55.0	2.19	57.8	54.8
0.53	58.4	55.4	1.37	57.9	55.0	2.21	57.8	54.8
0.55	58.4	55.4	1.39	57.9	55.0	>=2.23	25.0	13.3
0.57	58.4	55.4	1.41	57.9	55.0			
0.59	58.4	55.4	1.43	57.9	55.0			
0.61	58.2	55.3	1.45	57.9	55.0			
0.63	58.2	55.3	1.47	57.9	55.0			
0.65	58.2	55.3	1.49	57.9	55.0			
0.67	58.2	55.3	1.51	57.9	55.0			
0.69	58.1	55.2	1.53	57.9	55.0			
0.71	58.1	55.2	1.55	57.9	55.0			
0.73	58.1	55.2	1.57	57.9	55.0			
0.75	58.1	55.2	1.59	57.9	55.0			
0.77	58.1	55.2	1.61	57.9	55.0			
0.79	58.1	55.2	1.63	57.8	54.8			
0.81	57.9	55.1	1.65	57.8	54.8			
0.83	58.2	55.4	1.67	57.8	54.8			

Table II.6: Effect of changing ratio of labeled data when applying label propagation with RBF kernel (Gamma=0.07).

Ratio	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**	**R11**	**R12**	**Max**
0.01	60.55	60.26	49.28	50.00	45.81	47.25	46.97	48.55	50.87	47.83	46.39	49.42	60.55
0.03	60.12	59.68	48.12	49.71	47.98	45.38	47.83	47.69	47.40	50.58	45.52	48.70	60.12
0.05	60.55	59.54	48.99	47.98	48.41	47.83	45.95	48.12	46.82	48.70	46.82	46.97	60.55
0.07	60.84	59.25	47.83	47.54	45.66	47.98	46.39	45.81	46.24	49.57	46.97	48.84	60.84
0.09	60.84	58.96	47.40	48.41	48.84	46.82	47.83	45.95	46.39	47.25	47.25	47.54	60.84
0.11	60.84	59.54	45.95	45.95	48.84	48.27	47.54	46.10	47.83	47.40	48.84	47.25	60.84
0.13	60.84	59.10	49.42	47.11	48.84	48.12	48.41	47.11	47.11	46.82	48.41	48.84	60.84
0.15	60.55	58.67	48.27	46.97	46.10	47.83	48.55	46.24	48.27	46.82	47.54	48.84	60.55
0.17	60.26	58.67	48.41	47.98	48.41	47.98	49.42	47.25	47.69	46.68	46.68	47.40	60.26
0.19	59.54	59.10	46.10	48.41	48.41	50.58	47.11	49.42	50.00	45.66	49.13	48.41	59.54
0.21	59.83	58.67	46.82	45.66	46.68	49.28	46.10	48.99	46.68	46.10	46.39	45.66	59.83
0.23	59.97	58.82	47.11	49.13	47.69	46.97	48.84	49.13	47.69	47.54	48.55	45.95	59.97
0.25	59.83	59.10	47.83	48.27	46.97	48.55	46.10	45.23	46.68	49.57	44.51	46.53	59.83
0.27	60.26	59.10	46.24	46.97	47.83	44.51	47.98	49.13	47.11	47.69	49.13	46.10	60.26
0.29	60.26	59.25	47.98	46.10	47.83	43.93	44.80	50.58	48.12	45.95	47.11	48.27	60.26
0.31	60.40	59.25	48.99	46.82	48.27	47.25	47.54	48.27	46.68	48.41	47.54	46.68	60.40
0.33	59.97	59.25	46.97	45.66	48.84	47.11	45.66	48.27	48.70	45.66	49.13	46.97	59.97
0.35	60.12	59.25	49.42	50.87	47.11	48.12	45.66	47.54	45.81	49.13	46.53	46.82	60.12
0.37	60.12	59.10	46.53	47.83	48.12	48.41	44.65	47.11	47.40	47.83	48.84	48.41	60.12
0.39	59.68	59.10	46.24	48.41	46.82	48.84	48.41	47.69	48.55	47.40	48.84	48.70	59.68
0.41	59.83	59.97	47.25	46.53	45.66	45.81	48.41	47.98	47.40	46.97	46.97	48.70	59.97
0.43	59.39	59.68	47.25	46.10	49.86	45.52	47.69	45.81	49.71	48.84	46.82	46.10	59.68
0.45	59.97	59.54	47.40	46.97	47.83	46.24	48.12	48.55	48.84	46.82	48.70	43.93	59.97
0.47	60.26	59.25	45.95	46.24	48.27	47.69	46.10	48.12	47.54	47.11	46.39	50.14	60.26
0.49	59.68	58.96	46.68	46.53	46.68	47.25	48.41	46.24	47.40	47.69	46.82	47.40	59.68
0.51	59.83	58.96	44.51	47.25	45.66	47.40	46.53	47.40	46.39	48.70	46.82	44.94	59.83
0.53	59.10	59.25	47.25	46.10	46.10	48.27	49.42	49.13	46.82	47.40	44.94	44.94	59.25
0.55	59.39	59.25	42.49	45.66	48.84	46.97	44.65	47.40	43.35	45.81	48.99	43.93	59.39
0.57	58.82	59.10	46.68	44.94	47.54	48.55	46.39	48.12	50.14	47.25	45.52	44.36	59.10
0.59	58.53	59.10	48.12	45.23	45.23	45.66	45.38	46.10	45.23	45.09	48.70	44.94	59.10
0.61	56.21	58.24	46.53	47.11	45.52	47.69	48.12	45.38	44.08	44.80	44.94	47.69	58.24
0.63	55.92	58.53	45.66	46.53	48.55	44.65	43.50	46.97	44.80	47.54	44.51	46.10	58.53
Max	**60.84**	**60.26**	**49.42**	**50.87**	**49.86**	**50.58**	**49.42**	**50.58**	**50.87**	**50.58**	**49.13**	**50.14**	**60.84**
Ratio	**0.07**	**0.01**	**0.13**	**0.35**	**0.43**	**0.19**	**0.17**	**0.29**	**0.01**	**0.03**	**0.19**	**0.47**	**0.07**

Table II.7: Effect of changing number of neighbors with label propagation model.

Neighbors #	**Accuracy**	**macro-average F1-score**
1	40.3	38.3
3	53.8	49.6
5	55.5	50.8
7	**58.1**	**53.1**
9	56.1	49.9
11	57.4	51.4
13	55.8	48.9
15	56.6	49.1
17	55.9	47.4
19	56.1	47.2
21	56.2	47.0
23	56.2	46.7
25	56.5	47.0
27	55.8	45.5
29	56.1	45.9
31	55.6	45.0
33	56.2	45.9
35	56.1	45.7
37	55.5	44.6
39	55.2	44.1
41	54.5	42.6
43	54.2	42.1
45	54.0	41.5
47	53.9	41.1
49	53.9	41.1
51	53.9	41.1
53	53.9	40.9
55	54.0	41.0

Table II.8: Effect of changing ratio of labeled data when applying label spreading with kNN kernel (*k*=7).

Ratio	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**	**R11**	**R12**	**Max**
0.01	57.23	57.80	48.12	43.50	44.36	45.09	46.82	50.00	47.11	46.68	47.98	43.79	57.80
0.03	57.95	57.08	47.98	45.52	45.95	49.42	47.54	44.80	44.94	46.68	47.69	45.09	57.95
0.05	57.51	57.37	47.69	48.41	46.82	45.09	45.81	46.53	48.70	46.24	44.22	44.94	57.51
0.07	57.66	57.23	47.25	45.23	49.57	47.25	47.25	45.38	43.06	50.00	45.23	47.98	57.66
0.09	57.51	57.37	41.76	47.69	47.83	43.21	45.81	49.13	47.40	48.27	49.42	44.08	57.51
0.11	58.09	57.51	49.71	48.70	44.94	45.81	46.53	43.35	45.66	46.97	48.27	47.40	58.09
0.13	57.80	57.95	47.25	46.82	46.24	46.68	45.38	44.65	45.66	42.63	43.93	47.40	57.95
0.15	57.95	58.38	48.84	45.81	42.20	44.36	45.95	45.81	44.22	45.38	47.83	45.52	58.38
0.17	58.38	58.67	47.11	45.09	46.53	46.82	46.68	46.68	45.09	44.65	45.38	46.53	58.67
0.19	59.10	58.53	43.64	48.99	45.66	45.81	45.38	47.98	44.80	46.24	45.38	46.39	59.10
0.21	59.10	58.24	44.94	48.41	46.68	45.09	46.97	46.53	45.23	41.76	44.80	47.40	59.10
0.23	59.25	58.09	42.34	45.81	45.52	45.23	44.51	44.65	46.53	46.97	46.24	48.12	59.25
0.25	59.54	58.67	43.64	42.92	41.62	48.41	41.62	44.22	39.74	45.23	42.49	44.94	59.54
0.27	59.83	58.38	45.23	44.08	41.62	44.80	48.99	45.95	47.25	43.93	45.23	43.35	59.83
0.29	59.25	58.96	42.34	43.79	45.81	42.63	43.93	46.24	44.80	48.41	45.66	44.94	59.25
0.31	59.10	58.38	44.22	43.35	45.52	44.51	44.22	45.81	46.39	42.34	38.15	44.22	59.10
0.33	58.67	58.38	46.53	43.06	47.98	44.22	41.91	46.10	46.97	40.32	44.65	46.82	58.67
0.35	58.24	58.38	48.70	45.38	44.80	45.23	40.32	47.54	44.80	44.80	46.68	42.77	58.38
0.37	58.82	58.09	43.06	44.65	44.65	44.65	45.38	48.70	48.41	45.23	50.00	46.39	58.82
0.39	57.95	57.51	42.92	40.90	45.95	44.51	42.34	46.24	45.52	46.82	47.83	45.23	57.95
0.41	58.53	58.96	44.94	44.22	44.22	45.52	46.68	45.38	40.75	43.79	43.21	45.66	58.96
0.43	58.96	58.67	43.79	42.77	43.93	46.82	42.34	40.17	41.62	45.23	44.80	43.35	58.96
0.45	58.53	57.80	48.12	42.34	42.63	43.06	41.76	45.52	40.46	42.20	42.92	40.03	58.53
0.47	58.09	59.10	47.83	44.51	42.34	42.34	39.74	40.90	39.16	41.33	41.18	43.64	59.10
0.49	57.95	58.53	43.79	42.34	40.17	42.63	42.77	46.82	44.51	39.60	41.76	40.46	58.53
0.51	57.80	58.09	46.24	46.10	40.46	45.23	44.94	45.09	45.38	41.33	43.21	39.88	58.09
0.53	57.51	56.94	44.08	38.58	44.80	44.51	42.05	41.62	40.17	42.92	44.94	46.10	57.51
0.55	57.80	57.66	43.64	43.93	44.65	41.33	40.32	40.46	39.02	44.22	43.21	40.32	57.80
0.57	57.80	56.79	40.90	44.65	41.18	41.47	41.18	40.61	40.17	46.39	43.93	42.05	57.80
0.59	57.08	56.94	41.76	43.79	42.63	43.93	42.49	43.93	44.94	42.77	38.58	43.06	57.08
0.61	55.92	55.64	39.88	41.33	43.93	43.21	43.35	41.62	43.64	43.50	41.91	44.51	55.92
0.63	55.35	56.94	39.16	43.93	44.36	40.32	40.17	43.93	42.05	42.92	41.18	42.63	56.94
Max	**59.83**	**59.10**	**49.71**	**48.99**	**49.57**	**49.42**	**48.99**	**50.00**	**48.70**	**50.00**	**50.00**	**48.12**	**59.83**
Ratio	**0.27**	**0.47**	**0.11**	**0.19**	**0.07**	**0.03**	**0.27**	**0.01**	**0.05**	**0.07**	**0.37**	**0.23**	**0.27**

Table II.9: Effect of changing Gamma when applying label spreading with RBF kernel.

Gam	Acc	F1	Gam	Acc	F1	Gam	Acc	F1
0.01	50.1	22.6	0.99	56.5	53.4	1.97	55.6	52.7
0.03	56.2	44.3	1.01	56.6	53.6	1.99	55.6	52.7
0.05	56.5	47.6	1.03	56.6	53.6	2.01	55.6	52.7
0.07	58.7	51.9	1.05	56.6	53.6	2.03	55.6	52.7
0.09	58.1	51.8	1.07	56.9	54.0	2.05	55.6	52.7
0.11	59.0	53.1	1.09	56.9	54.0	2.07	55.6	52.7
0.13	58.8	53.6	1.11	56.8	53.9	2.09	55.5	52.5
0.15	59.2	54.1	1.13	56.8	53.9	2.11	55.5	52.5
0.17	59.2	54.4	1.15	56.8	53.9	2.13	55.5	52.5
0.19	**59.2**	**54.7**	1.17	56.8	53.9	2.15	55.5	52.5
0.21	59.2	54.7	1.19	56.8	53.9	2.17	55.5	52.5
0.23	58.7	54.2	1.21	56.6	53.8	2.19	55.3	52.3
0.25	57.9	53.5	1.23	56.6	53.8	2.21	55.3	52.3
0.27	57.5	53.2	1.25	56.6	53.8	2.23	55.3	52.3
0.29	57.8	53.7	1.27	56.4	53.5	2.25	55.3	52.3
0.31	57.8	53.9	1.29	56.2	53.4	2.27	55.3	52.3
0.33	57.9	54.3	1.31	56.2	53.4	2.29	55.3	52.3
0.35	58.1	54.6	1.33	56.2	53.4	2.31	55.3	52.3
0.37	57.9	54.5	1.35	56.2	53.4	2.33	55.3	52.3
0.39	58.1	54.7	1.37	56.1	53.2	2.35	55.3	52.3
0.41	57.8	54.4	1.39	56.1	53.2	2.37	55.3	52.3
0.43	57.7	54.2	1.41	55.9	53.1	2.39	55.3	52.3
0.45	57.2	53.8	1.43	55.9	53.1	2.41	55.3	52.3
0.47	57.5	54.3	1.45	55.9	53.1	2.43	55.3	52.3
0.49	57.5	54.3	1.47	55.9	53.1	2.45	55.3	52.3
0.51	56.9	53.9	1.49	55.9	53.1	2.47	55.3	52.3
0.53	56.9	53.8	1.51	55.9	53.1	2.49	55.3	52.3
0.55	56.9	53.8	1.53	55.9	53.1	2.51	55.3	52.3
0.57	56.8	53.7	1.55	56.1	53.2	2.53	55.3	52.3
0.59	56.9	53.8	1.57	56.1	53.2	>=2.55	25.0	13.3
0.61	56.9	53.8	1.59	55.9	53.0			
0.63	56.6	53.4	1.61	55.9	53.0			
0.65	56.6	53.4	1.63	55.8	52.8			
0.67	56.4	53.1	1.65	55.8	52.8			
0.69	56.4	53.1	1.67	55.9	52.9			
0.71	56.4	53.1	1.69	55.9	52.9			
0.73	56.4	53.1	1.71	55.9	52.9			
0.75	56.2	53.0	1.73	55.8	52.8			
0.77	56.2	53.0	1.75	55.8	52.8			
0.79	56.2	53.0	1.77	55.8	52.8			
0.81	56.2	53.0	1.79	55.8	52.8			
0.83	56.4	53.2	1.81	55.6	52.7			
0.85	56.4	53.2	1.83	55.6	52.7			
0.87	56.2	53.0	1.85	55.6	52.7			
0.89	56.2	53.0	1.87	55.6	52.7			
0.91	56.1	52.9	1.89	55.6	52.7			
0.93	56.1	52.9	1.91	55.6	52.7			
0.95	55.9	52.8	1.93	55.6	52.7			
0.97	56.2	53.1	1.95	55.6	52.7			

Table II.10: Effect of changing ratio of labeled data when applying label spreading with RBF kernel (Gamma=0.19).

Ratio	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**	**R11**	**R12**	**Max**
0.01	59.97	60.69	47.25	43.06	44.65	45.23	42.77	46.24	45.23	47.11	44.51	42.63	60.69
0.03	59.97	61.13	46.39	45.81	45.52	48.41	43.64	44.65	45.09	44.80	43.64	43.64	61.13
0.05	59.83	61.42	45.95	44.94	46.97	44.08	46.53	42.34	44.36	43.21	43.93	41.62	61.42
0.07	59.83	60.98	44.65	44.22	45.52	46.24	43.93	44.36	46.24	46.68	42.49	42.77	60.98
0.09	60.55	60.84	42.34	47.69	45.52	42.92	44.08	45.09	45.81	42.63	42.49	45.52	60.84
0.11	59.97	60.98	43.79	40.17	46.24	44.51	42.63	43.64	42.63	44.94	41.91	43.79	60.98
0.13	58.67	60.55	42.92	41.47	45.38	46.24	41.91	40.46	42.34	42.34	44.36	41.62	60.55
0.15	59.83	60.84	47.25	43.64	42.49	42.20	45.66	42.92	42.77	45.09	44.08	42.49	60.84
0.17	60.55	60.26	43.64	40.03	45.09	43.50	43.93	42.63	43.06	43.50	43.50	39.45	60.55
0.19	60.40	60.26	44.65	46.53	44.36	43.35	46.24	39.31	44.94	43.64	45.38	44.94	60.40
0.21	60.98	60.84	42.34	45.95	42.49	43.35	43.21	44.65	45.38	42.05	42.05	42.20	60.98
0.23	60.40	60.55	42.20	44.08	44.08	40.32	43.79	44.94	45.81	41.62	37.72	44.36	60.55
0.25	60.40	60.26	47.98	43.06	43.64	43.93	46.10	42.34	41.76	44.94	41.33	44.36	60.40
0.27	60.12	59.97	44.08	40.32	41.76	45.23	43.64	43.06	45.38	44.08	43.06	41.04	60.12
0.29	60.84	59.68	43.35	44.08	43.64	44.36	39.88	41.04	41.04	43.79	43.21	44.22	60.84
0.31	60.69	59.25	39.45	42.77	44.80	41.33	44.51	39.74	41.76	44.94	43.50	42.05	60.69
0.33	60.69	59.10	41.76	40.46	41.33	42.77	42.05	41.04	45.52	43.21	38.73	38.58	60.69
0.35	61.42	58.96	38.44	43.64	42.49	44.80	40.46	43.64	42.05	42.34	40.17	45.52	61.42
0.37	61.42	58.82	44.80	42.05	42.05	42.20	41.76	42.77	44.36	43.35	42.05	43.64	61.42
0.39	60.69	58.53	40.61	40.61	41.62	37.14	42.34	44.36	43.64	41.33	43.64	43.50	60.69
0.41	61.13	57.95	45.66	39.02	41.76	45.23	42.63	39.45	41.47	39.74	43.06	39.45	61.13
0.43	61.27	57.95	42.49	40.46	41.18	41.47	38.01	41.33	43.64	42.49	40.61	39.31	61.27
0.45	60.84	58.53	39.31	42.20	42.20	39.74	42.49	43.79	41.18	44.36	41.04	39.31	60.84
0.47	60.26	59.39	41.47	41.76	36.71	40.46	41.62	41.47	41.91	39.60	40.61	42.92	60.26
0.49	59.68	59.25	40.75	42.49	40.17	41.33	43.21	37.72	42.92	41.62	42.20	37.14	59.68
0.51	60.12	59.25	38.58	40.90	42.49	42.05	41.47	40.46	38.87	44.08	40.17	43.79	60.12
0.53	59.39	58.82	40.61	42.63	39.16	38.29	43.21	41.47	42.05	40.61	40.75	42.05	59.39
0.55	59.68	60.40	40.32	41.18	42.05	38.01	36.56	40.03	40.03	42.49	42.20	39.74	60.40
0.57	59.25	60.26	40.61	40.90	44.51	41.33	43.93	43.93	40.61	40.03	43.64	41.62	60.26
0.59	59.10	59.68	38.44	39.45	40.75	39.88	40.75	41.91	41.18	39.74	45.66	43.93	59.68
0.61	56.65	59.10	42.20	43.50	40.17	41.33	36.99	40.17	36.42	38.87	41.91	36.99	59.10
0.63	56.50	59.10	43.50	39.60	40.61	39.45	42.20	39.88	42.77	38.44	40.32	38.73	59.10
Max	**61.42**	**61.42**	**47.98**	**47.69**	**46.97**	**48.41**	**46.53**	**46.24**	**46.24**	**47.11**	**45.66**	**45.52**	**61.42**
Ratio	**0.35**	**0.05**	**0.25**	**0.09**	**0.05**	**0.03**	**0.05**	**0.01**	**0.07**	**0.01**	**0.59**	**0.09**	**0.05**

Table II.11: Effect of changing ratio of labeled data when applying semi-supervised K-means.

Ratio	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	Max
0.01	26.88	29.62	29.34	45.66	32.23	44.22	44.08	38.58	42.20	40.90	41.91	28.32	45.66
0.03	42.05	42.05	45.52	44.22	31.94	42.63	43.64	41.62	45.52	29.77	25.29	46.39	46.39
0.05	41.91	43.64	41.62	40.17	40.61	41.91	41.47	43.64	43.93	44.08	46.39	43.79	46.39
0.07	42.20	41.76	46.53	45.52	30.20	43.64	44.08	31.36	41.47	45.38	45.52	41.47	46.53
0.09	41.91	41.76	38.87	41.91	46.53	43.64	45.23	46.24	41.47	41.91	46.68	41.91	46.68
0.11	41.91	43.93	42.34	46.10	41.76	41.76	43.06	45.81	46.53	41.62	45.81	46.53	46.53
0.13	41.62	43.79	41.91	42.05	41.91	43.50	43.79	46.53	46.68	45.23	45.81	46.53	46.68
0.15	46.39	45.38	41.47	41.62	42.49	41.76	46.39	43.79	45.66	41.91	43.79	41.76	46.39
0.17	41.62	45.38	40.46	46.39	45.66	46.24	45.95	46.39	41.91	45.66	46.10	41.62	46.39
0.19	43.93	45.38	41.91	32.37	41.04	45.38	46.53	31.36	45.52	42.05	46.10	42.05	46.53
0.21	46.24	45.38	37.72	45.38	46.68	45.81	46.24	41.76	43.50	41.76	41.91	46.68	46.68
0.23	46.10	45.52	46.53	46.68	41.76	46.39	41.47	41.76	46.24	41.62	42.20	46.24	46.68
0.25	46.10	45.38	46.39	43.50	41.62	43.93	46.39	46.24	43.93	41.62	41.62	45.66	46.39
0.27	43.93	46.10	45.81	43.35	46.24	41.04	43.64	46.39	45.81	46.53	45.95	40.90	46.53
0.29	43.79	45.38	41.76	45.52	41.62	45.95	41.62	46.10	46.24	46.53	45.95	45.81	46.53
0.31	44.08	45.09	46.24	45.38	46.24	46.68	46.24	41.76	46.39	43.93	46.10	46.24	46.68
0.33	44.08	45.52	41.47	41.76	46.24	45.81	46.68	41.62	41.62	41.76	45.23	45.23	46.68
0.35	46.10	45.66	42.77	46.39	46.39	45.09	44.36	46.53	45.38	41.62	44.80	45.66	46.53
0.37	46.24	45.81	41.18	46.39	46.53	41.47	45.38	45.52	45.66	46.24	46.53	46.82	46.82
0.39	46.24	45.81	45.52	46.53	46.24	41.62	41.62	45.95	46.39	46.39	46.39	43.50	46.53
0.41	46.10	45.81	46.24	42.20	41.47	45.81	44.94	41.33	46.82	45.52	45.95	44.80	46.82
0.43	46.10	45.38	46.24	46.39	42.34	46.53	45.52	46.10	46.10	45.23	41.33	41.91	46.53
0.45	46.53	45.81	45.38	46.39	45.81	41.62	45.95	45.81	46.53	45.95	43.64	41.18	46.53
0.47	46.39	45.95	46.24	45.52	44.94	41.62	40.90	46.10	46.53	46.68	45.52	45.23	46.68
0.49	46.39	45.52	41.47	41.76	45.09	46.39	41.76	44.36	41.91	45.52	45.38	46.39	46.39
0.51	46.68	45.38	45.52	43.21	44.65	43.35	44.94	42.20	45.38	46.24	46.68	41.62	46.68
0.53	46.68	45.23	43.79	45.38	45.81	41.62	42.20	46.39	41.47	45.09	45.66	45.81	46.68
0.55	46.53	45.38	41.18	46.53	45.81	41.76	45.52	46.24	41.62	46.24	44.80	41.91	46.53
0.57	46.82	44.94	46.24	46.68	46.39	45.95	45.95	45.38	46.10	46.24	44.94	43.35	46.82
0.59	46.68	45.09	46.53	44.08	45.09	46.53	44.80	45.38	45.95	45.81	41.33	46.68	46.68
0.61	46.68	42.34	45.52	45.81	45.38	44.08	45.66	45.52	41.18	45.38	41.76	46.24	46.68
0.63	46.53	44.65	46.10	46.53	41.91	44.80	43.35	44.80	41.33	45.38	41.33	46.10	46.53
Max	**46.82**	**46.10**	**46.53**	**46.68**	**46.68**	**46.68**	**46.68**	**46.53**	**46.82**	**46.68**	**46.68**	**46.82**	**46.82**
Ratio	**0.57**	**0.27**	**0.07**	**0.23**	**0.21**	**0.31**	**0.33**	**0.13**	**0.41**	**0.47**	**0.09**	**0.37**	**0.37**

Table II.12: Effect of changing threshold δ when applying self-training with distant confidence.

δ	**Acc**	δ	**Acc**	Δ	**Acc**
0.01	69.8	0.34	70.1	0.67	70.1
0.02	69.5	0.35	70.1	0.68	69.9
0.03	69.7	0.36	70.1	0.69	69.9
0.04	69.5	0.37	70.1	0.7	69.8
0.05	69.5	0.38	70.1	0.71	69.9
0.06	69.4	0.39	70.1	0.72	69.9
0.07	69.2	0.4	70.1	0.73	69.9
0.08	69.2	0.41	70.1	0.74	69.9
0.09	69.4	0.42	70.1	0.75	69.9
0.1	69.2	0.43	69.9	0.76	69.7
0.11	69.1	0.44	69.9	0.77	69.7
0.12	69.1	0.45	69.9	0.78	70.1
0.13	69.8	0.46	69.9	0.79	70.1
0.14	69.7	0.47	69.9	0.8	69.9
0.15	69.5	0.48	69.9	**0.81**	**70.2**
0.16	69.5	0.49	69.9	0.82	70.2
0.17	69.7	0.5	69.9	0.83	70.2
0.18	69.7	0.51	69.9	0.84	70.2
0.19	69.7	0.52	69.9	0.85	70.2
0.2	69.5	0.53	69.9	0.86	70.2
0.21	69.7	0.54	69.9	0.87	70.2
0.22	69.4	0.55	69.9	0.88	70.2
0.23	69.5	0.56	69.9	0.89	70.2
0.24	69.5	0.57	69.9	0.9	70.4
0.25	69.4	0.58	70.1	0.91	70.2
0.26	69.4	0.59	70.1	0.92	70.2
0.27	69.5	0.6	70.1	0.93	70.2
0.28	69.7	0.61	70.1	0.94	70.1
0.29	69.9	0.62	69.9	0.95	70.1
0.3	69.9	0.63	69.9	0.96	70.2
0.31	69.9	0.64	69.9	0.97	70.2
0.32	69.9	0.65	69.9	0.98	70.2
0.33	70.1	0.66	69.9	0.99	70.2

Table II.13: Effect of changing ratio of labeled data when applying logistic regression classifier.

Ratio	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**	**R11**	**R12**	**Max**
0.01	46.53	51.01	52.60	51.30	52.02	51.59	52.89	50.14	55.35	52.89	52.89	51.01	55.35
0.03	60.98	57.37	57.66	58.09	55.49	55.20	54.19	56.94	54.48	52.31	54.77	56.21	60.98
0.05	63.15	57.37	60.12	61.42	59.10	61.85	58.24	59.54	58.96	61.99	60.12	59.97	63.15
0.07	65.75	60.12	58.67	60.55	62.28	62.57	61.56	60.40	62.14	62.57	60.26	60.55	65.75
0.09	66.47	62.14	61.99	58.24	63.01	63.01	63.29	63.44	63.87	64.31	64.16	64.16	66.47
0.11	66.47	61.99	63.87	65.03	63.58	67.92	60.55	65.75	65.61	62.57	62.14	63.01	67.92
0.13	66.76	63.29	66.18	63.29	62.86	64.88	63.73	63.01	66.62	63.44	64.74	63.44	66.76
0.15	67.34	64.16	66.76	64.02	65.61	65.17	64.60	66.04	64.88	67.92	65.32	63.58	67.92
0.17	67.77	64.02	65.32	68.06	67.92	65.61	65.46	64.45	66.47	66.62	66.62	65.03	68.06
0.19	67.20	65.90	66.91	66.33	65.46	66.76	66.91	64.88	66.33	67.77	64.74	66.04	67.77
0.21	66.91	65.46	66.33	64.16	64.88	66.33	66.33	64.88	67.05	66.04	66.76	66.04	67.05
0.23	67.63	65.90	65.46	67.20	68.06	68.21	67.05	66.91	67.05	65.03	67.20	67.77	68.21
0.25	67.77	65.90	67.34	65.46	66.76	67.77	68.21	67.63	66.62	67.63	66.76	65.61	68.21
0.27	68.35	65.61	67.92	67.05	66.47	67.05	67.77	68.06	67.77	65.46	67.34	66.33	68.35
0.29	68.21	66.04	66.18	66.33	67.77	67.77	66.62	67.34	66.18	66.62	68.06	67.77	68.21
0.31	69.22	65.75	67.20	67.20	68.64	68.06	66.91	66.91	67.20	68.50	68.64	69.08	69.22
0.33	69.22	66.47	67.49	68.79	67.20	68.64	65.46	68.93	68.79	66.33	68.06	68.35	69.22
0.35	70.09	65.90	68.79	66.18	67.92	68.06	68.79	67.92	67.34	67.92	68.35	67.77	70.09
0.37	69.51	66.47	67.49	68.06	68.79	68.50	66.33	68.35	68.35	67.77	67.63	69.36	69.51
0.39	69.65	65.75	69.08	67.63	68.06	66.33	68.06	69.51	70.23	67.63	68.06	68.64	70.23
0.41	70.23	66.76	68.35	68.50	68.06	68.06	68.93	68.35	66.62	68.21	67.63	67.34	70.23
0.43	70.66	67.63	67.92	67.49	69.08	68.35	68.21	68.64	69.08	68.79	69.80	69.22	70.66
0.45	71.97	67.63	67.49	67.63	69.65	70.52	68.06	67.77	67.63	70.52	68.06	68.79	71.97
0.47	71.39	66.76	67.63	68.21	68.79	68.93	68.93	69.65	67.77	70.09	68.93	68.35	71.39
0.49	71.24	67.05	68.93	67.20	70.09	69.51	68.50	68.50	69.80	68.21	70.52	69.65	71.24
0.51	71.24	67.20	68.21	69.65	68.93	68.79	67.63	69.08	68.93	68.79	69.36	69.08	71.24
0.53	71.10	67.20	68.50	69.51	70.52	68.21	68.64	68.93	69.94	69.94	69.36	67.92	71.10
0.55	71.10	68.06	68.06	68.79	68.79	68.50	69.80	68.64	69.36	69.80	70.81	69.51	71.10
0.57	70.66	67.92	68.93	68.93	69.51	69.08	68.35	69.80	69.51	69.65	69.36	67.92	70.66
0.59	70.81	68.35	68.35	70.38	68.50	69.94	70.52	68.79	69.80	69.22	69.51	69.22	70.81
0.61	70.52	69.08	69.08	70.23	67.63	69.22	69.22	68.50	68.79	69.80	69.22	69.51	70.52
0.63	70.09	68.79	68.06	67.34	69.80	69.22	68.64	68.50	69.80	68.06	69.08	69.94	70.09
Max	**71.97**	**69.08**	**69.08**	**70.38**	**70.52**	**70.52**	**70.52**	**69.80**	**70.23**	**70.52**	**70.81**	**69.94**	**71.97**
Ratio	**0.45**	**0.61**	**0.39**	**0.59**	**0.53**	**0.45**	**0.59**	**0.57**	**0.39**	**0.45**	**0.55**	**0.63**	**0.45**

Table II.14: Effect of changing ratio of labeled data when applying self-training with distant confidence (δ=0.81).

Ratio	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**	**R11**	**R12**	**Max**
0.01	46.10	51.45	51.73	50.72	50.87	51.16	51.45	49.57	53.90	52.46	52.17	50.58	53.90
0.03	59.68	57.08	57.08	58.24	53.76	53.61	52.17	55.06	53.03	51.59	53.76	55.64	59.68
0.05	64.02	57.51	57.80	61.56	55.35	60.40	55.92	57.08	56.65	60.55	60.84	58.67	64.02
0.07	64.74	59.39	57.95	60.84	61.56	59.68	59.39	59.83	60.55	59.97	58.38	59.10	64.74
0.09	65.17	61.56	59.83	56.65	62.72	63.29	62.28	62.43	62.14	63.58	63.58	64.16	65.17
0.11	65.17	61.56	62.43	63.44	62.72	67.77	60.55	65.61	65.46	59.25	60.84	63.87	67.77
0.13	66.18	63.29	64.60	61.42	62.14	62.86	63.44	63.01	65.90	63.44	61.71	62.28	66.18
0.15	65.03	63.29	65.75	62.72	65.46	61.99	64.16	65.61	63.87	65.90	64.60	62.86	65.90
0.17	65.90	65.17	64.31	66.33	66.04	65.17	64.31	62.28	65.75	64.31	65.46	64.60	66.33
0.19	65.90	65.90	67.63	64.88	66.18	67.05	65.75	64.16	66.04	66.47	63.58	64.60	67.63
0.21	65.17	65.61	65.03	64.02	64.60	66.18	65.17	64.88	67.05	64.74	64.02	64.16	67.05
0.23	65.75	65.75	66.47	67.05	67.34	66.62	65.61	68.21	67.05	65.03	67.49	65.46	68.21
0.25	67.20	66.18	65.61	66.62	67.49	66.47	67.63	66.47	65.61	67.34	64.31	64.74	67.63
0.27	66.18	65.75	65.46	66.04	66.18	66.76	66.04	66.33	68.50	65.90	66.62	65.32	68.50
0.29	66.91	65.90	64.74	66.33	66.18	66.47	65.61	65.03	65.32	66.04	67.34	66.91	67.34
0.31	67.92	65.32	67.77	67.20	67.20	66.04	66.18	67.20	67.49	68.79	67.49	68.35	68.79
0.33	68.35	66.04	66.91	68.79	67.77	68.21	65.75	66.76	67.92	66.91	67.20	67.34	68.79
0.35	67.63	66.18	68.50	67.20	67.77	66.33	68.35	67.20	65.90	66.47	67.05	67.49	68.50
0.37	67.77	65.32	67.20	65.61	68.21	67.92	66.76	67.49	68.21	67.34	67.05	68.35	68.35
0.39	67.34	66.18	69.22	67.20	66.76	65.61	67.05	68.06	67.20	67.05	67.77	68.21	69.22
0.41	67.49	66.04	68.21	66.62	67.20	67.49	68.50	66.47	67.20	68.35	67.63	66.18	68.50
0.43	68.21	66.76	67.77	67.92	68.21	66.47	66.62	68.50	66.62	68.35	67.92	69.22	69.22
0.45	68.93	66.18	67.49	67.34	68.64	68.93	67.92	65.61	67.49	68.06	66.18	67.92	68.93
0.47	69.08	67.05	67.05	68.06	68.35	68.21	68.21	68.50	68.35	69.08	70.09	67.05	70.09
0.49	69.22	66.62	68.64	67.05	68.93	69.08	67.34	68.79	68.79	67.77	69.22	68.35	69.22
0.51	69.65	66.62	68.35	69.51	67.49	68.06	67.05	68.64	69.51	68.64	69.51	67.63	69.65
0.53	68.93	66.18	68.21	68.64	69.22	68.64	67.77	68.06	69.08	69.65	68.21	67.34	69.65
0.55	69.36	67.34	67.34	69.36	68.50	69.08	69.08	67.92	69.65	68.06	70.38	68.93	70.38
0.57	69.08	67.05	67.63	68.21	68.21	68.50	67.63	68.50	67.77	68.79	69.22	68.50	69.22
0.59	69.94	67.20	69.51	68.21	67.63	70.81	69.65	69.08	69.08	70.09	68.21	68.93	70.66
0.61	69.94	67.92	68.21	68.79	67.20	68.93	68.06	67.49	68.06	68.93	68.93	68.64	69.94
0.63	69.65	67.63	68.06	68.06	69.51	68.35	68.93	68.79	69.65	68.06	68.06	68.35	69.65
Max	**69.94**	**67.92**	**69.51**	**69.51**	**69.51**	**70.81**	**69.65**	**69.08**	**69.65**	**70.09**	**70.38**	**69.22**	**70.81**
Ratio	**0.59**	**0.61**	**0.59**	**0.51**	**0.63**	**0.59**	**0.59**	**0.59**	**0.55**	**0.59**	**0.55**	**0.43**	**0.59**

Table II.15: Effect of changing probabilistic threshold when applying self-training with probabilistic confidence.

Prop	**Acc of Round 1**	**Acc of Round 2**	**Acc of Round 3**
0.01	68.9	68.9	68.9
0.02	68.9	68.9	68.9
0.03	68.9	68.9	68.9
0.04	68.9	68.9	68.9
0.05	68.9	68.9	68.9
0.06	68.9	68.9	68.9
0.07	68.9	68.9	68.9
0.08	68.9	68.9	68.9
0.09	68.9	68.9	68.9
0.1	68.9	68.9	68.9
0.11	68.9	68.9	68.9
0.12	68.9	68.9	68.9
0.13	68.9	68.9	68.9
0.14	68.9	68.9	68.9
0.15	68.9	68.9	68.9
0.16	68.9	68.9	68.9
0.17	68.9	68.9	68.9
0.18	68.9	68.9	68.9
0.19	68.9	68.9	68.9
0.2	68.9	68.9	68.9
0.21	68.9	68.9	68.9
0.22	68.9	68.9	68.9
0.23	68.9	68.9	68.9
0.24	68.9	68.9	68.9
0.25	68.9	68.9	68.9
0.26	68.9	68.9	68.9
0.27	68.9	68.9	68.9
0.28	68.9	68.9	68.9
0.29	68.9	68.9	68.9
0.3	68.9	68.9	68.9
0.31	68.9	68.9	68.9
0.32	68.9	68.9	68.9
0.33	68.9	68.9	68.9
0.34	69.1	68.9	68.9
0.35	68.8	68.9	68.9
0.36	68.8	68.6	68.6
0.37	68.9	68.6	68.6
0.38	68.6	68.6	68.6
0.39	68.5	68.6	68.6
0.4	68.5	68.6	68.5
0.41	68.6	68.5	68.6
0.42	68.6	68.6	68.6
0.43	68.6	68.6	68.6
0.44	69.1	68.6	68.4
0.45	69.2	68.6	68.6
0.46	69.2	68.6	68.9
0.47	69.1	68.8	68.6
0.48	69.1	67.9	68.5
0.49	69.5	68.5	68.5
0.5	69.1	68.2	67.8

Prop	Acc of Round 1	Acc of Round 2	Acc of Round 3
0.51	69.4	68.5	68.1
0.52	69.4	68.2	67.6
0.53	69.1	68.2	67.8
0.54	69.5	68.5	67.3
0.55	69.8	68.5	67.9
0.56	69.8	69.1	68.2
0.57	69.7	69.4	68.8
0.58	69.9	69.2	68.4
0.59	70.7	68.9	68.5
0.6	70.4	69.1	68.5
0.61	70.5	68.5	68.2
0.62	70.1	69.1	68.4
0.63	70.1	69.1	68.5
0.64	69.5	69.2	68.5
0.65	70.2	69.8	68.2
0.66	69.9	69.8	68.8
0.67	70.1	69.7	68.8
0.68	70.1	68.9	68.8
0.69	69.9	69.2	68.9
0.7	70.2	69.2	68.5
0.71	71.1	69.1	68.6
0.72	70.8	69.5	68.8
0.73	71.2	69.7	68.9
0.74	71.2	70.1	69.5
0.75	71.1	70.2	69.9
0.76	71.0	70.5	70.2
0.77	71.4	71.0	70.4
0.78	71.0	71.1	71.0
0.79	71.2	71.1	70.8
0.8	71.4	71.0	71.1
0.81	71.4	71.4	70.7
0.82	71.4	71.5	71.4
0.83	71.7	71.7	71.5
0.84	71.5	71.2	71.4
0.85	71.7	71.5	71.5
0.86	72.0	71.7	71.5
0.87	72.0	72.0	71.8
0.88	71.7	71.5	71.7
0.89	72.1	72.0	72.0
0.9	**72.1**	**72.1**	**72.1**
0.91	72.0	72.0	72.0
0.92	72.0	72.0	72.0
0.93	72.0	72.0	72.0
0.94	72.0	72.0	72.0
0.95	72.0	72.0	72.0
0.96	72.0	72.0	72.0
0.97	72.0	72.0	72.0
0.98	72.0	72.0	72.0
0.99	72.0	72.0	72.0

Table II.16: Effect of changing ratio of labeled data when applying self-training with probabilistic confidence (Prop=0.9).

Ratio	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	Max
0.01	46.53	51.01	50.43	51.01	55.20	48.55	53.32	51.88	49.13	49.28	52.17	51.01	55.20
0.03	60.98	57.37	54.19	56.50	58.82	56.21	57.95	55.06	58.82	53.76	57.37	55.64	60.98
0.05	63.15	57.37	59.39	59.25	60.69	61.27	57.23	61.56	61.99	61.56	61.56	59.39	63.15
0.07	65.75	60.12	63.15	58.82	61.42	61.99	63.01	60.98	62.43	58.96	61.99	62.28	65.75
0.09	66.47	61.71	62.86	65.03	64.74	62.86	62.86	64.02	62.72	64.16	63.87	60.55	66.47
0.11	66.62	61.85	64.31	64.31	62.43	65.17	63.15	63.15	63.73	65.46	62.43	65.46	66.62
0.13	66.91	63.44	65.03	64.60	65.32	62.86	66.04	65.61	64.60	66.18	63.87	62.86	66.91
0.15	67.20	64.16	67.20	63.87	65.75	63.01	65.17	67.20	63.58	62.43	64.88	66.18	67.20
0.17	67.77	64.02	63.73	66.91	66.47	64.88	65.03	65.46	64.74	64.31	65.03	64.60	67.77
0.19	67.34	65.75	66.47	66.62	66.04	66.62	67.05	65.32	66.33	67.20	67.92	64.16	67.92
0.21	66.91	65.32	67.05	65.90	65.17	65.32	66.33	65.03	65.61	66.18	66.33	66.18	67.05
0.23	67.77	65.90	66.76	67.77	65.90	66.04	64.88	66.76	66.91	68.35	67.34	66.62	68.35
0.25	67.77	65.90	67.20	66.47	68.50	65.90	68.06	67.05	66.76	67.20	68.50	67.92	68.50
0.27	68.35	65.46	66.76	66.62	69.22	67.20	65.46	65.75	68.50	67.49	66.76	67.20	69.22
0.29	68.21	66.04	66.91	67.63	67.49	66.91	66.91	68.79	66.33	68.79	66.33	65.90	68.79
0.31	69.36	65.75	68.06	66.91	67.92	67.34	67.34	66.33	66.62	68.64	65.75	66.62	69.36
0.33	69.08	66.47	69.36	68.06	67.92	68.06	68.35	67.34	68.93	69.08	65.46	69.22	69.36
0.35	69.65	65.75	67.49	67.77	68.35	69.36	68.64	66.04	68.50	67.92	67.49	67.63	69.65
0.37	69.36	66.47	68.79	68.06	67.77	67.05	66.76	67.34	68.79	67.34	68.50	67.77	69.36
0.39	69.65	65.75	69.51	69.08	68.21	68.64	68.35	69.22	68.06	67.63	69.08	68.35	69.65
0.41	70.23	66.76	68.93	69.36	67.49	69.80	68.21	69.22	67.20	69.51	67.77	67.77	70.23
0.43	70.66	67.63	68.21	69.08	70.23	69.36	66.04	67.63	67.63	68.93	69.94	66.76	70.66
0.45	72.11	67.63	68.64	68.35	66.62	68.06	68.79	68.79	68.93	69.36	68.06	69.94	72.11
0.47	71.68	66.62	68.79	68.79	67.77	69.08	68.21	68.06	68.93	68.93	69.22	69.22	71.68
0.49	71.10	67.34	67.05	70.09	68.21	68.64	67.05	69.36	68.35	70.95	70.09	68.50	71.10
0.51	71.39	67.20	67.49	69.80	69.51	67.05	68.93	68.93	68.21	69.65	70.09	67.92	71.39
0.53	71.24	67.20	69.51	69.51	68.93	68.50	69.22	68.79	69.51	67.34	68.93	68.50	71.24
0.55	71.10	68.06	69.51	68.35	68.64	71.10	69.08	69.65	68.64	69.22	70.23	68.93	71.10
0.57	70.66	67.92	69.94	69.51	69.51	68.50	68.64	68.79	68.50	69.22	67.92	69.36	70.66
0.59	70.95	68.50	70.38	68.93	69.51	69.51	68.50	69.08	68.79	69.65	69.22	68.79	70.95
0.61	70.23	69.08	68.64	69.94	68.35	68.93	69.65	69.94	69.51	69.65	69.51	68.50	70.23
0.63	70.09	68.79	69.51	68.79	69.51	70.66	69.80	69.80	71.10	70.09	68.50	68.93	71.10
Max	**72.11**	**69.08**	**70.38**	**70.09**	**70.23**	**71.10**	**69.80**	**69.94**	**71.10**	**70.95**	**70.23**	**69.94**	**72.11**
Ratio	**0.45**	**0.61**	**0.59**	**0.49**	**0.43**	**0.55**	**0.63**	**0.61**	**0.63**	**0.49**	**0.55**	**0.45**	**0.45**

Table II.17: Effect of changing both Lamda and LamdaU when applying QN-S3VM BFGS optimizer for semi-supervised SVM with OvR multiclass strategy.

Lam	**Acc**	**Lam**	**Acc**	**Lam**	**Acc**	**Lam**	**Acc**	**Lam**	**Acc**
0.001	64.60	0.101	69.36	0.201	63.58	0.301	60.26	0.401	59.10
0.005	69.08	0.105	69.51	0.205	64.02	0.305	60.98	0.405	58.96
0.009	70.23	0.109	69.36	0.209	63.29	0.309	60.69	0.409	59.39
0.013	70.52	0.113	69.08	0.213	63.15	0.313	59.68	0.413	59.83
0.017	70.95	0.117	69.80	0.217	63.44	0.317	60.55	0.417	58.53
0.021	70.52	0.121	69.22	0.221	63.15	0.321	61.13	0.421	58.67
0.025	**71.53**	0.125	69.08	0.225	62.72	0.325	60.69	0.425	58.67
0.029	70.38	0.129	68.50	0.229	62.86	0.329	59.97	0.429	58.96
0.033	71.10	0.133	69.22	0.233	61.56	0.333	60.69	0.433	58.82
0.037	70.52	0.137	69.08	0.237	61.27	0.337	59.83	0.437	58.38
0.041	70.52	0.141	69.22	0.241	61.56	0.341	59.39	0.441	59.54
0.045	70.38	0.145	68.06	0.245	62.43	0.345	59.25	0.445	59.68
0.049	70.23	0.149	68.64	0.249	62.57	0.349	60.84	0.449	58.67
0.053	70.95	0.153	68.21	0.253	61.42	0.353	59.83	0.453	59.97
0.057	70.38	0.157	67.77	0.257	61.56	0.357	59.10	0.457	59.39
0.061	70.81	0.161	68.50	0.261	61.99	0.361	60.69	0.461	60.40
0.065	69.80	0.165	68.50	0.265	61.56	0.365	60.98	0.465	57.37
0.069	70.23	0.169	69.08	0.269	61.42	0.369	60.69	0.469	57.66
0.073	70.23	0.173	68.06	0.273	62.28	0.373	59.83	0.473	58.38
0.077	68.79	0.177	68.06	0.277	61.85	0.377	59.10	0.477	59.54
0.081	68.50	0.181	63.87	0.281	61.27	0.381	59.68	0.481	58.09
0.085	69.08	0.185	67.77	0.285	61.42	0.385	59.83	0.485	58.96
0.089	69.08	0.189	63.15	0.289	61.71	0.389	59.68	0.489	56.65
0.093	68.64	0.193	63.29	0.293	61.42	0.393	58.82	0.493	58.09
0.097	69.08	0.197	63.58	0.297	60.69	0.397	59.68	0.497	58.67

Table II.18: Effect of changing LamdaU when applying QN-S3VM BFGS optimizer for semi-supervised SVM (Lamda=0.025) with OvR multiclass strategy.

LamdaU	Acc
0.001	70.81
0.005	**71.53**
0.009	**71.53**
0.013	71.39
0.017	71.24
0.021	70.95
0.025	**71.53**
0.029	70.09
0.033	70.66
0.037	70.52
0.041	71.10
0.045	70.52
0.049	69.94
0.053	71.24
0.057	70.95
0.061	71.10
0.065	70.23
0.069	70.95
0.073	70.09
0.077	69.51
0.081	70.23
0.085	69.94
0.089	70.09
0.093	69.65
0.097	70.09
0.101	70.23
0.105	70.95

Table II.19: Effect of changing ratio of labeled data when applying QN-S3VM BFGS optimizer for semi-supervised SVM (Lamda=0.025) with OvR multiclass strategy.

Ratio	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	Max
0.01	42.63	53.18	50.43	44.36	42.63	43.21	43.06	43.06	42.49	43.64	43.06	42.92	53.18
0.03	57.23	54.19	54.34	57.23	58.24	57.95	58.24	58.53	58.53	58.24	58.09	59.10	59.10
0.05	60.26	51.73	60.12	55.92	60.26	60.84	59.83	60.55	60.40	59.25	59.54	60.26	60.84
0.07	61.42	57.37	58.24	55.78	61.71	61.27	61.56	61.27	60.98	60.84	60.55	60.84	61.71
0.09	64.45	56.50	59.10	59.54	64.74	65.17	64.88	64.74	65.17	65.32	64.88	65.46	65.46
0.11	63.87	58.82	61.56	59.39	64.02	64.02	63.73	63.15	63.73	63.15	63.44	64.31	64.31
0.13	62.72	61.99	64.60	63.58	62.86	62.72	63.87	63.73	62.72	63.15	63.87	63.15	64.60
0.15	65.03	61.99	62.14	63.58	64.74	65.03	65.90	65.61	65.75	65.03	65.75	64.74	65.90
0.17	64.60	62.28	64.60	66.62	65.90	64.31	65.61	64.45	64.88	64.88	64.74	64.74	66.62
0.19	67.92	61.85	63.44	63.29	67.34	67.92	68.21	67.49	67.49	66.76	67.92	67.34	68.21
0.21	65.90	63.01	63.15	65.32	66.33	66.91	66.33	67.20	66.47	66.76	66.33	66.62	67.20
0.23	66.04	64.02	63.29	64.02	66.18	66.76	66.33	66.18	66.18	66.62	66.47	66.33	66.76
0.25	67.63	64.88	65.75	65.46	67.34	67.92	67.20	66.47	67.49	67.05	67.20	67.05	67.92
0.27	66.04	63.15	64.88	66.18	67.49	66.62	66.76	67.20	67.34	67.63	67.77	67.63	67.77
0.29	66.18	64.88	66.18	68.50	66.33	67.05	67.05	67.34	67.77	67.05	67.34	67.05	68.50
0.31	66.76	64.31	64.74	66.04	66.76	67.49	67.92	67.20	66.91	66.91	67.34	67.63	67.92
0.33	67.05	65.46	67.05	66.47	67.77	67.05	67.63	67.49	67.77	67.77	67.05	66.33	67.77
0.35	68.64	65.32	67.63	67.34	68.35	68.50	69.65	68.50	68.64	69.22	67.92	69.51	69.65
0.37	68.21	64.60	67.77	68.35	68.93	68.50	69.36	67.92	68.93	68.06	68.50	68.64	69.36
0.39	68.21	65.17	67.92	67.34	67.77	68.93	68.79	68.79	68.35	68.50	68.50	68.79	68.93
0.41	68.35	65.46	68.06	65.46	68.50	69.36	69.36	68.21	68.50	69.08	69.51	69.08	69.51
0.43	69.65	66.47	65.90	67.05	69.22	69.51	68.64	70.38	69.08	68.50	68.79	69.22	70.38
0.45	69.65	66.33	67.63	66.47	68.93	69.51	69.65	69.65	69.08	69.94	69.65	69.51	69.94
0.47	69.65	65.90	65.75	67.05	69.80	70.09	69.51	69.65	69.65	68.79	69.36	69.65	70.09
0.49	70.23	66.62	67.49	68.50	69.94	70.81	70.95	70.09	70.38	69.36	69.94	70.81	70.95
0.51	70.52	67.05	68.50	66.91	70.81	70.23	71.53	71.10	70.09	71.53	70.23	70.38	71.53
0.53	70.52	66.18	67.49	67.49	70.81	70.81	70.66	70.23	71.10	71.24	69.51	70.23	71.24
0.55	70.38	67.77	65.46	67.77	70.66	70.38	69.51	69.22	70.95	70.23	70.09	70.23	70.95
0.57	71.10	67.20	66.62	67.05	69.51	69.22	70.52	70.38	69.94	69.08	69.94	70.66	71.10
0.59	69.80	65.75	68.50	69.80	70.09	70.52	70.09	69.65	69.08	70.95	70.23	70.09	70.95
0.61	70.09	67.63	68.93	68.64	70.81	69.94	70.66	70.52	70.52	70.38	70.81	70.66	70.81
0.63	70.23	66.91	69.94	68.50	71.82	70.23	70.52	70.66	71.39	70.81	71.39	70.52	71.82
Max	**71.10**	**67.77**	**69.94**	**69.80**	**71.82**	**70.81**	**71.53**	**71.10**	**71.39**	**71.53**	**71.39**	**70.81**	**71.82**
Ratio	**0.57**	**0.55**	**0.63**	**0.59**	**0.63**	**0.49**	**0.51**	**0.51**	**0.63**	**0.51**	**0.63**	**0.49**	**0.63**

Table II.20: Effect of changing both Lamda and LamdaU when applying QN-S3VM BFGS optimizer for semi-supervised SVM with OvO multiclass strategy.

Accuracy	**Lamda**	**Accuracy**	**Lamda**
64.7	0.001	67.1	0.157
68.4	0.005	66.3	0.161
68.5	0.009	65.9	0.165
68.6	0.013	66.5	0.169
69.4	0.017	65.6	0.173
69.7	0.021	65.0	0.177
69.7	0.025	65.6	0.181
68.6	0.029	65.8	0.185
69.4	0.033	65.9	0.189
69.7	0.037	65.3	0.193
69.4	0.041	66.0	0.197
70.1	**0.045**	65.6	0.201
69.2	0.049	65.3	0.205
69.8	0.053	65.9	0.209
69.4	0.057	65.0	0.213
69.5	0.061	64.6	0.217
68.9	0.065	64.6	0.221
69.1	0.069	62.9	0.225
69.8	0.073	63.6	0.229
69.4	0.077	64.2	0.233
69.2	0.081	62.7	0.237
69.2	0.085	62.0	0.241
68.6	0.089	63.2	0.245
68.9	0.093	63.6	0.249
68.4	0.097	64.2	0.253
69.2	0.101	64.0	0.257
68.6	0.105	64.0	0.261
68.5	0.109		
68.8	0.113		
67.5	0.117		
68.1	0.121		
67.9	0.125		
67.9	0.129		
67.5	0.133		
67.9	0.137		
67.5	0.141		
66.3	0.145		
67.2	0.149		
66.3	0.153		

Table II.21: Effect of changing ratio of labeled data when applying QN-S3VM BFGS optimizer for semi-supervised SVM (Lamda=0.045) with OvO multiclass strategy.

Ratio	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**	**R11**	**R12**	**Max**
0.01	40.46	52.60	48.27	45.23	40.61	52.46	41.47	52.46	41.18	52.17	39.31	52.75	52.75
0.03	57.80	53.90	58.82	49.86	58.96	54.05	57.51	55.06	58.53	53.61	58.53	54.34	58.96
0.05	61.99	55.06	59.54	56.94	61.71	54.77	61.85	54.77	61.99	55.06	61.42	54.77	61.99
0.07	62.57	58.53	60.40	60.55	62.57	58.96	63.58	58.09	63.58	58.96	61.85	58.09	63.58
0.09	65.61	58.96	61.56	60.55	65.03	58.96	64.88	57.95	65.17	58.53	65.17	58.38	65.61
0.11	64.31	60.84	64.74	63.58	65.03	60.84	63.87	60.55	64.45	61.99	64.16	60.69	65.03
0.13	65.61	62.43	63.58	64.60	65.17	62.72	64.45	63.87	65.75	62.86	64.60	63.44	65.75
0.15	64.60	63.58	61.71	65.46	64.60	61.99	64.74	62.43	64.60	61.42	64.74	63.15	65.46
0.17	65.75	62.28	65.46	65.61	64.45	64.02	65.03	63.44	65.32	63.01	64.45	63.44	65.75
0.19	64.45	63.44	65.17	65.75	66.04	64.02	65.46	63.29	64.88	63.87	65.32	63.87	66.04
0.21	66.33	64.31	65.75	64.45	65.61	65.46	65.61	65.75	65.90	64.31	65.46	65.46	66.33
0.23	65.03	64.60	65.61	64.74	65.75	65.17	65.46	65.32	65.75	65.61	66.18	65.32	66.18
0.25	65.75	65.75	69.36	64.60	65.32	65.75	65.61	65.90	65.46	66.47	66.18	65.90	69.36
0.27	66.18	65.32	65.32	65.03	65.90	65.46	66.18	65.75	66.62	66.18	67.05	66.04	67.05
0.29	65.75	66.04	67.49	66.62	66.18	65.90	66.04	66.18	66.62	64.88	66.33	66.76	67.49
0.31	66.33	65.75	68.35	67.34	66.47	66.33	65.75	66.33	66.47	65.75	66.47	64.88	68.35
0.33	68.06	66.04	68.50	67.20	66.91	66.33	67.20	65.75	66.91	65.61	67.49	65.61	68.50
0.35	69.36	66.18	66.33	66.33	68.21	65.03	67.49	66.04	68.79	66.04	68.21	66.62	69.36
0.37	69.22	65.90	65.75	68.06	69.08	65.61	68.64	64.74	69.22	66.04	69.08	65.46	69.22
0.39	68.35	66.91	68.35	66.76	69.51	66.33	68.64	66.62	68.21	66.04	69.80	65.46	69.80
0.41	69.36	68.21	69.36	66.62	68.21	67.49	69.94	67.05	69.08	67.34	68.93	67.49	69.94
0.43	68.93	66.76	69.51	67.05	69.80	66.04	68.93	67.20	69.22	66.62	69.36	66.62	69.80
0.45	69.94	67.05	67.05	67.63	69.80	66.76	69.65	66.04	70.09	66.62	69.65	67.34	70.09
0.47	69.51	67.05	68.64	67.92	68.79	66.76	69.51	66.62	68.93	66.76	68.93	67.34	69.51
0.49	70.09	65.90	67.92	68.50	69.51	67.20	70.23	67.63	70.23	67.20	69.51	67.49	70.23
0.51	69.22	66.33	67.77	67.77	69.22	66.62	69.22	66.62	69.51	67.05	69.51	66.91	69.51
0.53	69.51	66.76	68.06	68.21	69.08	66.62	68.93	66.62	69.51	66.91	68.79	66.62	69.51
0.55	68.79	67.05	67.05	68.35	68.93	66.76	68.93	66.76	69.65	65.75	68.79	66.47	69.65
0.57	69.36	66.76	68.50	68.64	68.93	66.47	68.64	67.34	68.93	67.49	68.50	66.33	69.36
0.59	69.36	66.76	67.63	67.77	69.51	67.34	69.94	67.05	68.93	67.34	68.64	66.47	69.94
0.61	70.52	67.49	68.35	69.08	69.94	67.20	69.80	67.20	69.51	66.91	70.38	67.63	70.52
0.63	69.51	67.20	68.21	69.08	69.94	67.34	70.52	66.62	70.23	66.62	69.94	67.63	70.52
Max	70.52	68.21	69.51	69.08	69.94	67.49	70.52	67.63	70.23	67.49	70.38	67.63	70.52
Ratio	0.61	0.41	0.43	0.61	0.61	0.41	0.63	0.49	0.49	0.57	0.61	0.61	0.61

Table II.22: Distribution of data points when clustering testing data by using Birch algorithm.

Data points of class negative in cluster negative	146
Data points of class neutral in cluster negative	312
Data points of class positive in cluster negative	150
Data points of class negative in cluster neutral	1
Data points of class neutral in cluster neutral	0
Data points of class positive in cluster neutral	0
Data points of class negative in cluster positive	26
Data points of class neutral in cluster positive	34
Data points of class positive in cluster positive	23

Table II.23: Distribution of data points when clustering training data by using Birch Algorithm.

Data points of class negative in cluster negative	339
Data points of class neutral in cluster negative	623
Data points of class positive in cluster negative	348
Data points of class negative in cluster neutral	191
Data points of class neutral in cluster neutral	455
Data points of class positive in cluster neutral	280
Data points of class negative in cluster positive	1030
Data points of class neutral in cluster positive	2049
Data points of class positive in cluster positive	933

Table II.24: Distribution of data points when clustering testing data after clustering training data with Birch.

Data points of class negative in cluster negative	34
Data points of class neutral in cluster negative	74
Data points of class positive in cluster negative	28
Data points of class negative in cluster neutral	30
Data points of class neutral in cluster neutral	42
Data points of class positive in cluster neutral	25
Data points of class negative in cluster positive	109
Data points of class neutral in cluster positive	230
Data points of class positive in cluster positive	120

Table II.25: Clustering testing data by using K-means algorithm while centroids initialized by using K-means++.

Run #	**Accuracy**	**macro-average F1-score**
1	27.5	23.5
2	27.6	23.6
3	22.5	18.8
4	27.7	23.8
5	22.1	18.1
6	22.4	18.6
7	22.3	18.2
8	22.5	18.8
9	27.3	23.5
10	27.9	23.8
11	22.0	18.0
MAX	**27.9**	**23.8**
MIN	**22.0**	**18.0**
AVRG	**24.7**	**20.8**
CI High	**26.6**	**22.6**
CI Low	**22.8**	**18.9**

Table II.26: Distribution of data points when clustering testing data by using K-means while centroids initialized by using K-means++.

Data points of class negative in cluster negative	54
Data points of class neutral in cluster negative	87
Data points of class positive in cluster negative	37
Data points of class negative in cluster neutral	1
Data points of class neutral in cluster neutral	0
Data points of class positive in cluster neutral	0
Data points of class negative in cluster positive	118
Data points of class neutral in cluster positive	259
Data points of class positive in cluster positive	136

Table II.27: Average distribution of data points when clustering testing data by using K-means while centroids initialized by using K-means++.

Data points of class negative in cluster negative	98	75	74	97	74	75	74	98	75	97
Data points of class neutral in cluster negative	226	124	120	224	120	121	120	225	122	223
Data points of class positive in cluster negative	114	63	59	111	59	60	59	114	62	111
Data points of class negative in cluster neutral	1	1	1	1	1	1	1	1	1	1
Data points of class neutral in cluster neutral	0	0	0	0	0	0	0	0	0	0
Data points of class positive in cluster neutral	0	0	0	0	0	0	0	0	0	0
Data points of class negative in cluster positive	74	97	98	75	98	97	98	74	97	75
Data points of class neutral in cluster positive	120	222	226	122	226	225	226	121	224	123
Data points of class positive in cluster positive	59	110	114	62	114	113	114	59	111	62

Table II.28: Clustering testing data by using K-means algorithm while centroids initialized randomly.

Run #	**Accuracy**	**macro-average F1-score**
1	42.8	27.8
2	27.7	23.8
3	29.9	22.8
4	27.6	23.7
5	32.5	24.9
6	44.9	30.4
7	42.9	27.5
8	27.6	23.5
9	32.8	25.0
10	27.6	23.5
MAX	**44.9**	**30.4**
MIN	**27.6**	**22.8**
AVRG	**33.6**	**25.3**
CI High	**38.7**	**27.1**
CI Low	**28.5**	**23.5**

Table II.29: Distribution of data points when clustering testing data by using K-means while centroids initialized randomly.

Data points of class negative in cluster negative	1
Data points of class neutral in cluster negative	0
Data points of class positive in cluster negative	0
Data points of class negative in cluster neutral	97
Data points of class neutral in cluster neutral	222
Data points of class positive in cluster neutral	110
Data points of class negative in cluster positive	75
Data points of class neutral in cluster positive	124
Data points of class positive in cluster positive	63

Table II.30: Average distribution of data points when clustering testing data by using K-means while centroids initialized randomly.

Data points of class negative in cluster negative	1	75	97	75	1	75	1	74	1	74
Data points of class neutral in cluster negative	0	123	222	123	0	124	0	120	0	120
Data points of class positive in cluster negative	0	63	110	62	0	63	0	59	0	59
Data points of class negative in cluster neutral	97	1	75	1	75	97	98	1	75	1
Data points of class neutral in cluster neutral	222	0	124	0	124	222	226	0	121	0
Data points of class positive in cluster neutral	110	0	63	0	63	110	114	0	59	0
Data points of class negative in cluster positive	75	97	1	97	97	1	74	98	97	98
Data points of class neutral in cluster positive	124	223	0	223	222	0	120	226	225	226
Data points of class positive in cluster positive	63	110	0	111	110	0	59	114	114	114

Table II.31: Distribution of data points when clustering testing data randomly initialized from classes by using K-means.

Data points of class negative in cluster negative	56
Data points of class neutral in cluster negative	91
Data points of class positive in cluster negative	40
Data points of class negative in cluster neutral	116
Data points of class neutral in cluster neutral	255
Data points of class positive in cluster neutral	133
Data points of class negative in cluster positive	1
Data points of class neutral in cluster positive	0
Data points of class positive in cluster positive	0

Table II.32: Average distribution of data points when clustering testing data initialized randomly from classes by using K-means.

Data points of class negative in cluster negative	167	97	75	97	165	98	169	75	164	75
Data points of class neutral in cluster negative	328	222	124	222	322	228	340	124	329	124
Data points of class positive in cluster negative	160	110	63	110	154	115	169	63	166	63
Data points of class negative in cluster neutral	1	1	1	75	1	74	1	97	1	1
Data points of class neutral in cluster neutral	0	0	0	124	0	118	0	222	0	0
Data points of class positive in cluster neutral	0	0	0	63	0	58	0	110	0	0
Data points of class negative in cluster positive	5	75	97	1	7	1	3	1	8	97
Data points of class neutral in cluster positive	18	124	222	0	24	0	6	0	17	222
Data points of class positive in cluster positive	13	63	110	0	19	0	4	0	7	110

Table II.33: Experiment results of initialize each centroid from its corresponding class by using K-means algorithm.

Run #	**Accuracy**	**macro-average F1-score**	**Run #**	**Accuracy**	**macro-average F1-score**
1	44.9	30.4	20	26.6	16.2
2	24.0	13.9	21	27.6	23.6
3	42.8	27.8	22	29.8	22.7
4	24.7	13.6	23	27.5	23.3
5	32.5	24.9	24	42.3	27.1
6	22.5	18.8	25	49.1	23.3
7	24.9	14.5	26	49.3	23.2
8	50.3	23.2	27	49.4	24.2
9	45.1	30.2	28	49.9	24.6
10	27.3	23.5	29	51.2	25.4
11	25.6	15.2	30	29.9	22.8
12	24.6	14.6	31	26.7	23.7
13	50.4	25.4	32	50.7	24.2
14	42.6	27.1	33	49.7	23.0
15	26.0	17.6	**MAX**	**51.2**	**30.4**
16	50.1	22.6	**MIN**	**22.4**	**13.6**
17	48.7	23.2	**AVRG**	**36.8**	**22.0**
18	22.4	18.2	**CI High**	**40.8**	**23.7**
19	24.4	14.3	**CI Low**	**32.8**	**20.3**

Table II.34: Distribution of data points when clustering training data randomly initialized from classes by using K-means.

Data points of class negative in cluster negative	191
Data points of class neutral in cluster negative	454
Data points of class positive in cluster negative	278
Data points of class negative in cluster neutral	966
Data points of class neutral in cluster neutral	1991
Data points of class positive in cluster neutral	979
Data points of class negative in cluster positive	403
Data points of class neutral in cluster positive	682
Data points of class positive in cluster positive	304

Table II.35: Clustering training data when initializing each centroid randomly by using K-means algorithm.

Run #	**Accuracy**	**macro-average F1-score**	**Run #**	**Accuracy**	**macro-average F1-score**
1	41.5	33.9	20	30.8	29.2
2	30.5	29.0	21	28.7	27.6
3	42.8	35.4	22	42.9	34.9
4	29.4	28.5	23	28.1	27.9
5	39.8	30.8	24	26.9	25.6
6	30.9	29.0	25	27.6	26.1
7	42.8	35.4	26	32.2	32.5
8	32.7	30.7	27	36.9	30.6
9	40.4	30.9	28	41.6	28.9
10	29.8	27.0	29	31.9	30.5
11	30.6	25.5	30	42.7	39.0
12	30.8	29.2	31	29.6	27.1
13	40.3	30.9	32	31.2	25.4
14	43.4	36.2	33	29.9	27.0
15	29.6	27.1	**MAX**	**43.9**	**39.3**
16	28.3	27.5	**MIN**	**26.9**	**25.4**
17	43.9	39.3	**AVRG**	**34.4**	**30.1**
18	27.2	25.5	**CI High**	**36.5**	**31.5**
19	39.8	30.8	**CI Low**	**32.3**	**28.8**

Table II.36: Distribution of data points when clustering training data randomly initialized from classes by using K-means.

Data points of class negative in cluster negative	445
Data points of class neutral in cluster negative	790
Data points of class positive in cluster negative	363
Data points of class negative in cluster neutral	1058
Data points of class neutral in cluster neutral	2288
Data points of class positive in cluster neutral	1112
Data points of class negative in cluster positive	57
Data points of class neutral in cluster positive	49
Data points of class positive in cluster positive	86

Table II.37: Clustering training data when initializing each centroid randomly from its corresponding class by using K-means algorithm.

Run #	**Accuracy**	**macro-average F1-score**	**Run #**	**Accuracy**	**macro-average F1-score**
1	44.4	38.6	20	37.1	32.7
2	49.6	24.7	21	48.4	24.3
3	33.9	32.8	22	45.8	33.3
4	41.6	33.5	23	36.9	30.5
5	47.5	31.9	24	44.8	36.2
6	45.3	37.8	25	43.2	31.7
7	41.3	28.7	26	36.9	30.6
8	35.1	32.1	27	49.2	27.4
9	51.7	27.3	28	45.1	32.8
10	43.2	31.3	29	44.2	30.0
11	45.1	32.8	30	42.8	39.3
12	50.7	24.9	31	38.9	32.9
13	44.5	31.1	32	45.1	32.8
14	38.7	34.2	33	42.8	35.4
15	43.1	33.0	**MAX**	**51.7**	**39.3**
16	37.8	33.8	**MIN**	**31.3**	**24.3**
17	45.3	37.3	**AVRG**	**42.9**	**32.1**
18	31.3	28.7	**CI High**	**44.6**	**33.4**
19	42.9	35.4	**CI Low**	**41.1**	**30.8**

Table II.38: Applying K-means firstly on training data (and initialize centroids from training data) and then use the resulted centroids to classify testing data by assigning each sample to the closest centroid by using Braycurtis distance measure.

Run #	**Accuracy**	**macro-average F1-score**	**Run #**	**Accuracy**	**macro-average F1-score**
1	32.9	31.8	20	33.5	26.3
2	32.1	27.5	21	40.2	35.5
3	31.4	30.2	22	23.7	15.2
4	35.0	28.3	23	28.2	27.8
5	42.9	35.8	24	41.2	36.4
6	33.1	32.0	25	48.0	34.7
7	49.3	25.7	26	37.9	37.3
8	34.5	27.5	27	42.1	30.8
9	30.1	29.0	28	46.2	23.0
10	26.4	25.8	29	49.1	27.3
11	38.4	30.5	30	43.5	26.2
12	40.0	30.2	31	49.4	26.1
13	44.2	35.4	32	36.8	30.0
14	32.8	31.7	33	41.0	30.9
15	40.9	31.0	**MAX**	**49.4**	**40.5**
16	23.4	21.3	**MIN**	**23.4**	**15.2**
17	26.0	18.0	**AVRG**	**37.3**	**29.5**
18	34.1	33.2	**CI High**	**40.0**	**31.4**
19	43.9	40.5	**CI Low**	**34.7**	**27.6**

Table II.39: Applying K-means firstly on training data (and initialize centroids from training data) and then use the resulted centroids to classify testing data by assigning each sample to the closest centroid by using Canberra distance measure.

Run #	Accuracy	macro-average F1-score	Run #	Accuracy	macro-average F1-score
1	50.1	25.1	20	34.8	32.3
2	34.1	27.7	21	41.2	31.9
3	33.1	25.7	22	27.6	17.9
4	28.9	20.8	23	41.3	33.7
5	28.5	27.1	24	32.9	30.6
6	32.4	25.3	25	37.3	35.4
7	37.7	34.7	26	41.6	32.6
8	38.9	35.1	27	23.7	22.0
9	39.6	36.1	28	40.3	30.4
10	47.3	22.8	29	39.2	30.8
11	40.9	30.0	30	40.8	33.7
12	33.4	26.9	31	34.0	26.4
13	49.6	24.5	32	25.3	23.3
14	39.0	29.9	33	40.2	31.3
15	27.3	25.6	**MAX**	**50.1**	**36.1**
16	34.7	29.3	**MIN**	**23.7**	**17.9**
17	41.0	31.3	**AVRG**	**37.1**	**28.8**
18	40.0	36.0	**CI High**	**39.4**	**30.5**
19	47.1	23.7	**CI Low**	**34.7**	**27.1**

Table II.40: Applying K-means firstly on training data (and initialize centroids from training data) and then use the resulted centroids to classify testing data by assigning each sample to the closest centroid by using Chebyshev distance measure.

Run #	Accuracy	macro-average F1-score	Run #	Accuracy	macro-average F1-score
1	35.0	33.6	20	32.5	29.7
2	38.0	36.8	21	33.5	33.0
3	47.7	35.4	22	32.1	27.5
4	31.4	29.5	23	42.8	35.1
5	49.7	31.5	24	49.1	26.6
6	43.8	32.1	25	43.2	37.7
7	34.1	32.2	26	38.3	31.2
8	37.3	28.7	27	45.1	34.1
9	42.6	35.0	28	47.0	29.5
10	37.1	33.6	29	41.3	30.8
11	30.6	30.4	30	41.9	29.4
12	46.2	27.2	31	43.5	31.0
13	42.1	27.7	32	44.9	24.9
14	43.9	32.2	33	41.8	28.7
15	40.6	32.9	**MAX**	**49.7**	**37.7**
16	48.0	33.5	**MIN**	**29.3**	**24.9**
17	41.0	30.8	**AVRG**	**40.7**	**31.1**
18	48.8	25.7	**CI High**	**42.8**	**32.2**
19	29.3	28.3	**CI Low**	**38.7**	**30.0**

Table II.41: Applying K-means firstly on training data (and initialize centroids from training data) and then use the resulted centroids to classify testing data by assigning each sample to the closest centroid by using City Block (Manhattan) distance measure.

Run #	**Accuracy**	**macro-average F1-score**	**Run #**	**Accuracy**	**macro-average F1-score**
1	45.2	33.8	20	38.2	34.6
2	31.4	28.8	21	42.5	31.2
3	28.5	21.2	22	46.8	28.1
4	50.1	25.8	23	37.9	30.7
5	32.4	31.3	24	48.8	33.0
6	30.6	25.9	25	43.4	35.6
7	30.1	26.0	26	40.9	28.2
8	44.5	27.2	27	41.0	30.0
9	29.0	28.4	28	47.3	30.1
10	32.1	25.2	29	45.4	41.8
11	37.6	30.5	30	49.3	28.0
12	38.0	28.6	31	47.0	22.1
13	41.9	34.8	32	40.5	32.8
14	27.7	25.5	33	44.9	32.3
15	25.7	24.0	**MAX**	**50.1**	**41.8**
16	41.3	28.3	**MIN**	**25.7**	**21.2**
17	41.5	34.8	**AVRG**	**39.7**	**29.5**
18	47.8	26.5	**CI High**	**42.2**	**31.0**
19	41.6	29.6	**CI Low**	**37.2**	**28.0**

Table II.42: Applying K-means firstly on training data (and initialize centroids from training data) and then use the resulted centroids to classify testing data by assigning each sample to the closest centroid by using Correlation distance measure.

Run #	**Accuracy**	**macro-average F1-score**	**Run #**	**Accuracy**	**macro-average F1-score**
1	40.6	28.9	20	47.5	30.7
2	41.9	29.2	21	41.6	30.6
3	28.0	25.3	22	48.0	28.2
4	46.5	37.1	23	42.1	30.7
5	48.8	27.0	24	42.9	32.6
6	42.3	34.8	25	49.4	23.5
7	30.1	24.2	26	40.8	32.8
8	44.8	31.9	27	43.5	32.6
9	29.5	28.5	28	48.6	31.5
10	30.8	24.4	29	42.2	35.1
11	24.7	22.7	30	46.0	35.3
12	27.6	18.5	31	36.3	29.6
13	45.1	33.1	32	33.4	27.3
14	27.7	27.5	33	35.5	34.7
15	36.1	29.4	**MAX**	**49.4**	**37.1**
16	40.2	30.1	**MIN**	**24.7**	**18.5**
17	42.5	29.4	**AVRG**	**39.8**	**29.8**
18	43.6	32.7	**CI High**	**42.3**	**31.3**
19	44.4	34.6	**CI Low**	**37.3**	**28.3**

Table II.43: Applying K-means firstly on training data (and initialize centroids from training data) and then use the resulted centroids to classify testing data by assigning each sample to the closest centroid by using Cosine distance measure.

Run #	**Accuracy**	**macro-average F1-score**	**Run #**	**Accuracy**	**macro-average F1-score**
1	42.3	30.7	20	39.9	34.7
2	36.3	29.5	21	50.7	24.7
3	31.8	26.3	22	23.7	21.3
4	44.2	31.9	23	28.2	24.0
5	31.9	29.4	24	29.3	27.3
6	45.1	36.6	25	44.4	30.8
7	42.6	35.2	26	26.3	17.6
8	43.5	38.5	27	48.3	25.8
9	49.3	30.0	28	47.1	22.5
10	39.6	28.6	29	41.8	34.5
11	37.9	31.4	30	48.8	32.9
12	34.7	32.2	31	40.2	35.0
13	40.0	34.8	32	45.5	41.5
14	43.1	34.4	33	44.9	32.1
15	41.2	29.0	**MAX**	**50.7**	**41.5**
16	35.4	34.6	**MIN**	**23.7**	**17.6**
17	42.9	25.9	**AVRG**	**39.7**	**30.4**
18	35.8	27.9	**CI High**	**42.1**	**32.3**
19	33.1	32.7	**CI Low**	**37.2**	**28.6**

Table II.44: Applying K-means firstly on training data (and initialize centroids from training data) and then use the resulted centroids to classify testing data by assigning each sample to the closest centroid by using Euclidean distance measure.

Run #	**Accuracy**	**macro-average F1-score**	**Run #**	**Accuracy**	**macro-average F1-score**
1	41.8	28.9	20	43.5	30.2
2	30.1	24.2	21	38.3	27.3
3	43.6	36.4	22	49.9	27.3
4	41.2	33.5	23	39.6	32.4
5	50.3	28.3	24	42.5	39.5
6	31.1	26.9	25	44.8	32.4
7	39.5	27.9	26	46.8	29.0
8	30.8	29.5	27	48.1	31.9
9	30.6	27.7	28	31.5	31.6
10	44.9	32.5	29	48.8	25.0
11	42.9	29.7	30	34.1	27.6
12	46.1	39.2	31	32.8	27.2
13	50.7	29.3	32	40.2	34.0
14	40.0	27.7	33	41.6	30.1
15	44.7	31.3	**MAX**	**50.7**	**39.5**
16	49.6	29.3	**MIN**	**30.1**	**23.9**
17	47.7	23.9	**AVRG**	**41.7**	**30.1**
18	44.2	30.8	**CI High**	**44.0**	**31.4**
19	44.5	31.3	**CI Low**	**39.5**	**28.8**

Table II.45: Applying K-means firstly on training data (and initialize centroids from training data) and then use the resulted centroids to classify testing data by assigning each sample to the closest centroid by using Squared Euclidean distance measure.

Run #	**Accuracy**	**macro-average F1-score**	**Run #**	**Accuracy**	**macro-average F1-score**
1	31.5	28.7	20	47.3	24.7
2	36.8	31.8	21	42.1	30.1
3	44.5	31.3	22	47.8	22.8
4	50.1	24.1	23	30.3	23.7
5	41.8	33.9	24	32.2	25.2
6	41.6	30.1	25	42.5	29.1
7	40.5	34.2	26	41.2	32.4
8	42.9	29.7	27	44.7	32.4
9	44.4	32.8	28	33.7	27.1
10	37.0	29.6	29	49.9	26.7
11	45.2	34.3	30	46.0	26.3
12	41.5	28.7	31	40.0	27.7
13	44.9	32.5	32	44.1	31.4
14	40.2	34.1	33	45.1	32.8
15	46.4	26.1	**MAX**	**50.1**	**34.3**
16	32.7	31.9	**MIN**	**30.3**	**22.8**
17	37.3	29.7	**AVRG**	**41.2**	**29.6**
18	41.9	29.8	**CI High**	**43.1**	**30.7**
19	31.8	30.1	**CI Low**	**39.3**	**28.4**

Table II.46: Effect of changing ratio of labeled data when applying our proposed solution.

Ratio	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**	**R11**	**R12**	**Max**
0.01	46.53	50.87	51.88	51.88	52.31	51.59	49.86	50.58	51.16	55.20	48.84	51.45	55.20
0.03	60.84	57.23	56.36	55.92	57.51	58.24	56.07	56.50	56.94	56.79	60.26	58.24	60.84
0.05	63.15	57.23	59.54	59.68	60.12	57.80	61.13	57.66	61.71	54.77	61.27	62.28	63.15
0.07	65.75	59.97	60.84	61.13	62.43	63.01	61.99	62.72	61.13	62.57	62.28	63.29	65.75
0.09	66.33	61.56	62.86	62.86	61.85	60.55	62.14	64.31	62.72	64.74	62.43	62.14	66.33
0.11	66.62	61.71	64.74	61.13	65.46	65.46	63.29	63.01	64.45	63.44	64.31	63.44	66.62
0.13	66.91	63.29	64.16	64.74	63.87	64.74	64.02	64.31	64.45	64.16	64.02	64.31	66.91
0.15	67.05	64.16	64.60	65.75	63.44	65.90	63.87	65.90	63.87	65.46	66.18	65.17	67.05
0.17	67.63	64.16	64.45	65.17	65.17	66.18	64.88	67.20	66.18	63.44	64.74	66.04	67.63
0.19	67.34	66.04	65.32	66.47	65.32	66.76	63.15	65.90	66.18	65.61	66.47	65.61	67.34
0.21	66.76	65.32	66.62	66.33	66.62	64.60	66.91	65.03	65.46	67.20	68.06	68.21	68.21
0.23	67.49	66.04	66.47	65.61	66.33	66.62	67.77	65.90	66.91	66.47	65.32	66.18	67.77
0.25	67.63	66.04	65.61	66.18	66.62	66.91	65.61	65.32	66.33	66.47	67.20	64.88	67.63
0.27	68.21	65.61	66.33	67.05	67.63	68.64	68.64	67.49	66.91	67.63	66.18	66.91	68.64
0.29	68.21	65.90	70.23	67.49	67.34	68.79	68.06	68.50	68.50	67.34	67.77	68.35	70.23
0.31	69.08	65.61	68.35	67.20	67.63	67.63	68.35	68.93	66.62	67.20	68.06	66.33	69.08
0.33	68.79	66.33	68.35	67.49	66.04	69.08	67.49	66.33	68.93	67.92	67.63	68.50	69.08
0.35	69.36	65.75	67.34	68.06	68.50	67.20	68.35	67.49	66.33	68.64	67.77	69.94	69.94
0.37	68.50	66.04	68.21	66.62	68.79	68.93	67.77	67.63	67.34	68.21	67.20	68.50	68.93
0.39	69.08	65.46	68.06	67.92	68.06	68.21	68.64	69.36	67.05	66.04	70.81	69.36	70.81
0.41	69.94	66.33	67.77	69.08	67.20	69.22	66.76	69.94	67.05	68.93	68.06	69.08	69.94
0.43	70.38	67.49	69.08	68.79	68.79	67.63	68.64	67.20	69.94	67.49	67.77	68.79	70.38
0.45	72.25	67.34	68.79	67.34	66.62	68.79	69.08	68.06	68.50	69.22	68.06	68.35	72.25
0.47	71.39	66.47	67.63	66.76	68.35	67.77	68.35	69.22	68.35	68.50	67.77	67.77	71.39
0.49	70.95	67.20	68.50	68.64	66.04	70.23	67.77	68.06	68.50	68.35	69.80	68.21	70.95
0.51	71.53	67.05	68.06	69.36	67.92	68.06	66.91	69.08	68.79	69.51	69.08	68.06	71.53
0.53	70.95	67.20	68.21	68.06	68.64	69.65	68.79	68.79	68.79	69.22	69.22	69.94	70.95
0.55	70.66	67.92	69.94	70.23	69.08	69.65	68.79	69.22	68.79	66.62	68.21	69.36	70.66
0.57	70.38	67.77	69.08	68.35	69.22	69.65	68.93	69.65	68.93	68.79	69.36	68.93	70.38
0.59	70.66	68.35	69.22	69.94	68.79	68.35	69.51	69.36	69.94	68.64	68.79	67.77	70.66
0.61	70.09	68.93	70.81	70.23	68.35	68.50	68.79	69.65	69.08	68.79	69.22	68.93	70.81
0.63	69.80	68.64	69.08	70.09	70.09	68.64	69.65	67.92	69.65	68.79	69.80	68.21	70.09
Max	**72.25**	**68.93**	**70.81**	**70.23**	**70.09**	**70.23**	**69.65**	**69.94**	**69.94**	**69.51**	**70.81**	**69.94**	**72.25**
Ratio	**0.45**	**0.61**	**0.61**	**0.55**	**0.63**	**0.49**	**0.63**	**0.41**	**0.43**	**0.51**	**0.39**	**0.35**	**0.45**

Appendix III

EXPERIMENT RESULTS: IMPACT OF DIMENSION REDUCTION WITH TARGET-DEPENDENT SENTIMENT ANALYSIS

This appendix includes all details of experiment results that are illustrated in chapter 9 for target dependent sentiment analysis.

Table III.1: K-means with PCA (Cosine distance measure).

Dimensions #	**Run #**	**Accuracy**	**Macro-average F1-score**
50	1	50.1	22.6
	2	50.0	22.2
	3	46.8	30.3
	4	45.8	32.1
	5	45.4	32.3
	6	41.2	30.5
	7	39.3	34.5
	8	39.0	32.0
	9	38.9	31.2
	10	30.5	30.0
	11	30.3	26.4
	12	28.8	23.9
	13	28.3	26.8
	14	27.6	24.5
	15	26.0	20.9
	16	25.3	23.6
	17	25.1	13.7
	18	25.0	13.3
	19	24.9	13.3
	20	24.7	13.2
	MAX	**50.1**	**34.5**
	MIN	**24.7**	**13.2**
	AVRG	**34.7**	**24.9**
	CI High	**39.0**	**28.2**
	CI Low	**30.3**	**21.6**

Dimensions #	**Run #**	**Accuracy**	**Macro-average F1-score**
100	1	25.1	13.7
	2	25.0	13.3
	3	24.9	13.3
	4	35.4	30.5
	5	50.1	22.6
	6	21.7	18.5
	7	28.3	28.3
	8	50.0	22.2
	9	44.9	28.9
	10	41.5	31.6
	11	26.7	20.2
	12	29.2	25.8
	13	27.5	22.3
	14	27.6	27.1
	15	23.6	19.4
	16	31.5	30.9
	17	32.9	28.4
	18	24.7	13.2
	19	37.7	32.3
	20	46.1	28.9
	21	44.7	26.9
	22	39.6	31.6
	MAX	**50.1**	**32.3**
	MIN	**21.7**	**13.2**
	AVRG	**33.6**	**24.1**
	CI High	**37.7**	**27.0**
	CI Low	**29.5**	**21.2**
300	1	50.1	22.6
	2	50.0	22.2
	3	46.2	29.3
	4	46.1	31.3
	5	45.5	28.8
	6	44.1	30.6
	7	39.3	34.5
	8	36.6	31.5
	9	32.2	28.6
	10	28.8	24.0
	11	26.4	21.6
	12	26.3	20.5
	13	25.1	13.7
	14	25.0	13.3
	15	24.9	13.3
	16	24.7	13.2
	17	24.0	21.0
	MAX	**50.1**	**34.5**
	MIN	**24.0**	**13.2**
	AVRG	**35.0**	**23.5**
	CI High	**40.2**	**27.2**
	CI Low	**29.8**	**19.9**

Table III.2: Semi-supervised K-means with PCA.

Dims #	Ratio	R1	R2	R3	R4	Max	Average
300	0.01	24.9	24.9	50.1	50.1	50.1	37.5
	0.03	24.9	24.9	50.0	24.9	50.0	31.1
	0.05	24.9	24.9	25.1	25.1	25.1	25.0
	0.07	50.0	24.9	50.1	24.9	50.1	37.5
	0.09	50.0	24.9	25.1	24.9	50.0	31.2
	0.11	50.1	24.9	24.9	37.0	50.1	34.2
	0.13	50.1	24.9	24.9	24.9	50.1	31.2
	0.15	24.9	24.9	50.1	24.9	50.1	31.2
	0.17	24.9	24.9	23.7	50.1	50.1	30.9
	0.19	24.9	24.9	50.1	24.9	50.1	31.2
	0.21	24.9	24.9	24.9	50.1	50.1	31.2
	0.23	24.9	24.9	24.9	23.7	24.9	24.6
	0.25	24.9	24.9	26.0	24.9	26.0	25.1
	0.27	24.9	50.1	50.1	24.9	50.1	37.5
	0.29	24.9	50.1	50.1	24.9	50.1	37.5
	0.31	24.9	50.1	25.1	24.9	50.1	31.3
	0.33	24.9	24.9	24.9	24.0	24.9	24.6
	0.35	24.9	24.9	24.9	24.9	24.9	24.9
	0.37	24.9	24.9	24.9	50.1	50.1	31.2
	0.39	24.9	24.9	24.9	24.9	24.9	24.9
	0.41	24.9	24.9	24.9	24.9	24.9	24.9
	0.43	24.9	24.9	24.9	50.1	50.1	31.2
	0.45	24.9	24.9	24.9	24.9	24.9	24.9
	0.47	24.9	24.9	26.7	24.9	26.7	25.3
	0.49	24.9	24.9	25.7	24.9	25.7	25.1
	0.51	22.8	24.9	24.9	24.9	24.9	24.4
	0.53	22.8	24.9	24.9	24.9	24.9	24.4
	0.55	22.8	24.9	24.4	24.9	24.9	24.2
	0.57	24.9	24.9	24.9	24.9	24.9	24.9
	0.59	24.9	24.9	24.9	23.7	24.9	24.6
	0.61	24.9	24.9	24.9	24.1	24.9	24.7
	0.63	24.9	24.9	24.9	24.9	24.9	24.9
	Max	**50.1**	**50.1**	**50.1**	**50.1**	**50.1**	**37.5**
	Ratio	**0.11**	**0.27**	**0.01**	**0.01**	**0.01**	**0.01**

Dims #	**Ratio**	**R1**	**R2**	**R3**	**R4**	**Max**	**Average**
600	0.01	24.9	24.9	50.0	24.9	50.0	31.1
	0.03	24.9	24.9	25.1	24.9	25.1	24.9
	0.05	24.9	24.9	24.9	25.1	25.1	24.9
	0.07	50.0	24.9	24.9	24.9	50.0	31.1
	0.09	50.0	24.9	24.9	24.9	50.0	31.1
	0.11	50.1	24.9	50.1	24.9	50.1	37.5
	0.13	50.1	24.9	24.9	25.1	50.1	31.3
	0.15	24.9	24.9	50.1	24.9	50.1	31.2
	0.17	24.9	24.9	50.1	50.1	50.1	37.5
	0.19	24.9	24.9	32.5	24.9	32.5	26.8
	0.21	24.9	24.9	24.9	50.1	50.1	31.2
	0.23	24.9	24.9	50.1	24.9	50.1	31.2
	0.25	24.9	24.9	24.9	23.7	24.9	24.6
	0.27	24.9	50.1	36.3	50.1	50.1	40.4
	0.29	24.9	50.1	24.9	24.9	50.1	31.2
	0.31	24.9	50.1	24.9	24.9	50.1	31.2
	0.33	24.9	24.9	24.9	24.9	24.9	24.9
	0.35	24.9	24.9	25.0	24.9	25.0	24.9
	0.37	24.9	24.9	24.9	24.9	24.9	24.9
	0.39	24.9	24.9	24.9	24.9	24.9	24.9
	0.41	24.9	24.9	24.9	24.9	24.9	24.9
	0.43	24.9	24.9	25.0	24.9	25.0	24.9
	0.45	24.9	24.9	24.9	24.9	24.9	24.9
	0.47	24.9	24.9	23.0	24.7	24.9	24.4
	0.49	24.9	24.9	50.1	25.1	50.1	31.3
	0.51	22.8	24.9	23.7	24.9	24.9	24.1
	0.53	22.8	24.9	24.9	24.9	24.9	24.4
	0.55	22.8	24.9	50.1	34.0	50.1	32.9
	0.57	24.9	24.9	23.8	50.1	50.1	30.9
	0.59	24.9	24.9	24.7	24.9	24.9	24.8
	0.61	24.9	24.9	23.8	24.1	24.9	24.4
	0.63	24.9	24.9	24.9	24.9	24.9	24.9
	Max	**50.1**	**50.1**	**50.1**	**50.1**	**50.1**	**40.4**
	Ratio	**0.11**	**0.27**	**0.11**	**0.17**	**0.11**	**0.27**

Table III.3: Clustering and classifying training data when applying our proposed solution.

Run	Accuracy	F1-Score	Run	Accuracy	F1-Score
1	91.2	90.9	37	91.2	90.9
2	91.1	90.9	38	91.1	90.9
3	91.2	90.9	39	91.2	90.9
4	26.8	33.5	40	91.2	90.9
5	4.3	4.0	41	91.1	90.9
6	91.2	90.9	42	91.2	90.9
7	91.1	90.9	43	91.2	90.9
8	91.2	90.9	44	91.2	90.9
9	91.2	90.9	45	91.2	90.9
10	91.2	91.0	46	91.2	90.9
11	91.2	90.9	47	91.2	90.9
12	91.2	90.9	48	91.1	90.9
13	91.2	90.9	49	91.1	90.9
14	91.2	90.9	50	91.2	91.0
15	91.2	90.9	51	91.1	90.9
16	91.1	90.9	**Max**	**91.2**	**91.0**
17	91.1	90.9	**Min**	**4.3**	**4.0**
18	91.2	90.9	**AVRG**	**88.2**	**88.1**
19	91.2	90.9	**CI High**	**92.4**	**92.1**
20	91.1	90.9	**CI Low**	**84.0**	**84.0**
21	91.2	90.9			
22	91.2	91.0			
23	91.2	90.9			
24	91.2	90.9			
25	91.1	90.9			
26	91.1	90.9			
27	91.1	90.9			
28	91.2	90.9			
29	91.2	90.9			
30	91.2	90.9			
31	91.1	90.9			
32	91.1	90.9			
33	91.1	90.9			
34	91.2	90.9			
35	91.1	90.9			
36	91.2	90.9			

Table III.4: Clustering and classifying testing data when applying our proposed solution.

Run	Accuracy	F1-Score	Run	Accuracy	F1-Score
1	92.6	92.3	37	92.6	92.3
2	92.6	92.3	38	92.6	92.3
3	92.6	92.3	39	92.6	92.3
4	92.6	92.3	40	92.6	92.3
5	92.6	92.3	41	92.6	92.3
6	92.6	92.3	42	92.6	92.3
7	92.6	92.3	43	92.6	92.3
8	92.6	92.3	44	92.6	92.3
9	92.6	92.3	45	92.6	92.3
10	92.6	92.3	46	92.6	92.3
11	92.6	92.3	47	26.9	34.1
12	92.6	92.3	48	92.6	92.3
13	92.6	92.3	49	92.6	92.3
14	92.6	92.3	50	92.6	92.3
15	92.6	92.3	51	92.6	92.3
16	92.6	92.3	**Max**	**92.6**	**92.3**
17	92.6	92.3	**Min**	**26.9**	**34.1**
18	92.6	92.3	**AVRG**	**91.3**	**91.1**
19	92.6	92.3	**CI High**	**93.9**	**93.4**
20	92.6	92.3	**CI Low**	**88.8**	**88.8**
21	92.6	92.3			
22	92.6	92.3			
23	92.6	92.3			
24	92.6	92.3			
25	92.6	92.3			
26	92.6	92.3			
27	92.6	92.3			
28	92.6	92.3			
29	92.6	92.3			
30	92.6	92.3			
31	92.6	92.3			
32	92.6	92.3			
33	92.6	92.3			
34	92.6	92.3			
35	92.6	92.3			
36	92.6	92.3			

Table III.5: Clustering testing data and classifying training data when applying our proposed solution.

Run	Accuracy	F1-Score	Run	Accuracy	F1-Score
1	90.8	90.4	37	90.8	90.4
2	90.8	90.4	38	26.2	33.1
3	90.8	90.4	39	90.8	90.4
4	90.8	90.4	40	90.8	90.4
5	90.8	90.4	41	90.8	90.4
6	90.8	90.4	42	90.8	90.4
7	90.8	90.4	43	90.8	90.4
8	90.8	90.4	44	90.8	90.4
9	90.8	90.4	45	90.8	90.4
10	90.8	90.4	46	26.2	33.1
11	90.8	90.4	47	90.8	90.4
12	90.8	90.4	48	90.8	90.4
13	90.8	90.4	49	90.8	90.4
14	90.8	90.4	50	90.8	90.4
15	90.8	90.4	51	90.8	90.4
16	26.2	33.1	**Max**	**90.8**	**90.4**
17	90.8	90.4	**Min**	**6.5**	**6.3**
18	90.8	90.4	**AVRG**	**85.3**	**85.3**
19	90.8	90.4	**CI High**	**90.7**	**90.3**
20	6.5	6.3	**CI Low**	**80.0**	**80.4**
21	90.8	90.4			
22	90.8	90.4			
23	90.8	90.4			
24	90.8	90.4			
25	90.8	90.4			
26	90.8	90.4			
27	90.8	90.4			
28	90.8	90.4			
29	90.8	90.4			
30	90.8	90.4			
31	90.8	90.4			
32	90.8	90.4			
33	90.8	90.4			
34	90.8	90.4			
35	90.8	90.4			
36	90.8	90.4			

Table III.6: Clustering training data and classifying testing data when applying our proposed solution.

Run	Accuracy	F1-Score	Run	Accuracy	F1-Score
1	92.1	91.8	37	92.1	91.8
2	92.1	91.8	38	92.1	91.8
3	92.1	91.8	39	92.1	91.8
4	92.1	91.8	40	92.1	91.8
5	25.9	32.6	41	92.1	91.8
6	92.1	91.8	42	92.1	91.8
7	92.1	91.8	43	92.1	91.8
8	92.1	91.8	44	92.1	91.8
9	92.1	91.8	45	92.1	91.8
10	92.1	91.8	46	92.1	91.8
11	92.1	91.8	47	92.1	91.8
12	92.1	91.8	48	92.1	91.8
13	92.1	91.8	49	92.1	91.8
14	92.1	91.8	50	92.1	91.8
15	92.1	91.8	51	92.1	91.8
16	92.1	91.8	52	92.1	91.8
17	92.1	91.8	53	92.1	91.8
18	92.1	91.8	54	92.1	91.8
19	92.1	91.8	55	92.1	91.8
20	92.1	91.8	56	92.1	91.8
21	92.1	91.8	57	92.1	91.8
22	92.1	91.8	58	92.1	91.8
23	92.1	91.8	59	92.1	91.8
24	92.1	91.8	60	92.1	91.8
25	92.1	91.8	61	92.1	91.8
26	92.1	91.8	62	27.5	34.6
27	92.1	91.8	63	92.1	91.8
28	92.1	91.8	64	92.1	91.8
29	92.1	91.8	65	25.9	32.6
30	92.1	91.8	66	92.1	91.8
31	27.5	34.6	67-96	92.1	91.8
32	92.1	91.8	**Max**	**92.1**	**91.8**
33	92.1	91.8	**Min**	**25.9**	**32.6**
34	92.1	91.8	**AVRG**	**89.3**	**89.4**
35	92.1	91.8	**CI High**	**92.0**	**91.8**
36	92.1	91.8	**CI Low**	**86.7**	**87.0**

Appendix IV

EXPERIMENT RESULTS:

OPEN DOMAIN TARGETED SENTIMENT ANALYSIS

This appendix includes all details of experiment results that are illustrated in chapter 11 for open domain targeted sentiment analysis.

Table IV.1: Semi-supervised learning with dynamic feature generation using English data when labeling ratio=25%.

Fold	Model	Err	C	Obs #	Pred #	NER			SA		
						P	R	F1	P	R	F1
1	Super	76.42	11	347	298	61.74	53.03	57.05	42.62	36.6	39.38
	Semi			347	315	61.59	55.91	58.61	43.81	39.77	41.69
2	Super	75.94	11	324	259	64.86	51.85	57.63	44.02	35.19	39.11
	Semi			324	277	61.73	52.78	56.91	42.24	36.11	38.94
3	Super	73.58	11	346	265	65.66	50.29	56.96	43.77	33.53	37.97
	Semi			346	263	65.02	49.42	56.16	44.87	34.1	38.75
4	Super	76.42	1	318	174	64.37	35.22	45.53	49.43	27.04	34.96
	Semi			318	168	67.86	35.85	46.91	51.79	27.36	35.8
5	Super	73.58	11	346	265	65.66	50.29	56.96	43.77	33.53	37.97
	Semi			346	263	65.02	49.42	56.16	44.87	34.1	38.75
6	Super	75.47	11	319	249	62.25	48.59	54.58	42.17	32.92	36.97
	Semi			319	290	58.62	53.29	55.83	40.34	36.68	38.42
7	Super	75.0	1	309	162	68.52	35.92	47.13	50.62	26.54	34.82
	Semi			309	173	70.52	39.48	50.62	50.29	28.16	36.1
8	Super	72.64	11	320	262	67.94	55.63	61.17	51.53	42.19	46.39
	Semi			320	263	67.68	55.63	61.06	51.33	42.19	46.31
9	Super	76.42	11	346	307	62.21	55.2	58.5	41.04	36.42	38.59
	Semi			346	312	63.14	56.94	59.88	43.59	39.31	41.34
10	Super	75.94	11	319	255	64.31	51.41	57.14	46.67	37.3	41.46
	Semi			319	271	61.62	52.35	56.61	45.02	38.24	41.36

Table IV.2: Semi-supervised learning with dynamic feature generation using English data when labeling ratio=50%.

Fold	Model	Err	C	Obs #	Pred #	NER			SA		
						P	R	F1	P	R	F1
1	Super	76.42	21	347	304	64.47	56.48	60.22	46.05	40.35	43.01
	Semi			347	311	65.92	59.08	62.31	47.91	42.94	45.29
2	Super	74.06	21	324	274	68.25	57.72	62.54	46.72	39.51	42.81
	Semi			324	286	67.83	59.88	63.61	46.85	41.36	43.93
3	Super	71.7	11	346	241	63.07	43.93	51.79	43.57	30.35	35.78
	Semi			346	259	63.71	47.69	54.55	47.1	35.26	40.33
4	Super	73.58	11	318	249	65.06	50.94	57.14	49.8	38.99	43.74
	Semi			318	255	65.1	52.2	57.94	49.41	39.62	43.98
5	Super	74.53	11	340	250	67.2	49.41	56.95	50.0	36.76	42.37
	Semi			340	263	66.16	51.18	57.71	49.43	38.24	43.12
6	Super	70.28	11	319	247	67.61	52.35	59.01	50.2	38.87	43.82
	Semi			319	284	63.03	56.11	59.37	45.77	40.75	43.12
7	Super	71.23	11	309	225	64.89	47.25	54.68	47.11	34.3	39.7
	Semi			309	218	67.43	47.57	55.79	49.54	34.95	40.99
8	Super	73.11	11	320	235	72.34	53.12	61.26	57.45	42.19	48.65
	Semi			320	235	72.34	53.12	61.26	58.3	42.81	49.37
9	Super	74.06	11	346	267	70.04	54.05	61.01	47.57	36.71	41.44
	Semi			346	258	70.54	52.6	60.26	48.45	36.13	41.39
10	Super	73.11	41	319	279	61.65	53.92	57.53	40.14	35.11	37.46
	Semi			319	283	62.19	55.17	58.47	40.64	36.05	38.21

Table IV.3: Semi-supervised learning with dynamic feature generation using English data when labeling ratio=75%.

Fold	Model	Err	C	Obs #	Pred #	NER			SA		
						P	R	F1	P	R	F1
1	Super	71.7	31	347	300	68.67	59.37	63.68	48.0	41.5	44.51
	Semi			347	306	67.97	59.94	63.71	47.39	41.79	44.41
2	Super	70.75	11	324	231	71.43	50.93	59.46	52.81	37.65	43.96
	Semi			324	251	70.92	54.94	61.91	54.18	41.98	47.3
3	Super	69.34	31	346	250	66.0	47.69	55.37	49.6	35.84	41.61
	Semi			346	264	65.15	49.71	56.39	49.62	37.86	42.95
4	Super	74.06	11	318	233	68.67	50.31	58.08	53.22	38.99	45.01
	Semi			318	228	70.18	50.31	58.61	53.51	38.36	44.69
5	Super	71.7	21	340	253	66.4	49.41	56.66	50.59	37.65	43.17
	Semi			340	239	66.95	47.06	55.27	51.46	36.18	42.49
6	Super	68.87	21	319	249	68.67	53.61	60.21	49.0	38.24	42.96
	Semi			319	243	68.72	52.35	59.43	51.44	39.18	44.48
7	Super	69.34	21	309	230	65.65	48.87	56.03	49.13	36.57	41.93
	Semi			309	242	65.29	51.13	57.35	49.17	38.51	43.19
8	Super	71.23	21	320	235	74.47	54.69	63.06	58.72	43.13	49.73
	Semi			320	270	69.63	58.75	63.73	56.3	47.5	51.53
9	Super	70.75	11	346	254	71.26	52.31	60.33	47.64	34.97	40.33
	Semi			346	259	70.27	52.6	60.17	48.65	36.42	41.65
10	Super	70.75	31	319	241	68.05	51.41	58.57	46.89	35.42	40.36
	Semi			319	258	67.05	54.23	59.97	46.9	37.93	41.94

Table IV.4: Semi-supervised learning with dynamic feature generation using Spanish data when labeling ratio=25%.

Fold	Model	Err	C	Obs #	Pred #	NER			SA		
						P	R	F1	P	R	F1
1	Super	70.04	51	677	607	68.2	61.15	64.49	39.87	35.75	37.69
	Semi			677	661	65.66	64.11	64.87	38.58	37.67	38.12
2	Super	69.98	31	656	566	71.02	61.28	65.79	41.34	35.67	38.3
	Semi			656	578	70.24	61.89	65.8	41.35	36.43	38.74
3	Super	67.17	51	676	579	71.5	61.24	65.98	42.49	36.39	39.2
	Semi			676	574	71.6	60.8	65.76	42.68	36.24	39.2
4	Super	70.04	21	641	555	70.27	60.84	65.22	41.62	36.04	38.63
	Semi			641	566	68.73	60.69	64.46	43.11	38.07	40.43
5	Super	70.63	51	669	561	71.84	60.24	65.53	41.53	34.83	37.89
	Semi			669	589	70.63	62.18	66.14	40.24	35.43	37.68
6	Super	67.67	71	663	647	61.98	60.48	61.22	36.63	35.75	36.18
	Semi			663	647	61.98	60.48	61.22	36.63	35.75	36.18
7	Super	68.68	31	651	558	69.18	59.29	63.85	39.25	33.64	36.23
	Semi			651	614	66.78	62.98	64.82	39.25	37.02	38.1
8	Super	68.97	71	681	673	63.45	62.7	63.07	37.0	36.56	36.78
	Semi			681	752	59.57	65.79	62.53	35.11	38.77	36.85
9	Super	67.82	21	661	609	64.2	59.15	61.57	35.96	33.13	34.49
	Semi			661	670	62.54	63.39	62.96	35.52	36.01	35.76
10	Super	68.32	41	675	618	70.06	64.15	66.98	41.26	37.78	39.44
	Semi			675	612	71.57	64.89	68.07	43.3	39.26	41.18

Table IV.5: Semi-supervised learning with dynamic feature generation using Spanish data when labeling ratio=50%.

Fold	Model	Err	C	Obs #	Pred #	NER			SA		
						P	R	F1	P	R	F1
1	Super	65.95	91	677	572	74.83	63.22	68.53	46.15	39.0	42.27
	Semi			677	611	73.81	66.62	70.03	43.37	39.14	41.15
2	Super	66.95	71	656	552	74.64	62.8	68.21	42.39	35.67	38.74
	Semi			656	640	69.37	67.68	68.52	41.09	40.09	40.59
3	Super	66.95	71	676	545	73.76	59.47	65.85	45.87	36.98	40.95
	Semi			676	495	77.98	57.1	65.93	47.88	35.06	40.48
4	Super	67.24	21	641	507	74.16	58.66	65.51	48.13	38.07	42.51
	Semi			641	568	70.42	62.4	66.17	46.48	41.19	43.67
5	Super	68.03	21	669	516	77.13	59.49	67.17	50.78	39.16	44.22
	Semi			669	575	74.61	64.13	68.97	48.87	42.0	45.18
6	Super	65.95	51	663	555	71.89	60.18	65.52	44.14	36.95	40.23
	Semi			663	638	66.14	63.65	64.87	40.91	39.37	40.12
7	Super	66.95	41	651	515	74.95	59.29	66.21	44.66	35.33	39.45
	Semi			651	523	75.14	60.37	66.95	45.7	36.71	40.72
8	Super	67.67	101	681	612	70.75	63.58	66.98	42.65	38.33	40.37
	Semi			681	660	67.88	65.79	66.82	41.82	40.53	41.16
9	Super	67.17	51	661	576	70.14	61.12	65.32	42.19	36.76	39.29
	Semi			661	680	65.0	66.87	65.92	38.97	40.09	39.52
10	Super	67.03	51	675	576	75.0	64.0	69.06	47.22	40.3	43.49
	Semi			675	595	74.45	65.63	69.76	46.39	40.89	43.46

Table IV.6: Semi-supervised learning with dynamic feature generation using Spanish data when labeling ratio=75%.

Fold	**Model**	**Err**	**C**	**Obs #**	**Pred #**	**NER**			**SA**		
						P	**R**	**F1**	**P**	**R**	**F1**
1	Super	65.95	111	677	578	75.78	64.7	69.8	47.23	40.32	43.51
	Semi			677	577	75.91	64.7	69.86	48.7	41.51	44.82
2	Super	64.58	131	656	562	75.62	64.79	69.79	46.44	39.79	42.86
	Semi			656	625	72.48	69.05	70.73	44.16	42.07	43.09
3	Super	64.36	91	676	532	76.32	60.06	67.22	49.44	38.91	43.54
	Semi			676	494	75.71	55.33	63.93	47.57	34.76	40.17
4	Super	65.3	51	641	516	76.94	61.93	68.63	48.64	39.16	43.39
	Semi			641	562	73.67	64.59	68.83	47.51	41.65	44.39
5	Super	66.74	91	669	538	76.95	61.88	68.6	48.51	39.01	43.25
	Semi			669	564	79.26	66.82	72.51	50.0	42.15	45.74
6	Super	63.79	121	663	563	70.69	60.03	64.93	46.0	39.06	42.25
	Semi			663	562	69.93	59.28	64.16	45.55	38.61	41.8
7	Super	66.31	81	651	538	73.42	60.68	66.44	45.35	37.48	41.04
	Semi			651	512	76.56	60.22	67.41	47.66	37.48	41.96
8	Super	66.38	101	681	587	73.59	63.44	68.14	47.36	40.82	43.85
	Semi			681	654	69.72	66.96	68.31	45.11	43.32	44.19
9	Super	63.93	141	661	591	71.24	63.69	67.25	42.13	37.67	39.78
	Semi			661	580	73.1	64.15	68.33	43.62	38.28	40.77
10	Super	65.3	71	675	552	76.09	62.22	68.46	49.46	40.44	44.5
	Semi			675	546	76.74	62.07	68.63	49.45	40.0	44.23

Vitae

Name	: SHADI IBRAHIM HAFIZ ABUDALFA
Nationality	: Palestinian
Date of Birth	: September 15, 1980.
Email	: shadi_abudalfa@hotmail.com.
Address	: Al-Naser, Gaza, Palestine .

Academic Background : Shadi Abudalfa received the BSc and MSc Degrees both in Computer Engineering from the Islamic University of Gaza (IUG), Palestine in 2003 and 2010 respectively. He joined King Fahd University of Petroleum and Minerals (KFUPM) as a full time student to pursue the PhD degree in 2013. He just completed his PhD program in Computer Science and Engineering (CSE) at KFUPM in 2018.

Abudalfa is a lecturer at the University Collage of Applied Sciences, Palestine. From July 2003 to August 2004, he worked as a research assistant at Projects and Research Lab in IUG. From February 2004 to August 2004, he worked as a teaching assistant at faculty of engineering in IUG.

Abudalfa is a member of IEEE and has served as a technical program committee member and a reviewer of some international conferences and a journal. He has published several papers in ISI journals during and before his studies at KFUPM. He has also attended and delivered a research paper to an IEEE conference. His current research interests include artificial intelligence, data mining, data clustering, machine learning, and sentiment analysis.