Improved Sentiment Classification by Multi-model Fusion

by

Lige Gan

A Thesis Submitted to the Department of
Electrical and Computer Engineering
in Fulfillment the Requirement for the Degree of
Master of Science
at
Lakehead University

Thunder Bay, Ontario, Canada

October 2016

Acknowledgement

I first and foremost want to thank Dr. Richard Khoury and Dr. Rachid Benlamri for all the support and encouragement. During the research period in Lakehead University, they patiently supervised thesis work and kindly gave me advices for improving the project. It is with their supervision that this work came into existence.

I am grateful to the committee members: Dr. Ehsan Atoofian, Dr. Jinan Fiaidhi and Dr. Carlos Christoffersen, for taking time and patience in reviewing my thesis and giving significant suggestions.

I am also deeply thankful to my family for encouraging me and providing everything I need when I study abroad. Thanks for my friends in the lab and offering a good academic atmosphere for research. I thank all of my friends and appreciate their help when I felt frustrated during the period. Thanks Lakehead University and Thunder Bay for the environment of living and studying.

Lige Gan

Lakehead University

October 2016

Abstract

Sentiment Analysis (SA), also Opinion Mining, is a sub-field of Data Mining. It aims at studying and analyzing human's sentiments, opinions, emotions or attitudes through their written text. Since all sentiment information are hiding in the text content, a way for acquiring them is using Natural Language Processing (NLP) techniques. On the other hand, with the development of Artificial Intelligence (AI), more Machine Learning (ML) algorithms have been developed. In recent years, these ML algorithms are widely applied in SA field for classifying the text instead of traditional methods. However, selecting an appropriate ML algorithm is a controversial topic in SA research. In this study, we investigated nine commonly used algorithms such as Naïve Bayes (NB), Support Vector Machine (SVM) and Logistic Regression (LR).

A comprehensive comparison of the nine ML algorithms using different metrics enabled to develop a merging model for deriving an optimum algorithm for a specific SA task. The proposed merging model, also called the multi-model, combines multiple ML algorithms' results by using some fusion method to get the best performance out of these algorithms. The performance of the multi-model has also been evaluated and compared to the single ML algorithms.

Keywords: *Sentiment Analysis (SA), Natural Language Processing (NLP), Machine Learning (ML), classification, multi-model, and data fusion*

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Chapter Overview

Data Mining (DM), as a well-known concept in Artificial Intelligence (AI) and Big Data fields, in recent years has attracted the concern of the information industry and IT domains. Sentiment Analysis (SA) as one sub-field of DM, due to its wide use in commercial companies and online social media, has become a popular study area for academic researchers.

This chapter includes a brief overview of the thesis and addresses the problems encountered in SA tasks. In addition, the chapter explains the motivations behind this project and its objectives.

1.2 Problem Definition and Motivation

Sentiment analysis (also known as opinion mining) was first proposed by Pang et al. (2002). Sentiment Analysis (SA) can be defined as the computational treatment of sentiment, opinions or subjectivity in a source text" [1]. Sentiment analysis is also commonly defined as a classification machine learning task using natural language processing (NLP) techniques. This consists of text analysis and computational linguistics techniques to identify or extract subjective knowledge in source materials. Since SA is a sub-discipline of big data, it certainly carries big data model's characteristics. These are described by META Group (now Gartner) in 2001 as the "3Vs" model [28]. This stands for high Volume, Velocity and Variety, meaning that vast amounts of different types of data are generated from various sectors every second while the data transfer by high speed in a virtual world. However, all these data must be able to convert available information and knowledge for serving the real world.

In traditional ways of SA, dealing with such data is time consuming and wastes resources; thus, applying Machine learning (ML) algorithms to SA, to figure out the problems, has become a trend in recent AI developments. Various algorithms can be applied for processing data in SA, and therefore, it is important to determine which algorithm is most suitable. To make the right selection, one needs to conduct proper evaluation of various ML algorithms. While there are many performance evaluation studies for ML algorithms published in different surveys and papers, it is difficult to rely on these results, or to compare results from various papers. This is mainly because, the methods used for these evaluations were applied in different contexts using different test data, and sometimes the findings of some of these evaluations may lead to conflicting or opposite outcomes.

To fairly evaluate ML algorithms, one possible way is to use several evaluation metrics for measuring different aspects of the algorithm. Machine learning algorithms can then be studied and compared under these metrics. This approach is adopted in this research work by proposing a novel technique that first applies data fusion algorithms, and then, merges the results with machine learning algorithms. This multi-algorithm model aims to achieve better results than simply applying single ML algorithms. To measure the robustness of the proposed method, the later has been evaluated and compared with other existing techniques.

1.3 Overview and Thesis Objective

In this study, the main objective is to design a multi-algorithm model, which involves various ML algorithms for text classification, where all sentiment classification results are merged with data fusion methods. Beyond the design, we constructed a multi-metric evaluation model for comparing algorithms and verifying the capability of the multi-algorithm model experimentally. The following points sketch out the three vital components:

- Sentiment Analysis module: The module can show the text's sentiment by determining positive or negative sentiment. It contains nine commonly used machine learning algorithms for classifying text and determining polarities.
- Algorithm evaluating module: This module aims at objectively evaluating the used algorithms from different perspectives. The evaluation involves six metrics instead of one single metric for estimating algorithms performance.

- Multi-algorithm module: This module is the main contribution of the thesis. The aim is to build a multi-algorithm model that outperforms single algorithms. The experiential results show the robustness of the proposed method.

The organization of the thesis is as follows: Chapter 1 describes the problematic addressed in this thesis and discusses the motivations and objectives of the research. Chapter 2 presents background information and related literature review. Chapter 3 introduces the main machine learning algorithms, their evaluation methods and data fusion techniques. In Chapter 4, the experimental results for evaluating the performance of the multi-algorithm model are given and the results are discussed. Finally, the last chapter summarizes the research work in this thesis, provides conclusions, and discusses potential future research work.

Chapter 2

Background and Literature Review

2.1 Chapter Overview

The first part of this chapter illustrates the fundamentals of sentiment analysis, giving a bird's-eye view that explains how a sentiment analysis task works. Subsequently, two major stages in SA, which are feature selection (FS) and sentiment classification are presented.

The second part of the chapter reviews the literature in the SA field. In particular, three papers that provide good thoughts and horizons for the thesis are presented. We also discuss the major contributions and challenges addressed in these papers.

2.2 Sentiment Analysis (SA)

2.2.1 Background

Sentiment analysis, is a computational study of human being's sentiments, opinions and emotions from written texts (also known as corpus). It can be considered as a classification process, categorizing text materials into appropriate topics or sentiments. Since the explosive growth of the network service and social media in the past two decades, the rapid growth of sentiment analysis coincides with the development of social network media. Meanwhile, mining sentiment and acquiring objects' sentiment from social networks become attractive topics in the data mining community. Sentiment analysis also refers to several related fields, such as computational linguistics, Natural Language Processing (NLP), and Machine Learning (ML).

W. Medhat et al. (2014) [1] have investigated several popular methods for sentiment analysis. They provided a panorama of sentiment analysis techniques, as well as the challenges facing the research community. They stated that a sentiment analysis task could be treated as a classification problem that could be applied at three different levels: document-level, sentence-level and aspect-level. Their study focussed on document and sentence level sentiment analysis. Medhat et al. recognized that that aspect-level SA is a more complicated task than the two other levels. They further stated that the main difference between the two studied levels (i.e. sentence level and document level) consists in the basic information unit. In another related study, T. Wilson et al. (2005) focussed solely on document-level SA [2], assuming that sentences can be considered as short document, and therefore, the essential of both classification levels is the same.

2.2.2 Methodology

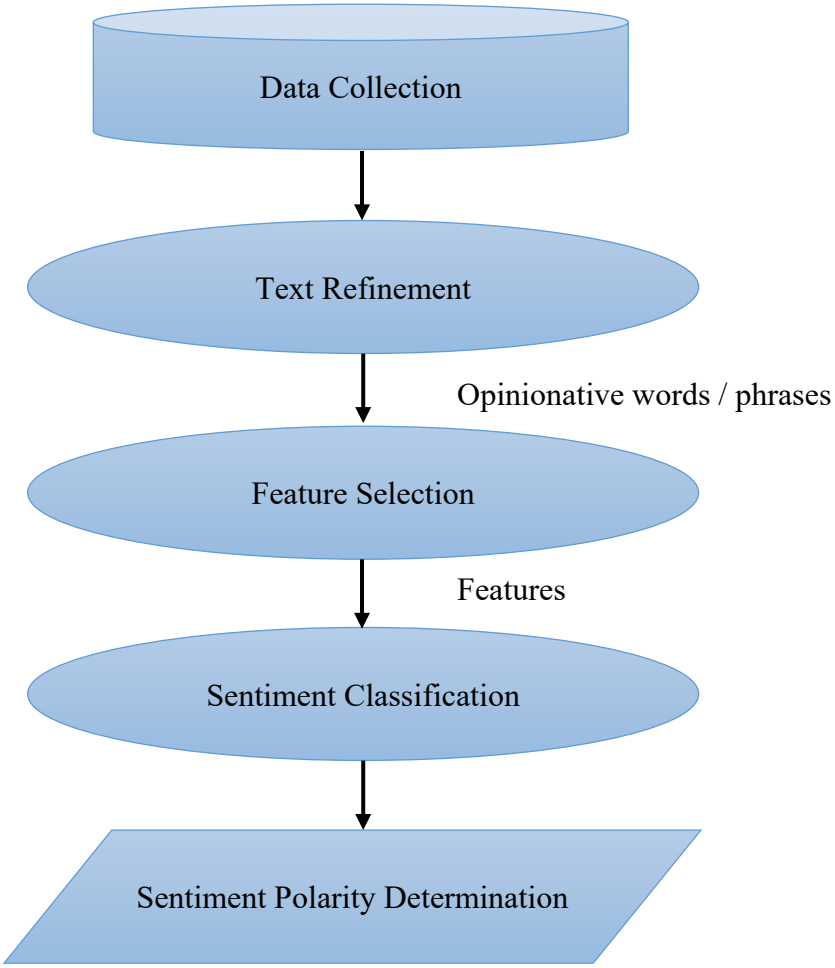

Figure 2.1: Sentiment analysis basic work flow. [1]

As illustrated in Fig. 2.1 [1], the basic work flow of a sentiment analysis task includes five critical stages: data collection, text refinement, feature selection, sentiment classification and sentiment polarity determination. Starting with data collection, massive scale of data is collected from the Internet. Then, in the second stage, the incoming raw data is refined by a text filter and many words and phrases that may contain emotional information are identified and sent to the

next phase. In feature selection stage, a number of potential words and terms are extracted. These form the sentiment features. The subsequent stage, sentiment classification, recognizes appropriate emotional polarity of the text according to the sentiment features. At the end of this task, the sentiment of a document is determined by the classifier.

Specifically, for collecting data stage, since data sources are of various sources, such as online forums, shopping platforms and micro blogs, different types of data may lead to diverse results. In the marketing area for instance, product reviews, while being analyzed, in the online shopping platform usually reveal customers' opinions (good or bad quality) [3].

It should be noted that all text content cannot be classified immediately because the raw data contains too much useless information. Thus, the natural language processing (NLP) technique is involved in the text refinement stage. This stage can be regarded as a filter which can refine useful opinionative words or phrases through some NLP processing techniques like tokenizing texts, tagging Part of Speech (POS), stemming words and so on. A. Pak and P. Parobek (2010) indicated in their paper that the majority of adjectives express emotions and opinions in subjective texts, and adverbs are more often used for providing emotional colors to verbs [4]. Therefore, adjectives and adverbs may be marked or filtered after this procedure. On the other hand, phrases such as idioms and proverbs; taking "cost me an arm and a leg" as an example, contain distinct sentiment orientation instead of using any opinionative words [1]. Such opinion phrases should also be considered as candidates for selecting features.

Feature selection however, is one of the complicated stage in sentiment analysis. Under this process, independent documents or texts are regarded as groups of words, also known as Bag of Words (BOWs). Only a portion of words and terms in this "bag" carry sentiment information. Therefore, BOWs can be treated as a vocabulary set consisting of feature candidates. Because of

this reason, it is not reasonable to employ such scale of words in classification work. Consequently, some feature selection methods are applied to get more informative features that strongly indicate emotional polarities for sentiment classification. The selection methods can roughly be categorized as lexicon-based and statistical methods. The former starts with a set of seed words that have already been annotated by humans. The lexicon extends through synonym detection and relying on online resources, though it has been proved that this approach is not effective because of the difficulties encountered at implementation [5]. Unlike lexicon-based ones, statistical methods are completely automatic. There are several statistical measures [1] that are commonly applied to extracting features such as Term Frequency-Inverse Document Frequency (TF-IDF), Point-wise Mutual Information (PMI), Chi-Square (χ^2), and Latent Semantic Indexing (LSI).

Sentiment classification techniques categorize selected features into their own model [1]. These techniques can be categorized into lexicon-based, machine learning-based or hybrid approaches. Similar to feature selection, lexicon-based approach needs support by sentiment lexicons. On the other hand, machine learning-based methods are more popular in sentiment analysis area. They follow machine learning algorithms that categorize text intelligently. The hybrid approach origins from the combination of lexicon-based and machine learning methods. It combines the merits of the two methods, and hits a higher accuracy in experiment [6]. This method seems to be promising, though, it is not mature yet due to the constraint that the experiment relies on a specific lexicon with a given keyword instead of general uses. In fact, lexicon-based method relies on a lexicon, or a dictionary, which offers sentiment words or terms polarity or their weight, in which the selected features are polarized by the lexicon [7]. Machine learning approach utilizes ML algorithms as classifiers to label text features; thus solving the

classification problem. The specific SA applications of machine learning algorithms are presented in the next section. It should be noted that the kind of ML classifier chosen depends on different requirements. For example, some supervised learning algorithms like Naive Bayes (NB), Support Vector Machine(SVM), Maximum Entropy (MaxEnt or ME) and decision tree are used in numbers of SA academic researches and specified for handling different types of documents.

2.3 Literature Review

There are three papers that have been reviewed in this section. They provide a general background and important foundations in the SA field, as well as inspiration for deep-in exploration in this thesis. These papers focus on feature selection, machine learning classifiers, algorithms evaluation and data fusion methods.

2.3.1 Feature Selection Techniques

A Feature Extraction Process for Sentiment Analysis of Opinions on Services [8]

The majority of literatures on the Internet are more focused on sentiment classification process, though feature selection also plays a core role in SA task. The process of selecting features will improve precision of classification in later procedure; thus, it is necessary to apply some statistic methods associating the extraction process.

H. Siqueira and F. Barros (2010) proposed a prototype system called *WhatMatter* system for Online review sentiment analysis [8]. The paper was more concentrated on feature extraction

process but also covered sentiment classification and visualization parts of the proposed system. By using a distinct corpus for validating the system, they obtained satisfactory results in the experiment.

2.3.1.1 The Process of Feature Selection

The paper detailed the process of feature extraction in the system. The text which holds opinions or emotions was used as the input to the system. These texts are refined for the system to return selected features after their processing. The most important information that they addressed was the two corpora they used in the experiment. One contains two thousand manually analyzed opinions and used by knowledge acquisition process, while the other consist of 200 reviews for validating the system. Both corpora were collected from a Brazilian review-site E-bit[1]. Thus, the language of all reviews was Portuguese. Instead of directly executing the selected features from the reviews, they pre-processed the texts by using NLP techniques for further usage. The NLP techniques they used include sentences and words tokenization, and POS tagging.

Next, every word in the text is tagged depending on their POS. They first identify the most frequent nouns by recording their frequency, and then, selecting the most frequent words in the text. According to their empirical threshold from the developed experiment, 3% of most frequent nouns became initial feature candidates. These words are recorded to a candidate list.

Sometimes the relevant features are expressed implicitly, like the adjective “expensive” that points to the noun “price”. These kind of features are also extracted from the text, adding to

[1] E-bit: http://www.ebit.com.br

the candidates list in this stage. In the study, the researchers manually compiled a list, which contains the connections between some common adjectives or adverbs and their related nouns. These adjectives and adverbs are called relevant feature indicators for finding the implicitly expressed nouns (there are 20 indicators for their chosen domain in the list).

After acquiring an original feature candidate list, they use a "filter" which can select informative features and abandon unrelated nouns by using Point-wise Mutual Information (PMI) measure. The PMI measure offers a formal method to construct the mutual information between the features and the classes [1]. Another researcher, P. Turney (2002) introduces PMI-IR, a PMI variant measure [7]. IR in this work means Information Retrieval for measuring the similarity of classes and words (or terms) by querying an online search engine with the words, and then computing the hits number of matched documents. The PMI-IR is defined by the following equation (see Equation (2.1)):

$$PMI_{IR}(t1, t2) = log_2\left(\frac{Hits(t1 \wedge t2)}{Hits(t1) * Hit(t2)}\right) \quad (2.1)$$

In the formula, $Hits(t1)$ and $Hits(t2)$ count the number of pages containing $t1$ and $t2$, and $Hits(t1 \wedge t2)$ is the number of the pages in which both $t1$ and $t2$ show up. In the *WhatMatter* system, the process executes like this: querying every noun in the candidate list on Google with computing each one's hits; then, computing the number of hits for a word that expresses the noun's opinion; and finally, calculating the hits that contain both words in the domain. According to their results, the system calculates the PMI-IR for each candidate noun, and filters out the irrelevant words based on the empirically determined threshold. Thus, the remaining part of the candidates are regarded as the features of the text. Subsequently, these features are used for later classification process.

2.3.1.2 Results of the experiment

The corpora as mentioned before with two-hundred distinct opinion reviews is used for validating the experiment. Additionally, they implemented sentiment classification and visualization to get the polarity of the reviews and presented the results. The research presented precision and recall for each feature extraction step, and pointed out that after finishing the feature selection process the *WhatMatter* prototype system reached 77% precision and above 91% recall.

2.3.1.3 Advantages and Disadvantages

The system proposed in the paper focuses on the process of feature extracting and shows great accuracy in the developed experiment. The advantage of the system is that the feature selection process has been treated as a vital part of the whole task. It applied an Information Theory named Point-wise Mutual Information for estimating the similarity of the feature candidates and opinions orientation. This statistical measure computes the probabilities of the words and opinions automatically and gains the relevant features rapidly. In addition, the system also returned confident results for the validating corpus.

On the contrary, its drawback is that it still relies on manual work, such as utilizing human collected corpus to assist the system acquiring knowledge. Moreover, the experiment applied a pre-compiled list of adjectives pre-selected by the researchers. The list contains 20 feature indicators to map the relevant nouns. Obviously, the prototype can only be applied in

limited domains because of the specified training corpus and the indicators list. Hence, it is not a widely-used model in SA area.

2.3.2 Sentiment Classification

Thumb up? Sentiment Classification using Machine Learning Techniques [3]

There are many statistical sentiment classification approaches developed from 2010 to 2013. These are surveyed by Medhat et al. (2014) [1]. Based on their literatures the trend of machine learning approaches that researchers used in their exploration is going up. On the contrary, the percentage of lexicon-based method related research in SA area decreases year after year, which means ML-based approach becomes a more popular for classification tasks. Following the trend of SA research, the sentiment classifiers in this paper applied different machine learning algorithms to solving the classification problem.

B. Pang et al. explored the use of ML techniques to sentiment analysis rather than traditional classification methods [3]. In their paper, they used IMDb[2], an online movie review database, as their data source. The motivation for their research is justified by the work of Turney (2002) [7], which reported that movie reviews are the most difficult text content, among so many domains, for sentiment classification using the traditional way. In B. Pang's study [3], the authors applied ML algorithms in dealing with such a problem.

Before the experiment, they made three human-based classifiers, manually selecting indicators or feature words for positive and negative sides. The research suggested that the

[2] IMDb: http://www.imdb.com

highest accuracy comes with only 69%. As a result of the previous experiment, they developed ML classifiers trying to improve the accuracy.

2.3.2.1 Methodologies of Sentiment Classification

In the feature extraction stage the researchers applied n-gram method, which tokenizes the text with n-word sets. For example, tokenizing text word by word is called unigram; while separating text into two-words units is called bigram. The next stage is sending all features to ML models. Pang et al. (2002) built three ML models by applying three different algorithms: Naïve Bayes (NB), Maximum Entropy (ME) and Support Vector Machine (SVM) [3].

The first approach is building a Naïve Bayes classifier; it simply follows Bayes probability rule:

$$P(c|d) = \frac{P(c) \cdot P(d|c)}{P(d)} \tag{2. 2}$$

where c represents the class, d is the given document. Assume that the probability of occurred feature is conditionally independent, m is the number of features $\{f_1, \cdots, f_m\}$, $n_i(d)$ counts the occurring times of the feature f_i. Thus, $P(d|c)$ is the prior probability product of all features occur in document d. Thus the formula can be regarded as:

$$P_{NB}(c|d) = \frac{P(c)\left(\prod_{i=1}^{m} P(f_i|c)^{n_i(d)}\right)}{P(d)} \tag{2. 3}$$

$P(c)$ and $P(d)$ represent the class prior probability and the documents prior probability respectively. Although $P(d)$ is always the same, it would be dropped most of time and only keep

the numerator for discussion. Essentially NB classifier only computes the posted probability for each document. The class which hits the highest probability in a document becomes the document's polarity.

An alternative algorithm in sentiment classification is Maximum Entropy. Unlike Naïve Bayes, this approach has no assumption about the conditional independence between features, so that the result should be more reliable than NB. ME expresses $P_{ME(c|d)}$ with the following formula:

$$P_{ME(c|d)} = \frac{1}{Z(d)} exp\left(\sum_i \lambda_{i,c} F_{i,c}(d,c)\right) \quad (2.4)$$

where $Z(d)$ is explained as a normalization function, $\lambda_{i,c}$ is a feature-weight parameter. According to the paper, this parameter is set for maximizing the entropy. The parameter $F_{i,c}$ is a function for feature f_i and class c, and it is given by:

$$F_{i,c}(d,c') = \begin{cases} 1, & n_i(d) > 0 \; and \; c' = c \\ 0, & otherwise \end{cases} \quad (2.5)$$

Under this function, the features which hold a strong orientation will be set to 1, or 0 otherwise. In the study, they drove the classifier with ten iterations for training the parameter and improving the iterative algorithm [9].

The third technique is Support Vector Machine (SVM). SVM has proved that it is a highly effective way of categorization [10]. Under the two-polarity case, the theory of this algorithm attempts to find a hyperplane $\vec{\omega}$ that can separate each class as large as possible, so the formula is expressed as:

$$\vec{\omega} = \sum_j \alpha_j c_j \vec{d}_j , \ \alpha_j \geq 0 \quad (2.6)$$

The two polarities (positive and negative) to document d_j are indicated by c_j which is set to 1 or -1 respectively. α_j indicate Lagrange multipliers, gained by solving the dual optimization issue. When α_j is greater than zero, $\vec{d}_j$ will be the support vector. All support vectors contribute for constructing a hyperplane $\vec{\omega}$ for splitting the two categories. An illustration of this example is shown in Fig. 2.2 [20].

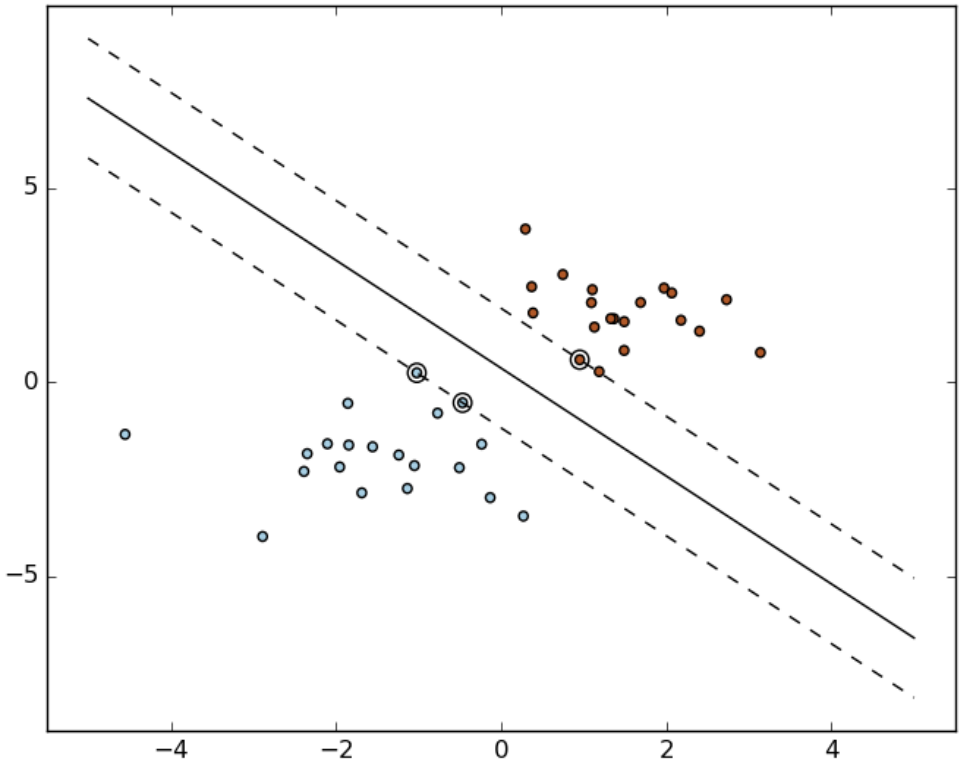

Figure 2.2: Two-class SVM Example. (The highlight points in each side are support vectors for corresponding class, the dash lines are class margins, then solid line represented to the hyperplane.)

2.3.2.2 Experiments and Results

Table 2.1: The results of Pang et al. experiment [3]. (The bond font for highlighting the best result in one experiment.)

No.	Features	# of features	Frequency/ presence	NB	ME	SVM
(1)	unigrams	16165	Freq.	**78.7**	N/A	72.8
(2)	unigrams	16165	Pres.	81.0	80.4	**82.9**
(3)	unigrams + bigrams	32330	Pres.	80.6	80.8	**82.7**
(4)	bigrams	16165	Pres.	77.3	**77.4**	77.1
(5)	unigrams + POS	16695	Pres.	81.5	80.4	**81.9**
(6)	adjectives	2633	Pres.	77.0	**77.7**	75.1
(7)	Top 2633 unigrams	2633	Pres.	80.3	81.0	**81.4**
(8)	unigrams + position	22430	Pres.	81.0	80.1	**81.6**

The experiment corpus is a IMDb movie reviews set with 700 positive and 700 negative documents for testing the classifiers in different feature selection scenarios. The experimental results are given in Table 2.1. For unigram features, firstly B. Pang et al. (2002) [3] counted the frequency of every feature, and the accuracy for NB and SVM classifier are found to be 78.7%

and 72.8% respectively. ME is out of this round due to the function $F_{i,c}$ which only accepts binary values. To compare the frequency count and presence, the researchers binarized the feature occurs counts $n_i(d)$ to 1 if the feature f_i occurred and zero otherwise). Moreover, the presence was applied to all later experiments as well. The results show that ME classifier yielded 80.4% accuracy, and NB and SVM boosted to 81.0% and 82.9% respectively. When the top 15% features are selected for all classifiers, only ME got a little bit improvement, while other classifiers' accuracy were decreased. They also explored another alternative by combining unigram and POS tagging technique. Three classifiers displayed different trends: NB classifier slightly increased by 0.5% from previous value; ME maintained the same accuracy; and SVM reduced accuracy to 81.9%. For bigrams, the three classifiers hit an accuracy of around 77%.

2.3.2.3 Merits and Drawbacks

This research investigated three machine learning sentiment classifiers. Naïve Bayes is the simplest algorithm to implement, though it is constrained by the conditional independent assumption, so that it could not be considered commonly as an outstanding classifier. It should be noted that Maximum Entropy outperformed NB, however, it suffers from one shortcoming that is the time spent iterating for training the parameter. Support Vector Machine is the best one in the exploration, and it is becoming a popular ML classifier applied in many SA applications [1]. In this paper, SVM achieved the best results in the majority of experiments. Furthermore, it was revealed that unigram features can acquire better results than bigrams in the experiments. Therefore, POS tagging was not quite helpful for improving classification accuracy.

2.3.3 Learning Algorithms Evaluation

Performance Evaluation for Learning Algorithms [11]

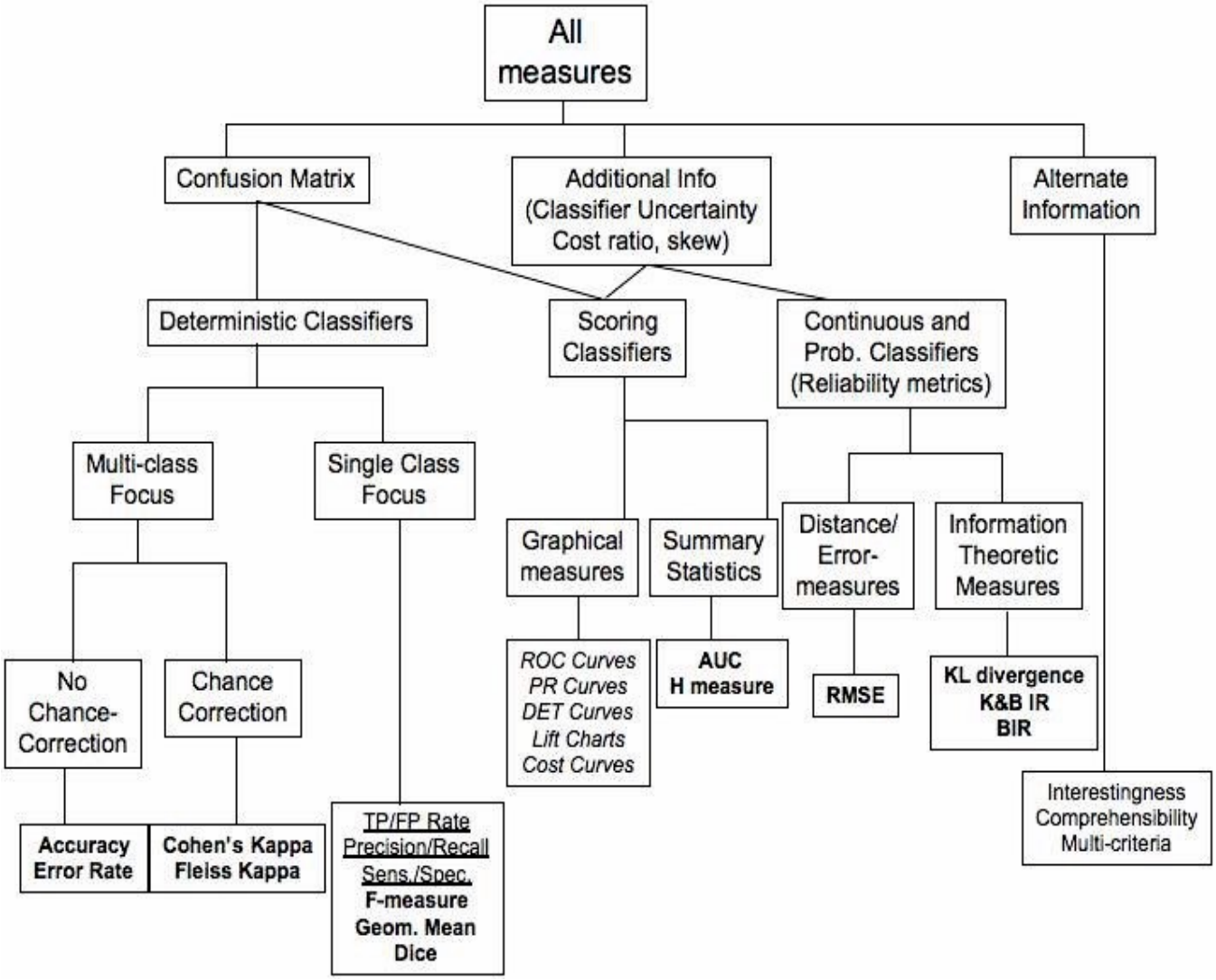

Figure 2.3: Overview of performance evaluation measures. [11]

N. Japkowicz published a book for estimating algorithms performance, "Evaluating Learning Algorithms: A Classification Perspective" (2011) [11]. The book discusses some major aspects of machine learning evaluation with a focus on classification algorithms. Also, the book investigated deeply how to measure the performance of machine learning algorithms, addressing

several techniques of evaluation process, and discussing their relevance and drawbacks in different environments.

The book addressed four areas of evaluating learning algorithms: performance measures, error estimation and re-sampling techniques; statistical significance testing; issues in data set selection; and evaluation benchmarks design.

2.3.3.1 Algorithm Performance Evaluation Measures

Machine Learning algorithms can commonly be categorized into probabilistic, deterministic and scoring algorithms. Literally, probabilistic algorithms or classifiers output probabilities to each class. For acquiring the deterministic class from previous probabilities, Bayesian estimate or Maximum a Posteriori (MAP) is used. On the other hand, deterministic classifiers label the class directly to instances. Scoring algorithms, which issue scores for instances, then, assign them into thresholds, and finally, classify instances according to their located threshold.

The researchers emphasized that sometimes the algorithm's performance is not only determined by the properties of the algorithm itself, but also depends on the evaluation methods. It means that different methods may lead to conflicting results of performance in one machine learning algorithm. To evaluate different learning algorithms more objectively, they offered a scenario for selecting appropriate performance evaluation measures. These are overviewed in Fig. 2.3.

The review only took the confusion matrix into account because of the limitation. Take a two-class model as an example, there are two polarities represented as positive (Pos) and negative (Neg). The confusion matrix for this binary model is illustrated in Table 2.2 [11].

In Table 2.2, TP is the number of correctly classified positives, and FP represents misclassified positive instances. The same theory for FN and TN is adopted torepresent correctly rejected and incorrectly identified negative cases respectively. Hence the number of positive instances is the sum of TP and FN; while the number of negative instances is the sum of FP and TN.

Table 2.2: A Binary Confusion Matrix

True class → Hypothesized class ↓	Pos	Neg
Yes	TP	FP
No	FN	TN
	P=TP+FN	N=FP+TN

Based on this information, a few evaluation measures are performed. These are derived from the confusion matrix. The first one is accuracy, and it reflects the correct classification portion in all test instances. The calculation of accuracy can be written as shown in the equation below. It shows how the knowledge in the confusion matrix has been applied.

$$Accuracy = \frac{(TP+TN)}{(P+N)} \quad (2.7)$$

The second metric is false positive rate (FPR), and it represents the proportion of negative cases that were classified as positive class incorrectly.

The subsequent metrics are Recall and Precision, also known as Sensitivity and Specificity (abbreviated Sens. & Spec.), and reflect comprehension and measure of relevance. Though recall (also called TP Rate or TPR) represents the fraction that how many relevant instances are retrieved; precision on the other hand, represents the proportion of how many retrieved instances are relevant.

2.3.3.2 Analysis

Table 2.3: Hypothetical confusion matrix for H_A.

H_A	Pred_Neg	Pred_Pos	
Act_Neg	**400**	100	N = 500
Act_Pos	300	**200**	P = 500

Table 2.4: Hypothetical confusion matrix for H_B.

H_B	Pred_Neg	Pred_Pos	
Act_Neg	**200**	300	N = 500
Act_Pos	100	**400**	P = 500

One of the issues that is addressed in the book is the fact that applying one measure for algorithm evaluation cannot fairly measure their performance. For instance, there are two hypothetical confusion matrices for hypothetical classifier H_A and H_B , as shown in Table 2.3 and Table 2.4, respectively. Both classifiers yield 60% accuracy, however, their behaviours are totally different and could not be t exhibited from the accuracy. It can be observed that hypothetical classifier H_A has higher negative recognition rate though H_B shows strong ability of tracking positive instances.

For this reason, more evaluation measures should be considered to measure learning algorithms. There are several performance measures that are recommended in the book: The Root-Mean-Square Error (RMSE); the True-Positive Rate and False-Positive Rate (TPR and FPR); Precision (Prec), Recall (Rec); the F Measure; and the area under the ROC curve (AUC). All these measures will help in evaluating learning algorithms more objectively and fairly.

2.3.4 Data Fusion Methods

Fusing Iris, Palmprint and Fingerprint in a Multi-Biometric Recognition System [12]

Instead of focusing on classification knowledge, the paper by H. Naderi et al. (n.d.) [12] reviewed a tri-model biometric recognition system for recognizing objects by fusing the bio-information with iris, palmprint and fingerprint. The contributions of this study are trifold. First, they applied some biometric techniques for extracting features. Second, they used six different

fusion algorithms on the biometric information to obtain the optimum decision. Third, they proposed a novel rank-based fusion algorithm Maximum Inverse Rank (MIR) that was compared to results produced by the used six algorithms. In the remaining part of this section, we focus on the six fusion algorithms.

2.3.4.1 Multi-Model Recognition System

Three unimodal biometrics were applied in the experiment. These are iris recognition system, palmprint recognition system and fingerprint identification system. These three models provide their own recognition determination for each instance. Once the researchers acquire enough knowledge from the three unimodal biometric systems, they considered some strategies that can fuse the determinations effectively and promote the recognition rate. The six well-known fusion algorithms are employed in their system.

There is a critical issue that should be considered before applying the fusion methods. This consists of the individual bio-models calculating the score in their own ways. Due to different types of outputs from individual models, and the different calculation scale, it is necessarily to make a normalization process for each model output. For instance, iris and palmprint recognition modules offer scores based on distances though fingerprint module provides similarity score. As a result of the comparison of normalization techniques (z-score, min-max, tanh and adaptive), z-score yields the best results. Hence, z-score was applied for normalizing all scores into a unique scale. After normalizing the individual scores, they merged the scores using the six data fusion algorithms: Borda Count, ordered weighted averaging, greatest max, greatest mean, greatest product selector and majority voting.

Take Borda Count for instance, it is essentially a classical method for election. For each candidate, the method will assign a weight according to the candidate's rank in a voting cycle. Then the combined matching score is calculated by summing all the weights of the candidate. Usually a higher ranker gets heavier weight for each turn. In this recognition problem, each sample is recognized and matched to a recorded object by three bio-models. The method computes the combined matching score for each object, the highest one will be the recognition result. Borda count is easy-implement due to its simple computing, and it is good at multi-class problem because it takes the ranks in different dimensions into account.

Innovatively, this research proposed an new rank-based algorithm Maximum Inverse Rank (MIR). Basically, it firstly collects the rank by ordering the scores of different classes, and then the combined matching score for each class will be calculated by the following equation:

$$S_c = \sum_{i=1}^{K} \frac{1}{r_{ic}} \tag{2. 8}$$

where S_c is the combined matching score, being calculated by accumulating the class's inversed rank. K is the number of classifiers, and r_{ic} is the rank of *c-th* class in *i-th* classifier's ranking.

2.3.4.2 Experiment Results

Table 2.5: Comparison of different fusion algorithms and in various multi-biometric configurations. [12]

	Iris + Palm + Finger	**Iris + Palm**	**Iris + Finger**	**Palm + Finger**
Greatest Max	**0.9964 ± 0.0037** 1.0000	0.9910 ± 0.0063 $< 10^{-4}$	0.9481 ± 0.0130 $< 10^{-4}$	0.9191 ± 0.0158 $< 10^{-4}$
Greatest Mean	**0.9973 ± 0.0033** 1.0000	0.9966 ± 0.0039 0.0733	0.9800 ± 0.0087 $< 10^{-4}$	0.9783 ± 0.0098 $< 10^{-4}$
Greatest Product	0.9584 ± 0.0127 $< 10^{-4}$	**0.9933 ± 0.0049** 1.0000	0.9151 ± 0.0159 $< 10^{-4}$	0.9304 ± 0.0146 $< 10^{-4}$
OWA	**0.9976 ± 0.0035** 1.0000	0.9579 ± 0.0099 $< 10^{-4}$	0.6604 ± 0.0338 $< 10^{-4}$	0.6717 ± 0.0325 $< 10^{-4}$
Majority Voting	**0.9834 ± 0.0078** 1.0000	0.9579 ± 0.0099 $< 10^{-4}$	0.6604 ± 0.0338 $< 10^{-4}$	0.6717 ± 0.0325 $< 10^{-4}$
Borda Count	0.8550 ± 0.0214 $< 10^{-4}$	**0.9823 ± 0.0088** 1.0000	0.7699 ± 0.0243 $< 10^{-4}$	0.7690 ± 0.0239 $< 10^{-4}$
MIR	**0.9958 ± 0.0053** 1.0000	0.9938 ± 0.0050 0.0004	0.8886 ± 0.0198 $< 10^{-4}$	0.8855 ± 0.0209 $< 10^{-4}$

Individually, the correct classification rates of three biometric reached 0.9729 ± 0.0101, 0.9690 ± 0.0066 and 0.3880 ± 0.0303 for iris, palmprint and fingerprint, respectively according to the unimodal recognition experiments results with the optimal threshold for each biometric (shown in Table 2.5). Due to the three biometric recognition models being fairly independent of each other, there are comparisons which are made up by different combination of three biometric in the results. The results demonstrate as following table (see Table 2.5). It shows the Correct Classification Rate (CCR) of those combinations in six different common fusion algorithms plus MIR algorithm.

For each bi-modal or tri-modal system, the first row shows the Correct Classification Rate (CCR) and its standard deviation, and the second row represents the p-value of t-test of that kind of fusion method. The bold numbers represent a combination gets the best CCR in the fusion algorithm.

2.3.4.3 Analysis

Since the fingerprint biometric recognition has the lowest accuracy rate, the results of bi-modals which are related to fingerprint are worse than the combinations of iris and palmprint. Interestingly, the multi-modal that contains iris, palmprint and fingerprint hit the best results in the majority of the fusion algorithms. It seems that fingerprint recognition does not impact the system to lower accuracy. On the contrary, the fingerprint biometric helps the tri-model gains better result.

It can be observed that the use of multiple biometric models has better performance than using a single biometric model. The fusion algorithms also influence the system performance. It

should also be noted that score-based methods such as Greatest Mean, Greatest Product and OWA achieve better accuracy than rank-based algorithms, such as Borda Count. In this study, the authors proposed a new rank-based fusion method, Maximum Inverse Rank (MIR), which outperforms all other biometric based methods. MIR can weaken the influence coming from bad recognitions or ineffective classifiers, and is sensitive for multi-subject recognition. Hence its potential use in other areas.

Chapter 3

Performance Comparison of

Machine Learning Algorithms

3.1 Chapter Overview

This chapter describes seven widely used supervised machine learning algorithms and some of their variations, including the aforementioned Naïve Bayes and support vector machine(SVM), which will be discussed in details in this chapter. Furthermore, a performance comparison between the nine ML-based classifiers will be studied with various algorithm evaluation methods.

3.2 Machine Learning Algorithms

In this study, nine ML algorithms: Naïve Bayes and its two variations (Multinomial and Bernoulli Naïve Bayes), SVM with two different kernel functions (linear and Radial Basis Function (RBF) kernel), Decision Trees, K Nearest Neighbours, Logistic Regression and Random Forest, will be described and compared among each other. For each ML algorithm, some specific profiles or settings of classifiers will be addressed.

3.2.1 Naïve Bayes and its variations

Naïve Bayes classifier is the most commonly used probabilistic classifier. It follows Bayes' theorem, plus the assumption that every pair of features are independent from each other making it "naive". "Naïve Bayes classification model computes the posterior probability of a class, based on the distribution of the words in the document" [27]. Based on the previous background, each document or instance is represented by some features from the vocabulary as $D = \{f_1, f_2, \cdots, f_i, \cdots f_n\}$. The NB classifier converts each word or term in the instance to a feature vector. The probability of the instance D that is classified to class c can be calculated by the class prior probability $P(c)$ times the product of features' conditional probabilities in class c. The conditional probability of each feature can be computed by using Maximum a Posteriori (MAP). Therefore, the mathematical expression is described as follows (Equation (3.1)) [13]:

$$P_{NB}(c|D) = P(D|c) \cdot P(c) = P(c) \left(\prod_{i=1}^{n} P(f_i|c)\right) \quad (3.1)$$

The two variants of NB, Multinomial Naïve Bayes (MNB) and Bernoulli Naïve Bayes (BNB)[3], calculate $P(D|c)$ according to multinomial distribution [14] and Bernoulli distribution [15], respectively:

$$Multinomial\ \ P(D|c) = P(\langle t_1, \dots, t_i, \dots, t_n \rangle | c) = \prod_{1 \le i \le n} P(t_i|c) \quad (3.2)$$

$$Bernoulli\ \ P(D|c) = P\left(\langle e_1, \dots, e_j, \dots, e_N \rangle \middle| c\right) = \prod_{1 \le j \le N} P\left(e_j \middle| c\right),\ e_j \in \{0, 1\} \quad (3.3)$$

where $\langle t_1, \dots, t_n \rangle$ represents the features sequence, in which the features come from the vocabulary and occur in document D in a Multinomial model. Comparatively, Bernoulli one $\langle e_1, \dots, e_j, \dots, e_N \rangle$ is a binarized vector, where N would be its dimensionality and the vector implies whether each feature occurs in D or not.

According to these differences, the calculation of conditional probabilities in the two distributions [20] are not the same:

$$Multinomial\ \ P(t_i|c) = \frac{T_{ct_i}+1}{\sum_{t' \in V}(T_{ct'}+1)} \quad (3.4)$$

$$Bernoulli\ \ P\left(e_j \middle| c\right) = \frac{N_{ce_j}+1}{N_c+2} \quad (3.5)$$

These expressions illustrate a key difference between the two variants of NB. The multinomial function is more focused on terms count. In the conditional probability equation, the denominator represents the sum of all words in class c. T_{ct_i} denotes the count of word t_i occurred in class c, while $T_{ct'}$ is the total of the words in c. On the other hand, Bernoulli

[3] Naïve Bayes and the variants: http://scikit-learn.org/stable/modules/naive_bayes.html

distribution takes the probability as the fraction of documents which contain e_j in class c; hence N_{ce_j} will be the count of documents that e_j occurred in c, and N_c will be the number of documents in c class. The add-one technique in both models is for eliminating zeros.

Generally, Naïve Bayes is one of the simplest classification model due to its conditional independence assumption and it still performs fairly well. However, it may show unsatisfactory results when the classifier meets interactive features.

3.2.2 Support Vector Machines with Kernel Functions

Support Vector Machines (SVM), is a state-of-the-art ML algorithm for solving classification problems. Essentially a SVM classifier, or so called Support Vector Classification (SVC) [16], based on vector space, can be regarded as a large margin classifier. Take a two-class classification problem as an example, the classifier tries to find a boundary between two classes' margins that can decidedly separate them with a maximum distance.

As mentioned in the introduction of the previous chapter, not all points in the vector space can become the support vectors for supporting the margin. Those points which are impossible to be support vectors will be treated as noise and ignored when constructing a class margin. To filter out the noise, a parameter Nu with an interval $[0, 1]$, has been set for controlling the number of support vectors. The upper bound is for the fraction of errors and the lower bound for the fraction of support vectors, means that higher value gain better error-tolerance. The SVCs used in this thesis all applied the Nu parameter[4], so all SVM classifiers are

[4] NuSVC: http://scikit-learn.org/stable/modules/generated/sklearn.svm.NuSVC.html

mentioned as NuSVC for convenience. It should also be noted that in this research Nu is set to 0.5 because of balancing the number of support vectors and the error tolerance.

In real world scenarios, the majority of datasets are non-linearly inseparable. SVM actually maps all data points into higher dimensional feature spaces, trying to convert them to a separable linear model in vector space. The kernel functions here are applied for the purpose of mapping original feature space to high-dimensional space by calculating the inner products between data [20]. In spite of the kernel functions capability of being customized depending on the demands, some functions have been proven to be capable without customization. In this study two classical kernel functions are selected, which are linear kernel and Radial Basis Function (RBF) kernel (also known as Gaussian kernel) [17]. The two kernel functions are described by the following equations [18] where x_i and x_j are two feature vectors in the vector space; and the γ parameter sets the width of the radial. In this study, the default setting of γ in scikit-learn[5] library has been kept, which is configured as $(1 \,/\, \#\ \text{of features})$.

$$Linear:\ \ K(x_i, x_j) = \ x_i^T\, x_j \tag{3. 6}$$

$$RBF(Gaussian):\ K(x_i, x_j) = exp\left(-\gamma \left\| x_i - x_j \right\|^2\right) \tag{3. 7}$$

In fact, the linear kernel is recommended for text classification by a number of studies because most of text classification problems are linearly separable [19]. On the other hand, RBF is one of the most commonly used kernels due to its suitability towards the majority of situations.

[5] scikit-learn: an open-source machine learning support library in Python. (scikit-learn.org/)

SVC is a dimension-effective classifier. It performs better in high dimensional spaces since text classification problems always bring high-dimensional feature vectors. In addition, SVC also classifies effectively when the number of dimensions is greater than the samples number. At the same time, a selection of kernel functions influences the classifier's performance even though there is no appropriate way to choose an accurate kernel function.

3.2.3 Decision Trees

Decision Tree (DT)[6] is a tree structure predictive model for classification, which reflects a mapping relationship between target attributes and values. In the tree structure, every non-leaf node represents an attribute test, and the branches are the outputs of the test within a threshold. Each leaf node corresponds to a class for the final classification. This method aims to build a model, which is capable of predicting target values by learning decision rules from the given corpus.

For constructing a decision tree, there are some criteria leading to split nodes according to the value of their attributes such as information gain (IG), gain ratio and Gini index. Specifically, Gini index, an impurity-based criterion, is usually used for classification instead of regression problems. In this case, Gini impurity is applied as the strategy to measure the divergences of the values of the target attribute.

One advantage is that DT requires less data preparation, but is simple to implement and interpret. The tree model is visible compared to the abstract data created by the other ML classifiers. However, this method is not very stable but easily overfit from the training data. This

[6] Decision Trees: http://scikit-learn.org/stable/modules/tree.html

means it may construct over-complex trees during the learning process. Due to the necessity of decision rules, it may also generate bias trees when acquiring knowledge from the training data. Hence it is recommended to use a balanced dataset when creating a decision tree.

3.2.4 K-Nearest Neighbors

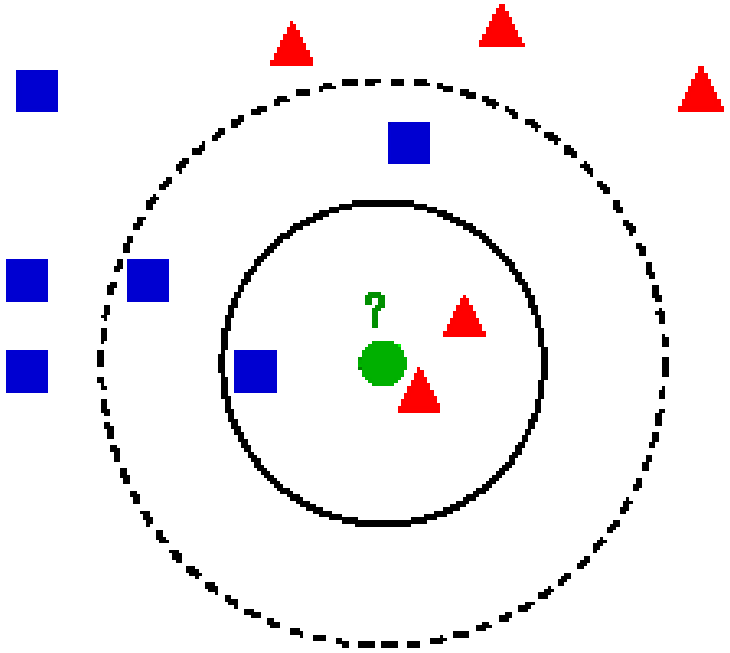

Figure 3.1: A Binary Classification by using KNN. (The solid line is for K=3, when K=5 the line shows as dashed.)

Nearest Neighbour[7], or the so-called K-Nearest Neighbor, is a supervised neighbours-based learning method. It is also an instance-based learning process, in the sense that it only "remembers" instances of the training data instead of building an abstract knowledge model. The principle behind classifying using the nearest neighbour method is to find a number of samples

[7] Nearest Neighbors Classification: http://scikit-learn.org/stable/modules/neighbors.html#classification

in the training data, which have the closest distances to the target, and then using spatial interpolation to predict and label the target according to these "neighbors". The number of samples is a predefined constant K, also called K-Nearest Neighbors (KNN). Generally, the distance between target and sample is usually measured by using standard Euclidean distance.

For instance, in the figure (see Figure 3.1)[8], a two class dataset are used for training. Some of the data points are shown as blue squares and others as red triangles. The figure also shows how the K value influences the predicted result. When $K = 3$, the 3 closest samples are taken for predicting an instance (the green point) within the solid line; in this case the green circle belongs to the triangle class. While the setting of K is 5, it can be observed that the number of squares is more than triangles in the dashed circle, thus the green point belongs to the square class.

KNN is a straight-forward method; due to its theorem, the nearest neighbor model can learn complex concepts using a relatively simplistic approach. On the contrary, its performance is highly restricted by the number of dimensions. When the number of datasets are extremely large, the computational cost of finding neighbours becomes expensive. In this study, the value of K is empirically configured to 5 for gaining satisfactory classification.

3.2.5 Logistic Regression

Logistic Regression (LR) is a part of Regression Analysis in statistics, also known as logit regression. Fundamentally, it estimates the probabilities of independent features for classification by using a logistic function (also called sigmoid function). Additionally, LR is also

[8] The figure referred from https://commons.wikimedia.org/wiki/File:KnnClassification.svg

a special case of the generalized linear model as it obeys independent assumptions. The mathematical notion of a linear model[9] is presented as:

$$z = \hat{y}(w, x) = w_0 + w_1 x_1 + \cdots + w_p x_p \quad (3.8)$$

or

$$\hat{y}(w, x) = w_0 + w^T x \quad (3.9)$$

where $\hat{y}$ is the predicted value of the sample, $x = (x_1, x_2, \cdots, x_n)$ is the set of features of the sample datasets, and the vector $w = (w_1, w_2, \cdots, w_n)$ is the weights that can distinguish features into their corresponding classes. The process of training a LR model consists of finding an appropriate coefficient w by solving an optimization problem. To determine the coefficient w, a logistic function[10] (see equation 3.10) is introduced for mapping $\hat{y}$ between the range $(0, 1)$. The classification problem in LR might then be regarded as calculating the probabilities between the different class, and the setting of thresholds is used for separating classes.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad (3.10)$$

Logistic Regression performs well while the number of samples is small and there are less categories, such as the instances of a binary classification problem. Although this is applying a one-vs.-rest scheme for dealing with multi-class issues, LR still has great accuracy in classification.

[9] Generalized Linear Models: http://scikit-learn.org/stable/modules/linear_model.html

[10] Logistic Regression Classifier: http://www.cnblogs.com/guyj/p/3800519.html

3.2.6 Random Forest

Random Forest (RF) classifier[11], is a classification method that constructs a number of decision trees by random use of different samples in the training dataset. Therefore, RF is an extension of decision trees but overcomes its drawbacks. In the "forest", each decision tree is built from a sample drawn with replacements from the training set. As a result of the forest by randomized trees, the forest bias reasonably increases because of the randomness. However, the average predictive accuracy of trees in the forest is improved, as well as the over-fitting problem can be controlled effectively rather than using a single tree.

Specifically, there are two critical parameters that should be manually assigned. In this experiment, the number of decision trees was set to 10 for the forest, this is the same split strategy as the previous decision tree configuration (Gini index). Then, according to Leo Breiman's suggestion[12], the maximum features value for looking for the best split can be assigned as the square root of the number of all features.

Random forests are often the preferred method for many problems in classification. It is a fast and scalable method that is not concerned about fine tuning parameters such as the settings of SVMs, making it preform exceptionally well in classification.

[11] Forests of randomized trees: http://scikit-learn.org/stable/modules/ensemble.html

[12] Leo Breiman: One of the man who developed the algorithm and holds the trademark "Random Forests".

3.3 Algorithm Performance Evaluating Methods

In this study the binary classification problem has been investigated by using the nine ML algorithms and variants, hence the focus in this section will be the measuring the efficiency of the algorithms. The evaluation methods are first described, then the results are presented and compared. Based on preceding knowledge [11], evaluating algorithm's performance by using only a single method may create bias towards different algorithms. Seven evaluating metrics have been used for measuring algorithms, including the accuracy, true positive rate, false positive rate, precision, recall, F-measure and root mean squared error. These ones will be described in the following subsections.

3.3.1 Accuracy

The accuracy (abbreviated as ACC for convenience) is one of the most commonly used metrics for machine learning algorithms performance evaluation because of its simplicity. It takes the number of correct predictions into account, so accuracy actually represents the percentage of true values in all instances predictions. The drawbacks of accuracy have already been mentioned in the background of this chapter. Evaluating algorithms' performance is still an important aspect of this research but also binary classification problems have been investigated, as well as evaluation metrics, true positive rate (TPR), false positive rate (FPR), precision and recall, F-measure and root mean squared error (RMSE), are involved for acquiring more objective results.

3.3.2 True Positive Rate and False Positive Rate

According to the ontology of performance metrics, the selected ML algorithms are categorized into different types. For instance, Naïve Bayes is a classical probability-based algorithm while others like SVM are typically a deterministic classifier. For the binary classification scenario, the first methodology of evaluation is a confusion matrix. Under the matrix, True Positive-False Positive rate and Precision-Recall are two couples of related metrics. In the confusion matrix, there are some notations that have been used for better comprehension (These notations will be applied in the remaining parts this thesis). Let TP, FP, TN and FN respectively be the number of true positives, false positives, true negatives and false negatives. True positive rate (TPR) is the ratio of positives that are correctly recognized in all positive instances. False positive rate (FPR) is the proportion of positives which are wrongly classified into negative classes. As opposed to accuracy, TPR and FPR can be described respectively by Equation (3.11) and Equation (3.12) [11]:

$$TPR = \frac{TP}{(TP+FN)} \tag{3. 11}$$

$$FPR = \frac{FP}{(FP+TN)} \tag{3. 12}$$

where $(TP + FN)$ and $(FP + TN)$ separately represent the numbers of actual positive instances and negative instances. Normally, the values of TPR and FPR range from 0 to 1; though they are opposite to each other. Essentially, a good performance classifier will acquire high TPR and low FPR.

3.3.3 Precision and Recall

Subsequently, another pair of metrics that is also related to the confusion matrix, recall and precision, are two basic metrics in the information retrieval area, however they are still suitable in this case. For algorithm evaluation, recall and precision are more effective in two-class classification problems. Recall is defined the same way as TPR, in which Recall represents how many instances are identified as positive in all positive sample sets. Precision, also called positive predictive values, represents the ratio of positive instances in the instances set which are classified as positive by the classifier. Precision is a different concept from accuracy. The former is more concentrated on the relations inside positive classes, while accuracy merely focuses on all right instances. The reason that recall and precision are added into the evaluation is that these two are ignoring to detect the opposite, while accuracy is more holistic. The two metrics can be written as shown in Equations (3.13) and (3.14) [11].

$$Recall = \frac{TP}{(TP+FN)} \tag{3. 13}$$

$$Precision = \frac{TP}{(TP+FP)} \tag{3. 14}$$

3.3.4 F1-Measure

Despite there is no explicit nexus that can be identified within the preceding two equations, recall and precision is a pair of interactive metrics. However, it is difficult to keep both metrics with high values in a classifier evaluation at the same time. Additionally, there is a trade-off between these two metrics. Also, by only comparing classifiers' recall and precision, it

may be hard to get satisfactory results. Thus, for balancing such biases between recall and precision, the F-measure is involved to prevent this contradiction. F-measure, as known as F-Score or F1 measure, is a metric that add an averaged weight to recall and precision for balancing the relative contribution of both metrics. The $F1$ score reaches its best at 1 and the worst value at 0, hence the interpretation of the F1 score for binary classification can be represented by Equation (3.15) [21]:

$$F1 = 2 \cdot \frac{(Precision \cdot Recall)}{(Precision + Recall)} \tag{3.15}$$

3.3.5 Root Mean Squared Error

Another metric, root mean squared error (RMSE), is used for measuring the deviation between the predictions and the true targets. Rather than the former metrics, RMSE is recommended by Caruana et al. (2004) in their research [22]; as it is a robust metric that is usually used in two-class scenarios but can also be applicable to regression and probabilistic problems. The equation of RMSE is described by Equation (3.16) [23]:

$$RMSE = \sqrt{\frac{1}{N}\sum\big(Pred(C) - True(C)\big)^2} \tag{3.16}$$

where N is the number of instance, $Pred(C)$ represents a prediction confidence of an instance which is given by classifier C, and $True(C)$ is the true value of the instance. Generally, in binary classification scenarios, if the classifier's outputs are naturally discrete and are in $\{0, 1\}$, the mean squared error (MSE) is interpreted as $(1 - ACC)$ where ACC is the accuracy of the classifier, and RMSE will be $\sqrt{(1 - ACC)}$ [23]. A lower RMSE means the classification

results are close to the true values; otherwise lager values of RMSE imply that the predictions deviate from the actual targets.

Chapter 4

Multi-Model Fusion System

for Classification

4.1 Chapter Overview

This chapter presents a novel classification system, called multi-model, which is a fusion model combining several ML algorithms with different fusion strategies to improve classification performance. The chapter provides a brief overview of the multi-model, followed by a detailed description of the fusion methods. Finally, the fusion strategies are addressed to select outstanding ML algorithms to build the multi-model system.

4.2 Multi-Model Fusion System

In the experiments presented in Chapter 3, various machine learning classifiers exhibited different classification behaviours according to the performance metrics results. The performance of these algorithms varies— some may not be stable overall, some are good at one polarity classification but worse in classifying another one, and others may yield greater accuracy with high bias.

To overcome the disadvantages of individual classifiers and to take advantage of the merits of the different classifiers, the research work presented in this thesis is proposing a multi-model system. The model can be described as a feasible scenario which takes into account multiple classifiers, and then merges their classification results with appropriate data fusion methods. In other word, the multi-model certainly contains several ML classifiers. The group of well-performing algorithms are exploited and their classification outputs are merged by some fusion techniques. Thus, the final classification result of this model is expected to reach better performance than any individual classifier.

4.2.1 The Design of the Multi-Model

The proposed multi-model is illustrated in Fig. 4.1. The multi-model is made up of two components: trained classifiers and fusion methods model. A sample of tweets is sent to the model. The tweets are then classified by a group of classifiers simultaneously. Their classification results, called here intermediate products, are then passed to a fusion model for processing by seven fusion methods. Each fusion method will return its own decision for

comparison. After the merging step, the outputs of fusion methods are to be stored individually for subsequent analysis.

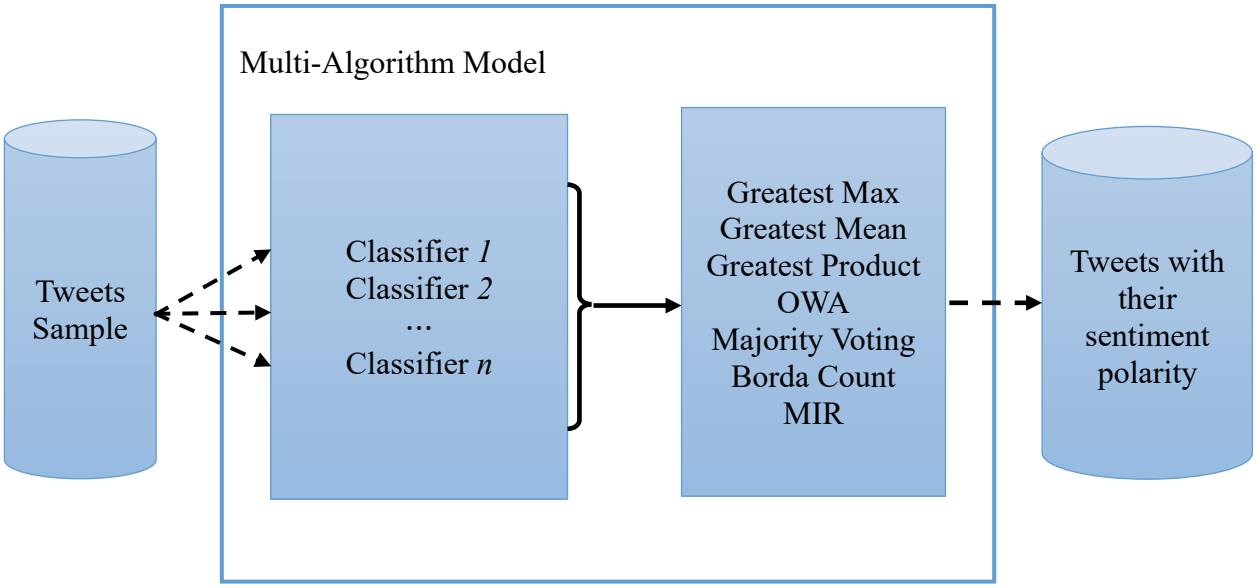

Figure 4.1: The design of the Multi-Model

In the following sections, the seven fusion methods mentioned in Fig. 4.1 are reviewed. Finding the best appropriate choice of the fusion method is discussed in Chapter 5.

4.2.2 Work Flow of Sentiment Analysis Task with the Proposed Model

Rather than reviewing separately each component of the proposed system, in Fig. 4.2, the complete functionality of the multi-model system is discussed. The figure shows the structure of the system where the sentiment analysis module is integrated with the multi-model when classifying the tweet dataset.

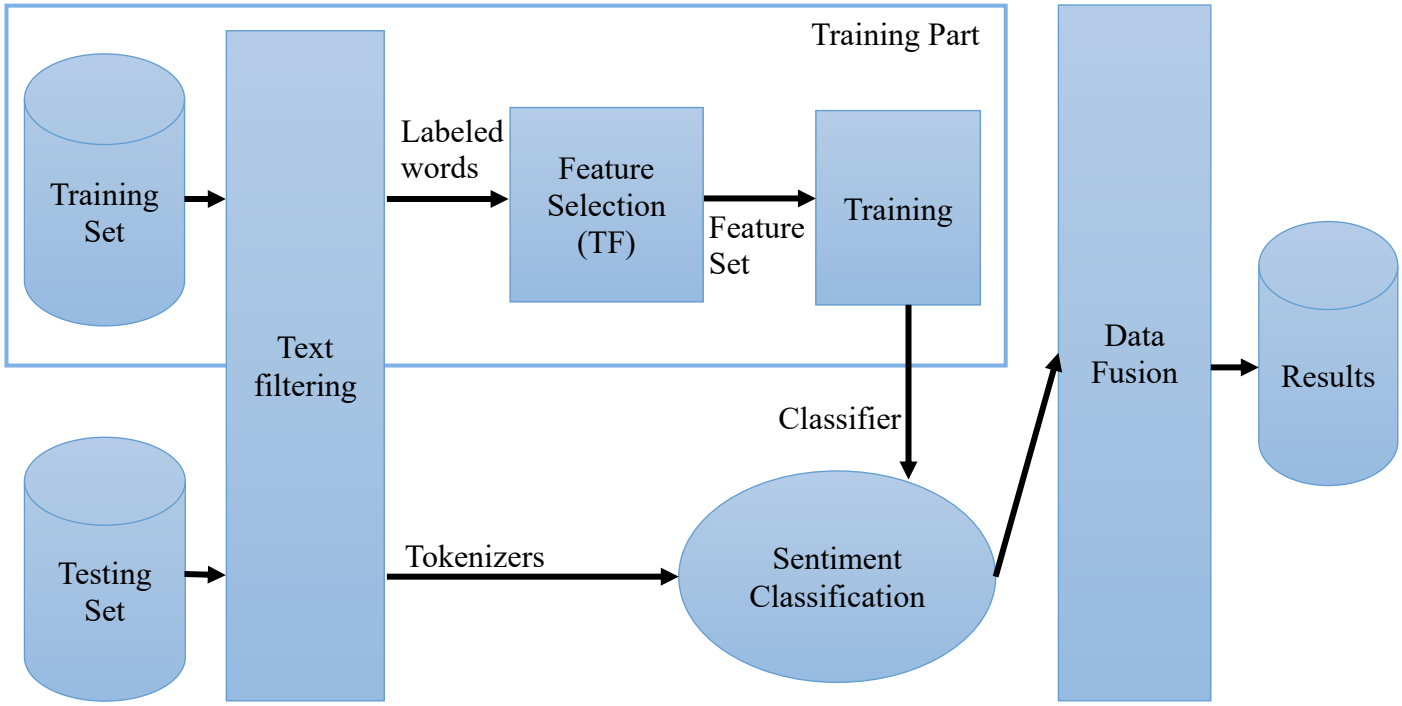

Figure 4.1: Basic work flow of the system

4.3 Fusion Method

To exceed the performance of individual classifiers, the multi-model system is proposed. Research in various areas indicate that it is possible to combine different classifiers by applying data fusion methods. In this section, the seven commonly used data fusion strategies along with the novel method (MIR) implemented in the study are discussed in detail [12]. The study includes majority voting, Borda count, ordered weighted averaging (OWA), greatest max, greatest mean, greatest product, and maximum inverse rank (MIR).

It should be noted that there are some common parameters which are used throughout the methods addressed in this section. Parameter C is the number of classes; parameter p_{ic} is the value of score for class c given by i-th classifier, and r_{ic} represent the rank of c-th subject given by i-th classifier.

4.3.1 Majority Voting

Majority voting is the most widely and simplest used method for solving data fusion problem. Each individual classifier offers its decision, and the combined matching score of each class is denoted by the number of classifiers which rank the class highest. The final result is determined as the class which acquires the most number of votes.

$$S_c = \sum_{i=1}^{K} r1_{ic} \tag{4. 1}$$

$$r1_{ic} = \begin{cases} 1 \; if \; r_{ic} = max_c r_{ic} \\ 0 \; otherwise \end{cases} \tag{4. 2}$$

4.3.2 Ordered Weighted Averaging (OWA)

The second strategy is Ordered Weighted Averaging (OWA). This method is commonly used because a weight is assigned to each classifier, which can reflect the confidence of the decisions made by each classifier. In the following equations, w_i is the weight for *i*-th classifier, m_{i1} and m_{i2} representing the first and second best scores of output classes offered by *i*-th classifier, and K is the number of classifier. Hence, the approach to compute w_i and the combined score S_c are given by equation (4.3) and (4.4) respectively as:

$$w_i = \frac{|m_{i1} - m_{i2}|}{m_{i1}} \tag{4. 3}$$

$$S_c = \frac{\sum_{i=1}^{K} w_i p_{ic}}{\sum_{i=1}^{K} w_i} \tag{4. 4}$$

4.3.3 Greatest Max

The greatest max is an easy-implementation option which considers the maximum score in the group classifiers as its combined matching score for class c.

$$S_c = max_i \, p_{ic} \tag{4. 5}$$

4.3.4 Greatest Mean

The greatest mean is acquired by the mean calculation for the group of classifiers. The method computes the average scores among all the classifiers for class c as the combined matching score.

$$S_c = \frac{1}{K} \sum_{i=1}^{K} p_{ic} \tag{4. 6}$$

4.3.5 Greatest Product

Likewise, this method calculates the greatest product for the classifiers. Similar to previous strategies, the combined matching score is calculated by multiplying all scores given by different classifiers.

$$S_c = \prod_{i=1}^{K} p_{ic} \tag{4. 7}$$

4.3.6 Borda Count

Borda Count is a data fusion algorithm that is based on ranking. It calculates the combined matching score for each class according to a rank that is provided by each classifier. The method for calculating Borda count score is described by equation (4.8). The parameter C in the equation represents the number of classes, since the research concentrates on bi-class problem, the value of C is 2.

$$S_c = \sum_{i=1}^{K} (C - r_{ic} + 1) \tag{4. 8}$$

4.3.7 Maximum Inverse Rank (MIR)

Maximum Inverse Rank (MIR) is proposed by H. Naderi et al. (n.d) [12] and is addressed in Chapter 2. The combined score, essentially calculates the sum of inversed rank in each class. This is given by equation (4.9).

$$S_c = \sum_{i=1}^{K} \frac{1}{r_{ic}} \tag{4. 9}$$

4.4 Machine Learning Algorithms Selection

In Chapter 3, seven different machine learning algorithms and two of their variations were introduced. The research question in this section is to identify the algorithm that should be used to constitute the multi-model. In this research, both options—combining all available

algorithms or considering only some specific algorithms are discussed. The study also makes a comparison of the two options.

In the former option, nine algorithms are calculated to make an “all-in-one” fusion by the multi-model, which means that each fusion method in the model collects and then processes the nine outputs of the classifiers individually. However, the combination is not only time consuming but is also a waste of resource regardless of the performance. Compared to dealing with abundance of data, and to avoid massive calculations; the research selects three out of nine algorithms (as the second option) based on the following two principles: choosing the three most unrelated classifiers for the multi-model by calculating the correlation-coefficient to individual classifiers in the testing phrase, or choosing the three best-performance classifiers to build the model by comparing the performance metrics of the nine individual classifiers. In the next chapter, the three criteria of algorithm selection are compared.

Chapter 5

Experimental Results and Analysis

5.1 Chapter Overview

This chapter contains four parts: the first part describes the experimental setup; the second shows the performance evaluation for the nine machine learning algorithms; part three presents the evaluation metrics of the multi-model by merging all classifiers; and finally, the last part evaluates the proposed multi-model approach based on the two options described in Chapter 4 (i.e. selecting the most unrelated three classifiers or the best-performing three classifiers).

5.2 Experiments Setup

For the experiment data, NLTK (Natural Language Toolkit)[13], a platform that offers interfaces to processing natural language, provided a corpus of Twitter[14] data. This Twitter corpus contains ten-thousand complete tweets, comprised of two classes (positive and negative), each of which includes 5,000 tweets in the data set. It should be mentioned that these tweets are selected randomly from the Twitter stream and cover various topics. Their polarities are labeled manually as positive or negative.

Due to the limited volume of data, a statistics technique called cross-validation (CV) is exploited. This technique aims at splitting sample data as random and independent subsets for training and testing individual classifiers but also for verifying the fusion methods' performances. Basically, k-fold cross validation is a common approach in the data mining field. This method equally separates the raw data into k sets, called here k "folds"; each time the test classifier uses $k - 1$ folds for training then the remaining fold for testing. The previous procedure is repeated k times, with each fold of sample data exactly applied once for validation purpose. The final results would be the average over the splits. K-fold cross-validation guarantees no overlap in the experiments, and it also avoids data over-fitting or under-fitting. 10-fold cross-validation is usually effective, so it acquires robust and persuasive validation results for analysis.

In the text pre-processing stage, the original tweets are filtered, and then some of the words in the training set are selected as features. Firstly, some methods that eliminate text noise such as repetitions removal, user names and hyperlinks replacement, word stemming, are

[13] Natural Language Toolkit: http://www.nltk.org/api/nltk.corpus.html#module-nltk.corpus
[14] Twitter, online social services: https://twitter.com/

exploited in the stage. Furthermore, SentiWordNet[15], a sentiment dictionary is also applied for assisting finding sentiment indicators from training set. The sentiment lexical resource provides sentiment polarity, score and synset for lots of sentiment words. These attributes help extract features from the tweets sample for building a feature set. Additionally, all ML algorithms mentioned in the previous chapter share the same knowledge for training themselves as classifiers.

For the testing phase, all the trained classifiers make use of the tweets in testing fold to validate their performances. Classifiers classify this subset and the results are stored and then applied to evaluating the performances.

It should be noted that the classification results given by individual classifiers may calculate in different levels. For example, Naïve Bayes classifier is a typical probabilistic-based one, where the output scores of this classifier are classes' probabilities, while other methods such as k-nearest neighbor returns distance scores. Also, these scores may not be rated on the same numerical scales, thus, when acquiring comparable data, the outputs should be normalized before evaluation. There are some well-known techniques for normalizing the outputs into the same level such as Min-Max (MM), Z-score (ZS), Tanh (TH), and Two-Quadrics (QQ) [24]. In this study, z-score is implemented for normalization.

Z-score implies how much distance from the standard deviation is the raw data from the mean. Generally, this method will transform data to a distribution for which the mean is 0 and the standard deviation is 1 [24]. In this case, the method can be described by Equation (5.1):

[15] SentiWordNet: http://sentiwordnet.isti.cnr.it/

$$Z = \frac{s - mean(S)}{std(S)} \quad (5.1)$$

where s is the raw score that comes from the score set S, and $mean(S)$ and $std(S)$ are the mean and standard deviation of S.

All classification results of ML classifiers can be evaluated and compared in the same scale after applying z-score normalization. Subsequently, the seven evaluation metrics are used for estimating individual classifiers' performances based on the normalized results. After running 10-fold cross validation five times, the evaluation results are more reliable for analysis.

5.3 Individual Classifiers Performance Evaluation

This section discusses the nine classifiers' performance in a group. This group contains Naïve Bayes (NB), Multinomial Naïve Bayes (MNB), Bernoulli Naïve Bayes (BNB), SVM with linear kernel (SVM_linear), SVM with RBF kernel (SVM_rbf), Decision Tree (DT), K-Nearest Neighbors (KNN), Logistic Regression (LR) and Random Forest (RF). In addition, the seven evaluation metrics, which are accuracy, true positive rate, false positive rate, precision, recall, F1 measure and root-mean-square error which are abbreviated by ACC, TPR, FPR, PRE, REC, F1 and RMSE, respectively.

Table 5.1 presents the comparison across the nine classifiers with seven evaluation metrics. For each classifier, the two values in the metrics represent the average after ten-fold cross-validation and the standard deviation. These results illustrate one of the evaluation issues in Japkowicz's book [11]; namely, different metrics may present diverse performance for one

Table 5.1: Comparison of nine classifiers in various evaluation metrics.

Algorithm	ACC	TPR	FPR	PRE	REC	F1	RMSE
NB	0.7155±0.0128	0.7693±0.0190	0.3384±0.0207	0.6945±0.0195	0.7693±0.0190	0.7298±0.0155	1.5293±0.0170
MNB	0.7161±0.0130	0.7015±0.0647	0.2662±0.0642	0.7279±0.0423	0.7015±0.0647	0.7107±0.0199	0.9995±0.0229
BNB	0.7165±0.0124	0.7695±0.0191	0.3366±0.0200	0.6957±0.0193	0.7695±0.0191	0.7305±0.0152	1.0067±0.0222
SVM_linear	0.7087±0.0123	0.6457±0.0636	0.2271±0.0620	0.7432±0.0355	0.6457±0.0636	0.6878±0.0244	1.0246±0.0216
SVM_rbf	0.7031±0.0144	0.7980±0.0195	0.3916±0.0284	0.6710±0.0214	0.7980±0.0195	0.7287±0.0144	1.0361±0.0243
DT	0.6770±0.0146	0.7331±0.0689	0.3777±0.0692	0.6625±0.0342	0.7331±0.0689	0.6927±0.0253	1.0933±0.0258
KNN	0.6686±0.0175	0.6669±0.0805	0.3311±0.0791	0.6725±0.0336	0.6669±0.0805	0.6656±0.0351	1.1067±0.0287
LR	0.7207±0.0127	0.7544±0.0347	0.3125±0.0340	0.7077±0.0246	0.7544±0.0347	0.7294±0.0167	0.9966±0.0226
RF	0.7004±0.0167	0.7242±0.0701	0.3221±0.0661	0.6950±0.0361	0.7242±0.0701	0.7058±0.0273	1.0333±0.0294

classifier, sometimes perform totally opposite. For instance, Naïve Bayes yields better accuracy at 0.7155 ± 0.0128 and TPR/recall at 0.7693 ± 0.0190 in the group. However, it also yields bad result in RMSE (high value means weak performance) with the highest value amongst the group of the nine classifiers. It means that the classifier might get a good performance in accuracy metrics, although it still performs terribly when measured by RMSE. To simplify the table and also for good understanding, the ranks for each individual classifier in the evaluation metrics have also been demonstrated in Table 5.2.

Table 5.2: Classifiers' ranks in different evaluation metrics

Algorithm	ACC	TPR	FPR	PRE	REC	F1	RMSE
NB	4	3	7	6	3	2	9
MNB	3	7	2	2	7	5	2
BNB	2	2	6	4	2	1	3
SVM_linear	5	9	1	1	9	8	4
SVM_rbf	6	1	9	8	1	4	6
DT	8	5	8	9	5	7	7
KNN	9	8	5	7	8	9	8
LR	1	4	3	3	4	3	1
RF	7	6	4	5	6	6	5

Starting with the dimension of the algorithm, based on the previous two tables, some classifiers can be easily observed, in that their performance somehow are stable from their results when considering the metrics and their ranks. For example, the logistic regression classifier, can be ranked as the top performing classifier in the majority of evaluation metrics (the classifier who gets the best in ACC and RMSE with 0.7207 ± 0.0127 and 0.9966 ± 0.0226, also keeps

the third or the fourth rankings in the rest of the metrics). Bernoulli Naïve Bayes ranks second-best out of the four of seven measures, and the other three metrics also perform well above average. On the contrary, the random forest classifier shows mediocre results, its rank is uniformly wavering in the fifth to seventh rank. In addition, the same situation occurred in the decision tree, the ranks for this classifier are also hesitating on the sixth and the eighth rank.

Nevertheless, it is still difficult to define whether the performance of the classifiers is good or not since the diverse ranks for one classifier is different in various metrics. Take the SVM with linear kernel as an example, it performs the best in precision though it yields the worst rank in the recall metrics in comparison. Regarding the harmonic mean of precision and recall, F1-measure for linear SVM is also lower than average. A similar situation occurs in RBF SVM; it gets the best in one metric (TPR/recall) but the worst by another (FPR). Even though K-Nearest Neighbors classifier has worse ranks in six metrics but shows good capability in the one which is FPR metric. This phenomenon indicates that the classifier's performances are uneven in distinct evaluation methods.

Moreover, classifiers may perform unstably in comparison. The multinomial Naïve Bayes classifier, for example, mostly oscillates between the second-best and the third worst in the metrics.

For the evaluation metrics dimension, the classification behaviours of the classifiers vary according to the results in different metrics. TPR and FPR are the two opposite metrics in the evaluation respectively, representing the true positive rate and false positive rate. In the majority of situations, a high TPR value leads to a low FPR value in one classifier. Thus, in Table 5.2 the ranks of the two are normally reversed (It should be noticed that the lower value is marked as higher rank for FPR rank in the table).

Accuracy usually presents the number of correct instances that the classifier hit in the entire sample set, although it is limited in observation of the inner behaviour. The multinomial Naïve Bayes got the second best in accuracy but in the metric of true positive rate it merely ranks at seventh. In other words, it has high accuracy than others although it has weak capability in recognizing the positive instances.

Furthermore, root mean squared error can reflect the deviation between the true value and predicted confidence value of the instance by the classifier. When the value of RMSE goes up, a large divergence of the prediction to the true value has occurred. In the rank table (Table 5.2) the classifier who gains higher rank indicates lower value in the RMSE metric. Thus, the metric can also assist to observe the classifier's behaviour. Still taking Naïve Bayes as an example, it gains the RMSE value at 1.5293 ± 0.0170, which is beyond the second-highest 0.4226 becoming the worst one in the metric even though its accuracy is above average and good at classifying positives (the third rank in TPR). It may be inferred that the classification of Naïve Bayes is extremely skewed (either correctly or wrongly classifying with high confidence).

5.4 Merging Scenario of the Multi-model

In this experiment, we identify which classification algorithm, in the previous comparison, should be selected and merged together into the multi-model. For studying this problem, two available fusion options are proposed; either merging all algorithms together, or only selecting some of them to fuse.

5.4.1 Multi-model with the Nine Algorithms Fusion

In this sub-section, the implementation results of the nine classifiers fusion are presented and marked as Set 1. Specifically, the performance results of the seven fusion algorithms (Borda count (Borda), ordered weighted averaging (OWA), the greatest max (Max), the greatest mean (Mean), the greatest product (Product), majority voting (Voting) and Maximum Inverse Rank (MIR)) based on the six metrics, are, displayed Table 5.3. The averages and the standard deviations of each fusion method is presented after running five times 10-fold cross validation.

Compared to the individual classifiers' performance metrics, all seven fusion methods show better capability than individual classifiers. Most of them seem to suppress more than half individual classifiers (the details of ranking between individuals and Set 1 can be seen in the appendix). For instance, the greatest mean is the best one in ranking of the seven metrics except TPR/recall, although TPR/recall still shows mediocre ranking in the group of fusion methods. In fact, excluding the greatest max and the greatest product, the five fusion strategies averagely perform better in varied evaluation metrics, and remarkably, five fusion measures are most accurate amongst the best four methods in terms of accuracy.

On the other hand, the fusion methods are also comparable in Set 1. The greatest mean selector still acquires the best in four metrics, which are Accuracy, FPR, precision and RMSE. The second best fusion method is OWA. It also performs well and its performances in all metrics are slightly lower than the greatest mean in this fusion methods comparison. However, the greatest max and greatest product selectors seem to be the worst two out of the seven fusion strategies, even comparing to individual classifiers, these two are not ideal solutions to merge data.

Table 5.3: Performance Evaluation of the fusion methods

Algorithm	ACC	TPR	FPR	PRE	REC	F1	RMSE
1-Borda	0.7204±0.0132	0.7545±0.0448	0.3129±0.0421	0.7079±0.0281	0. 7545±0. 0448	0.7289±0.0194	1.0365±0.0247
1-OWA	0.7214±0.0155	0.7522±0.0523	0.3094±0.0460	0.7099±0.0275	0. 7522±0. 0523	0.7287±0.0245	1.0032±0.0247
1-Max	0.6967±0.0155	0.7358±0.0597	0.3421±0.0542	0.6843±0.0285	0. 7358±0. 0597	0.7069±0.0259	1.0763±0.0282
1-Mean	0.7233±0.0146	0.7524±0.0521	0.3059±0.0444	0.7123±0.0262	0. 7524±0. 0521	0.7302±0.0243	0.9919±0.0254
1-Product	0.6981±0.0162	0.7471±0.0534	0.3509±0.0469	0.6815±0.0254	0. 7471±0. 0534	0.7112±0.0253	1.0204±0.0161
1-Voting	0.7204±0.0132	0.7547±0.0447	0.3132±0.0423	0.7078±0.0284	0. 7547±0. 0447	0.7290±0.0195	1.0545±0.0238
1-MIR	0.7204±0.0132	0.7545±0.0448	0.3129±0.0421	0.7079±0.0281	0. 7545±0. 0448	0.7289±0.0194	1.0365±0.0247

5.4.2 Multi-model with the Three Unrelated-Classifier Fusion

As stated at the beginning of this chapter, after testing the fusion of all classifiers, the results of the experiment (Set 1) have been verified to confirm that the fusion scenario is effective for improving the performance. Due to the large scale of calculation while fusing the nine classifiers, we provide two alternative approaches to use less classifiers and also maintain the advantages of the fusion methods. These two scenarios consist of selecting the most three unrelated and the three best-performance classifiers.

The research by H. Naderi et al. [12] indicated that despite the different kinds of recognition systems, the recognition rates are uneven, and the recognition results of the systems can also be fused. After fusion, the final result yielded better capability than any individual classifier within the combination. Inspired by the research, an assumption is made that selecting the classifiers that are not associated with one another. To figure out this issue, a statistical method involved with this case is computing the correlation between the outputs given by the nine classifiers. The correlation-coefficient measures the dependence or correlation between classifiers. Let r be the correlation-coefficient, so it can be interpreted by Equation (5.1) as [26]:

$$r_{xy} = \frac{\sum_i \left((x_i - \bar{x})(y_i - \bar{y})\right)}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}} \tag{5. 2}$$

where x and y are two comparable variables, denoted by two classifiers in this study, and i represents the number of instances in the testing set.

Since one calculation of correlation-coefficient only takes two classifiers into account, all the nine classifiers should make pairwise comparisons by computing the correlation multiple times. The correlation-coefficient results for the nine classifiers are shown in Table 5.4.

Table 5.4: Correlation-Coefficient of the nine classifiers

	NB	**MNB**	**BNB**	**SVM_ linear**	**SVM_ rbf**	**DT**	**KNN**	**LR**	**RF**
NB	1.0000	0.9793	0.9882	0.8894	0.8778	0.6832	0.6485	0.9353	0.7789
MNB		1.0000	0.9933	0.9111	0.8949	0.6872	0.6446	0.9677	0.7898
BNB			1.0000	0.8994	0.8893	0.6801	0.6668	0.9611	0.7837
SVM_linear				1.0000	0.9610	0.7037	0.6132	0.9424	0.7816
SVM_rbf					1.0000	0.6841	0.6183	0.9278	0.7741
DT						1.0000	0.6242	0.7023	0.8933
KNN							1.0000	0.6647	0.7244
LR								1.0000	0.8043
RF									1.0000

The correlation-coefficient reflects the similarity between two objects with the threshold from 0 to 1; the higher value of the two classifiers, the closer their similarity. For example, Naïve Bayes and its two variations have earned the correlation values quite close to 1.00. By Comparing all pairs of the algorithms, it can be noted that three algorithm pairs which are NB-DT, NB-KNN and DT-KNN merely acquired lower correlation values: 0.6832, 0.6485 and 0.6242, respectively. It seems that Naïve Bayes, Decision Trees and K-Nearest Neighbors are

the three most irrelevant algorithms that can be used for contributing the multi-model, as they are the three most unrelated classifiers.

The fusion of the three most unrelated classifiers is noted as Set 2 for which its seven evaluation metrics are displayed in Table 5.5.

Table 5.5: The evaluation metrics of two alternative scenarios. (The combination of the most three unrelated classifiers show as Set 2, the group of the three best-performance classifiers show as Set 3.)

Algorithm	ACC	TPR	FPR	PRE	REC	F1	RMSE
2-Borda	0.7052±0.0199	0.7615±0.0259	0.3511±0.0188	0.6843±0.0181	0.7615±0.0259	0.7208±0.0201	1.0668±0.0371
2-OWA	0.7048±0.0171	0.7610±0.0254	0.3512±0.0144	0.6840±0.0163	0.7610±0.0254	0.7203±0.0180	1.0315±0.0310
2-Max	0.6973±0.0166	0.7553±0.0248	0.3607±0.0131	0.6767±0.0162	0.7553±0.0248	0.7137±0.0179	1.0718±0.0297
2-Mean	0.7086±0.0153	0.7605±0.0230	0.3433±0.0124	0.6889±0.0149	0.7605±0.0230	0.7228±0.0162	1.0198±0.0287
2-Product	0.6948±0.0156	0.7595±0.0232	0.3699±0.0105	0.6723±0.0148	0.7595±0.0232	0.7132±0.0168	0.9732±0.0238
2-Voting	0.7052±0.0199	0.7615±0.0259	0.3511±0.0188	0.6843±0.0181	0.7615±0.0391	0.7208±0.0201	1.0829±0.0378
2-MIR	0.7052±0.0199	0.7615±0.0259	0.3511±0.0188	0.6843±0.0181	0.7615±0.0259	0.7208±0.0201	1.0668±0.0371
3-Borda	0.7223±0.0163	0.7715±0.0209	0.3265±0.0251	0.7028±0.0197	0.7715±0.0209	0.7353±0.0147	1.0435±0.0311
3-OWA	0.7192±0.0166	0.7744±0.0201	0.3357±0.0267	0.6978±0.0214	0.7744±0.0201	0.7338±0.0154	1.0053±0.0312
3-Max	0.7194±0.0165	0.7754±0.0209	0.3363±0.0255	0.6976±0.0213	0.7754±0.0209	0.7342±0.0159	1.0084±0.0318
3-Mean	0.7222±0.0151	0.7742±0.0206	0.3295±0.0213	0.7015±0.0193	0.7742±0.0206	0.7358±0.0152	0.9978±0.0296
3-Product	0.7220±0.0152	0.7740±0.0205	0.3297±0.0217	0.7013±0.0195	0.7740±0.0205	0.7356±0.0152	0.9579±0.0274
3-Voting	0.6223±0.0163	0.7715±0.0209	0.3265±0.0251	0.7028±0.0197	0.7715±0.0209	0.7353±0.0147	1.0511±0.0315
3-MIR	0.7223±0.0163	0.7715±0.0209	0.3265±0.0251	0.7028±0.0197	0.7715±0.0209	0.7353±0.0147	1.0435±0.0311

5.4.3 Multi-model with the Three Best-performance Classifiers Fusion

Table 5.6: The rank average of individual classifiers

Algorithm	Rank Avg.
NB	4.86
MNB	4.00
BNB	2.86
SVM_linear	5.29
SVM_rbf	5.00
DT	7.00
KNN	7.71
LR	2.71
RF	5.57

Instead of merging the unrelated classifiers, this experiment applies a classical selection strategy, which is selecting the best three classifiers amongst the nine. Based on the evaluations of the individual classifiers, the rule for choosing the best three is according to their ranks in metrics on average. The average has been presented in Table 5.6, the lower value means the higher the rank the classifier got. Logistic Regression, Bernoulli NB and multinomial NB are the best three algorithms, with rank average 2.71, 2.86 and 4.00, respectively. Nevertheless, Naïve Bayes series has the same origin of the theorem of Bayes. So, they may present bias if the model is made up of two highly similar methods out of the three methods. For this reason, the multinomial Naïve Bayes that occupied the third rank is replaced by the contrasting SVM with RBF kernel. Accordingly, the three classifiers are applied to the multi-model as the third scenario. The 10-fold cross validation for evaluating the model was re-done and the results are shown in Table 5.5 as Set 3.

By comparing these two scenarios in all the metrics, it can be seen that the seven fusion algorithms perform better under the three best-performance strategy than they do in the three unrelated ones. However, the combination of three unrelated classifiers seems to be not an effective choice when combining a small set of classifiers.

5.5 The Classification Performance Comparison throughout the Three Alternatives

To compare the individual classifiers and also the three strategies for the multi-model, an expanded version of the previous rank table (Table 5.2) is presented in Table 5.7.

There are three groups of experiments resulting from the three fusion alternatives marked as Set 1 to 3 respectively in the ranking Table 5.7. It can be clearly seen that the benefits of applying data fusion to the multi-model are apparent as the individual classifiers alone have worse but more fluctuated rankings as opposed to the rankings of multi-model algorithms, excluding Logistic Regression.

For the three fusion alternatives (Set1 to 3), it can be seen that the combination of the three best classifiers gives the best results and outstanding ranks in the metrics. The fusion of all available classifiers takes the second best fusion alternative. However, the three unrelated-classifier fusion seems to be the worst one in the three alternatives. Even some methods in this fusion perform worse results than the individual classifiers. It can be summarized that merging massive data is not as good as merging accurate or appropriate information. Also, combining different or unrelated types of information might not be a wise choice since the divergence will weaken the advantage of fusion.

Table 5.7: The total evaluation metrics rank. (There are three sets for the multi-model with different options: Set 1 for combining all nine classifiers; Set 2 for the three most unrelated classifiers and Set 3 for the three best-performance classifiers.)

Algorithm	ACC	TPR	FPR	PRE	REC	F1	RMSE
NB	16	10	19	18	10	10	30
MNB	15	28	2	2	28	25	6
BNB	14	9	18	16	9	8	9
SVM_linear	17	30	1	1	30	29	13
SVM_rbf	23	1	30	29	1	16	16
DTs	29	26	29	30	26	28	28
KNN	30	29	15	27	29	30	29
LR	8	21	5	8	21	11	4
RF	24	27	9	17	27	27	15
1-Borda	9	19	6	5	19	13	17
1-OWA	7	23	4	4	23	15	7
1-Max	27	25	20	20	25	26	26
1-Mean	1	22	3	3	22	9	3
1-Product	25	24	22	25	24	24	12
1-Voting	11	18	8	7	18	12	22
1-MIR	9	19	6	5	19	13	17
2-Borda	19	11	23	21	11	18	23
2-OWA	22	14	26	24	14	21	14
2-Max	26	17	27	26	17	22	25
2-Mean	18	15	21	19	15	17	11
2-Product	28	16	28	28	16	23	2
2-Voting	19	11	23	21	11	18	27
2-MIR	19	11	23	21	11	18	23
3-Borda	2	6	10	9	6	3	19
3-OWA	13	3	16	14	3	7	8
3-Max	12	2	17	15	2	6	10
3-Mean	5	4	13	12	4	1	5
3-Product	6	5	14	13	5	2	1
3-Voting	2	6	10	9	6	3	21
3-MIR	2	6	10	9	6	3	19

Furthermore, not all fusion methods get satisfactory results in the experiments; namely, the fusion methods are still comparable due to their performance not being equivalent at all. The method of the greatest mean yields the best in each fusion alternative on average. In addition, the greatest mean in the fusion of three best-performance algorithms (Set 3) is the best one with the highest average ranking in the entire comparison. On the contrary, the greatest max method yields the worst result in the three sets. The greatest product is the second worst method in both Set 1 and Set 2, although it has a remarkable enhancement when it is implemented in Set 3. It seems that the greatest product is more sensitive when the selected classifiers' performances are uneven. On the other hand, there is a notable phenomenon occurring in all fusion alternatives: Borda count and MIR always acquire exactly the same value in every evaluation metric. Considering this situation, and after reviewing other studies, this phenomenon could be explained by the following two reasons. First, these two fusion methods are based on rank, so their determinations are easily influenced by the rank. Also in the study presented in [12], the authors indicate that rank-based methods are effective when the number of classes is large. Second, according to their expression interpretations, the combined matching scores they calculated change in the same level. In this case, since the experiments only have two classes (positive and negative), the performances of Borda count and MIR are still not poor in regards to each fusion group. Besides, the majority voting also has similar behaviour to Borda count or MIR. Additionally, OWA, the method that assigns weights to classifiers according to their prediction confidence, expected better performance in the experiments but yielded mediocre ranking in the table. It could be inferred that this method performs unsatisfactory results when incorrect predictions with high confidence are assigned with heavy weights.

Throughout the comparison of individual classification algorithms versus the multi-model, merging the algorithms gives much better performance improvements. Also, when building a data fusion model, the technique that extracts minor variations (by taking an average) seems to be an appropriate method; if the selected algorithms show uneven performances, Borda count or MIR might be better than calculating the product.

Chapter 6

Conclusions and Future Work

6.1 Summarizations

In this research, three main objectives are studied: firstly, nine commonly used machine learning algorithms (Naïve Bayes, Multinomial NB, Bernoulli NB, SVM with linear kernel, SVM with RBF kernel, decision tree, K-nearest neighbours, logistic regression and random forest) are involved in the performance comparison; secondly, the multi-model approach has been proposed for improving the classification performance by merging classifiers' results; thirdly, we discussed the improvement of multi-model by a comparison between individual algorithms and the multi-model.

To evaluate the algorithm's classification performance, instead of using one or two standard (mostly using the accuracy), seven evaluation metrics (accuracy, FPR, TPR/recall, precision, F1 measure and RMSE) are applied to estimate the classifier in different aspects, providing robust evaluation results. In our experiment, with a two classes problem, a dataset with ten thousand tweets is used to train and test the algorithms. This experiment shows that logistic regression is the best one in the group of algorithms. Naïve Bayes and its variations are easy-to-use classifiers because of the simple calculation with a hypothesis of conditional independency. These three classifiers also perform well in the bi-class problem (Bernoulli NB performs better than other two classifiers). SVM with two kernel functions shows unstable performances in different metrics (they may perform the best in one but get the worst ranking in another).

Subsequently, this research proposed the multi-model approach with three alternatives: merging all the classifiers, carefully selecting the three best-performance classifiers and selecting three unrelated classifiers. From the experiments, it seems that the majority of fusion methods in the multi-model with only merging the best three individual classifiers gained the best on average rather than other alternatives especially better than fusing all the classifiers' outputs. On the contrary, the metrics in the three unrelated classifiers combination yielded unsatisfactory performance, some of them even worse than individuals.

In the third experiment, we also addressed the comparison between the fusion methods. The greatest max method performs worst in all fusion methods. Borda Count and MIR show their good capability in classifying two class instances although the literature [12] indicated that these rank-based methods are good at handling multi-class problem. The method of the greatest product, incredibly yields higher performances in the three best-performance set in spite of it performs terribly in other two sets. Finally, the greatest mean shows its outstanding results in each set of fusion alternatives, becoming the best and the most appropriate fusion method for this two-class problem.

Consequently, the idea of fusing different algorithms' results improves the classification performance in certain level. In our study, the multi-model with the strategy of the greatest mean by fusing the three best-performance classifiers' results has turned out to be a better and appropriate approach for classification rather than applying single machine learning algorithms.

6.2 Contributions

First of all, the basic sentiment classification model has been constructed for two-class problem by using Python. This system not only have capability for sentiment classification, but also contains nine selective machine learning algorithms which are trained and tested by the same corpus. As a result, for each test instance, the system offers multiple classification results depending on the choices of the algorithms. Besides, the classifiers are being tested by a ten-fold cross validation mechanism.

Secondly, for evaluating the performance across the classifiers and also against the multi-model, an evaluation model has been made, which consists of six different evaluation metrics (TPR and recall are the same concept). This model collects the classification results of the classifiers and the outputs of the multi-model; and then returns the evaluation results for each object by statistical calculations and stores them in an Excel file.

For improving the performance of classification, we proposed a multi-model which merges the outputs of individual classifiers by fusion methods. In the multi-model, we experimentally involve seven comparable fusion methods in order to study and figure out the best method for fusing the classification results.

Finally, we made a comparison of all the classifiers and fusion methods in multi-model based on multiple experiments. Through comparing and analyzing the results of evaluation, we found that the multi-model certainly improves the classification performance much better than the individual classifiers, especially when using the greatest mean as the fusion strategy to merge the three best-performance classifiers, thus, the multi-model yields the best results in the comparison.

6.3 Future Research

In the experiment, we only use a tweet dataset that is provided by Natural Language Toolkit (NLTK). Since there are so many reference datasets for Twitter sentiment analysis, we found one Twitter sentiment corpus provided by Sanders Analytics[16], which is widely referred in various research. Thus, one perspective work is applying the Twitter corpus to our multi-model as the alternative datasets for training and testing.

Secondly, due to the tweets dataset is the only corpus that was used for training and testing the classifiers, which means the experiment pays more attention in classifying short text than longer ones. For these tweets, it has been mentioned previously that the tweets are collected randomly from Twitter so they may cover various topics in this case. However, there is a hypothesis that the multi-model may perform differently when classifying the contents in specific areas or topics. To investigate this hypothesis, the multi-model should run with massive different areas of tweets, and concentrate on the irregular evaluation metrics results or try to find specific patterns for different kinds of topic. This attractive orientation can be researched more in the future.

Additionally, some machine learning algorithms may perform differently between two classes and multiple classes problems. As a result of two-class problem in the experiment, exploiting the multi-model dealing with tri-class even multi-class problems and evaluating its performance is also a potential topic for further research.

[16] Sanders Analytics: http://www.sananalytics.com/lab/twitter-sentiment/

Appendix A

The Ranking of comparing the individual algorithms and the multi-model

The following table presents the comparison between the individuals and the multi-model with the strategy of merging all the classifiers. Specifically, *Ranks Average* (Ranks Avg.) is the mean of all the ranks that the classifier or fusion method got.

Table A.1: Ranking of individuals and multi-model with fusing all the classifiers

Algorithm	Accu	TPR	FPR	Prec	Rec	F1	RMSE	Ranks Avg.
NB	9	3	12	11	3	3	16	8.14
MNB	8	14	2	2	14	11	3	7.71
BNB	7	2	11	9	2	1	5	5.29
SVM_linear	10	16	1	1	16	15	7	9.43
SVM_rbf	11	1	16	15	1	9	9	8.86
DTs	15	12	15	16	12	14	14	14.00
KNN	16	15	10	14	15	16	15	14.43
LR	3	7	5	8	7	4	2	5.14
RF	12	13	9	10	13	13	8	11.14
1-F_Borda	4	5	6	5	5	6	10	5.86
1-F_OWA	2	9	4	4	9	8	4	5.71
1-F_max	14	11	13	12	11	12	13	12.29
1-F_mean	1	8	3	3	8	2	1	3.71
1-F_product	13	10	14	13	10	10	6	10.86
1-F_voting	6	4	8	7	4	5	12	6.57
1-F_MIR	4	5	6	5	5	6	10	5.86

Appendix B

Experimental Configuration

In our study, the sentiment classification system with the multi-model is developed by using Python. To implement natural language processing techniques and also apply machine learning algorithms, the system also associates with two powerful platforms for Python: a natural language processing tool called Natural Language Toolkit (NLTK)[17]; and also, a machine learning platform on Python: scikit-learn[18].

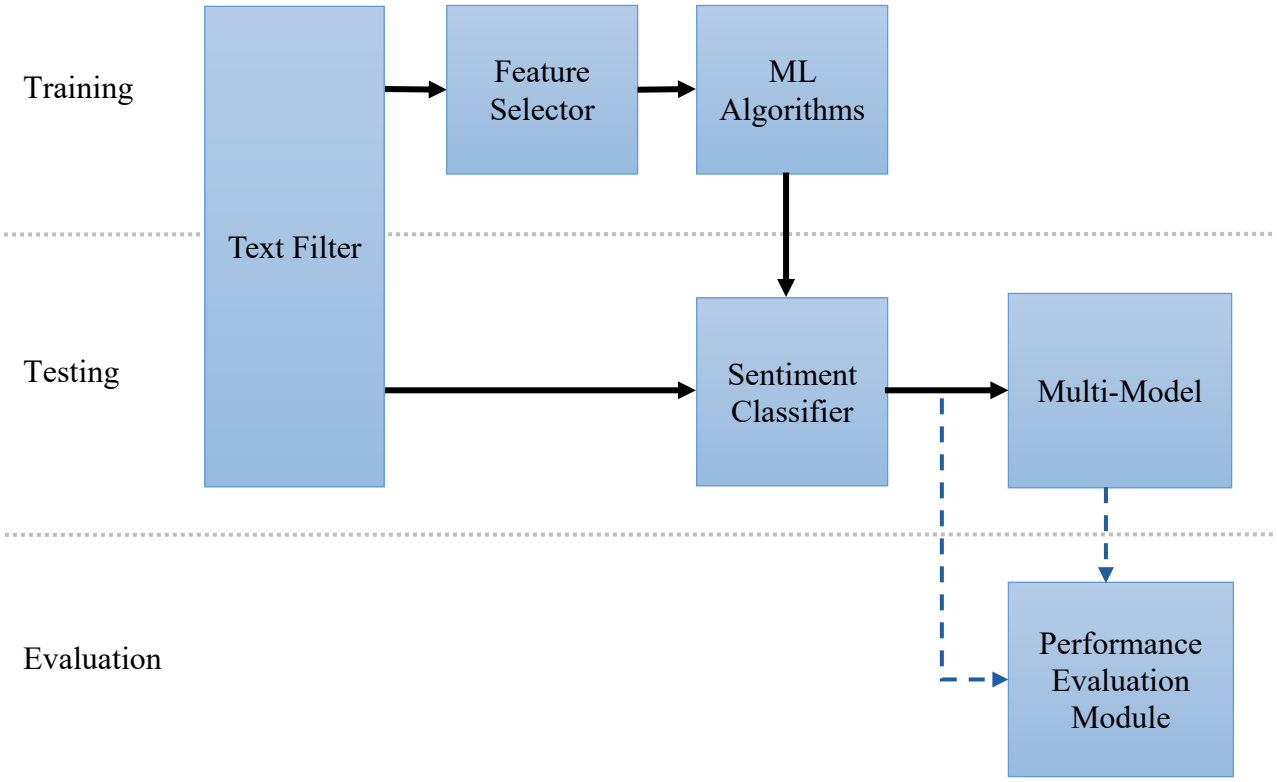

Figure B.1: Configuration of the Multi-Model Classification System

[17] Natural Language Toolkit (NLTK): http://www.nltk.org/
[18] scikit-learn: http://scikit-learn.org/

The sentiment classification consists of two parts: training part and testing part. In training part, first the raw labeled text data is pre-processing by a text filter, which contains several NLP techniques like tokenization, word stemming, stop-word removal, and POS (part-of-speech) tagging. The second step is feature selection. The pre-processed text data is refined as features in this stage. We apply two strategies for extracting features: finding sentiment words by using a sentiment dictionary[19] and calculating the TF (terms frequency) to find the most informative words. Thus, a feature set is generated after the selecting stage. Third, the feature set is sent to the selected ML algorithm individually, and then these algorithms will be trained as their corresponding classifiers by using the same knowledge.

On the other hand, when the classifiers are generated, they are capable for classifying the text materials. The testing part is literally the classifying part, in our sentiment classification system there are two data sources: testing data and real world data. The former comes from NLTK, there are numbers of labelled tweets; the later comes from twitter stream, we can directly grab fresh data to our system. Although we are more concentrated on the former one. These incoming data also be processed by the text filter, then they are classified by the classifiers that we constructed in the previous part.

Our proposed multi-model is implemented after classification by individual classifiers. The model will collect the classification results from different classifiers, and use a specific fusion method merging the classification results. Thus, the final decision for an instance has been made by the multi-model after calculations.

[19] SentiWordNet: http://sentiwordnet.isti.cnr.it/

To investigate the performance of the algorithms and the multi-model. We also built an evaluation module, which contains several evaluation metrics for estimating the algorithm's performance. When evaluate the ML algorithms' performance, the evaluation module is configured behind the testing part. In order to measure the performance of the fusion method (which is inside the multi-model), the module should be deployed behind the multi-model.

Bibliography

[1] W. Medhat, A. Hassan, and H. Korashy, "Sentiment analysis algorithms and applications: A survey", *Ain Shams Engineering Journal*, 5(4), 2014, pp. 1093-1113.

[2] T. Wilson, J. Wiebe, and P. Hoffmann, "Recognizing contextual polarity in phrase-level sentiment analysis", *Proceedings of the conference on human language technology and empirical methods in natural language processing*, Association for Computational Linguistics, 2005, pp. 347-454.

[3] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques", *Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, Vol. 10, Association for Computational Linguistics, 2002, pp. 79-86.

[4] A. Pak, and P. Paroubek, "Twitter as a Corpus for Sentiment Analysis and Opinion Mining", *LREc*, Vol. 10, 2010, pp. 1320-1326.

[5] C. Whitelaw, N. Garg, and S. Argamon, "Using appraisal groups for sentiment analysis", *Proceedings of the 14th ACM international conference on Information and knowledge management*, ACM, 2005, pp. 625-631.

[6] M. Ghiassi, J. Skinner, and D. Zimbra, "Twitter brand sentiment analysis: A hybrid system using n-gram analysis and dynamic artificial neural network", *Expert Systems with applications*, 40(16), 2013, pp. 6266-6282.

[7] P. D. Turney, "Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews", *Proceedings of the 40th annual meeting on association for computational linguistics*, Association for Computational Linguistics, 2002, pp. 417-424.

[8] H. Siqueira, and F. Barros, "A feature extraction process for sentiment analysis of opinions on services", *Proceedings of International Workshop on Web and Text Intelligence*, 2010.

[9] J. Lafferty, S. Della Pietra, and V. Della Pietra, "Statistical learning algorithms based on bregman distances", 1997.

[10] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features", *European conference on machine learning*, Springer Berlin Heidelberg, 1998, pp. 137-142.

[11] N. Japkowicz, and M. Shah, *Evaluating learning algorithms: a classification perspective*. Cambridge University Press, 2011.

[12] H. Naderi, B. Haji Soleimani, S. Matwin, B. Nadjar Araabi, H. Soltanian-Zadeh, "Fusing Iris, Palmprint and Fingerprint in a Multi-Biometirc Recognition System", n.d.

[13] H. Zhang, "The optimality of naive Bayes. " *AA*, 1(2), 2004, p.3.

[14] A. McCallum, and K. Nigam, "A comparison of event models for naive bayes text classification", *AAAI-98 workshop on learning for text categorization*, Vol. 752, 1998, pp.41-48.

[15] V. Metsis, I. Androutsopoulos, and G. Paliouras, "Spam filtering with naive bayes-which naive bayes?", *CEAS*, 2006, pp. 27-28.

[16] C. Cortes, and V. Vapnik, "Support-vector networks", *Machine learning* 20(3), 1995, pp. 273-297.

[17] I. Guyon, B. Boser, and V. Vapnik, "Automatic capacity tuning of very large VC-dimension classifiers", *Advances in neural information processing systems*, 1993, pp. 147-147.

[18] C. W. Hsu, C. C. Chang, and C. J. Lin, "A practical guide to support vector classification", 2003, pp. 1-16.

[19] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features", *European conference on machine learning*, Springer Berlin Heidelberg, 1998, pp. 137-142.

[20] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*, Cambridge University Press, 2008.

[21] D. M. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation", 2011.

[22] R. Caruana, and A. Niculescu-Mizil, "Data mining in metric space: an empirical analysis of supervised learning performance criteria", *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2004, pp. 69-78.

[23] M. J. Procopio, *An experimental analysis of classifier ensembles for learning drifting concepts over time in autonomous outdoor robot navigation*, ProQuest, 2007.

[24] R. Snelick, U. Uludag, A. Mink, M. Indovina, and A. Jain, "Large-scale evaluation of multimodal biometric authentication using state-of-the-art systems", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3), 2005, pp. 450-455.

[25] B. Pang, and L. Lee, "Opinion mining and sentiment analysis", *Foundations and trends in information retrieval*, 2(1-2), 2008, pp. 1-135.

[26] D. Shen, and Z. Lu, "Computation of correlation coefficient and its confidence interval in SAS", *SUGI: Paper 170-31, SUGI 31 Proceedings,* San Francisco, CA, March 26, 2006, p.29.

[27] C. C. Aggarwal, and C. Zhai, *Mining text data*, Springer Science & Business Media, February 3, 2012.

[28] L. Doug, "3D data management: Controlling data volume, velocity and variety", *META Group Research Note*, February 6, 2001, pp.70-74.

[29] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision", *CS224N Project Report*, Stanford, 1, 2009, p.12.

[30] J. Read, "Using emoticons to reduce dependency in machine learning techniques for sentiment classification", *Proceedings of the ACL student research workshop*, Association for Computational Linguistics, 2005, pp. 43-48.

[31] M. Z. Asghar, A. Khan, S. Ahmad, and F. M. Kundai, "A review of feature extraction in sentiment analysis", *Journal of Basic and Applied Scientific Research*, 4(3), 2014, pp. 181-186.

[32] A. Danesh, B. Moshiri, and O. Fatemi. "Improve text classification accuracy based on classifier fusion methods", *Information Fusion, 2007 10th International Conference on*, IEEE, 2007, pp. 1-6.