Abstract

This project is about text mining of review domain. The data set is derived from the information of international conference paper. Those scientific papers are evaluated by the academic peer reviewing process, and the corpus we used is derived from the reviewers' comments. After reading academic papers, reviewers give their opinions and also score or judgements (e.g. good, neutral, bad) for the paper. The research papers submitted to international conferences are typically reviewed by three Program Committee Members. Each review from those members contains several textual parts and a rating. Our approach is to use the ratings as target and to train a variety of classifiers to predict these targets based on textual features.

A challenge work for this project is how to take opinion extraction or features selection. The reviews of research paper presented scientific rigour, which means some distinguished words representing human's mood (such as "awesome", "rubbish") do never appear in academic reviews. This is very different from traditional reviews such as movie or e-products. For further details of sentiment expression or semantic representing, we used some advanced techniques, n-gram, part-of-speech (POS) [1] and also English sentences parser [20] for opinion extraction process.

In fact, we not only treat this project as classification problem, but also it is a ranking problem. The distribution of ratings is from score 1 to 6. As a matter of fact, we are able to deal with these scores as ranking relevant coefficient. Thus, there are three main machine learning algorithms we employed for traditional classification problem, Naive Bayes (NB), Decision Tree (DT) and Support Vector Machine (SVM). Also, we extended NB to Multinomial Naive Bayes as an independent classifier. In order to improve the accuracy of NB classifiers, we tried different weights for argmaxf(w, s) and evaluated them by taking Receiver Operating Characteristic (ROC) analysis, an intuitive technique for visualising and selecting value based on their performance.

Then, we transferred the classical classification problem into ranking problem and trained a ranker by using Ranking SVM. This method offers us more information about the corpus and it is easier than regression problem due to there is no distance calculation in ranking problem. What we need is to set a threshold or thresholds to determine a label for each testing instance. The evaluation of the threshold is made through the use of ROC analysis.

Finally, due to there were multiple algorithms which were required to be compared, we used Friedman's test [14, 10] instead of t-test. Friedman's test is able to state whether there is significant difference among algorithms. After Friedman's test and if the null hypothesis is rejected, we can process with the post-hoc Nemenyi test [24] to find out which classifier is differ. Additionally, we also employ T-test [29] to state whether weight learning does really work in our case.

Acknowledgements

I would like to express my deep gratitude to my supervisor Professor Peter Flach for proposing the topic of this project, and appreciate for his comments and continuous help throughout this project.

I would like to thank Elena Hensinger and Ilias Flaounas for their valuable hints.

Finally, I would like to thank my classmates and friends for their friendship, and also want to express my thanks to my parents and girlfriend, for their encouragement.

Table of Contents

1. Introduction

The explosion of information age is pushing the process of academic and industrial field. People are disenchanted in exterior information feedback. Machine learning and data mining play a significant role at present. By using those artificial intelligent techniques, one hand, we make use of machine to implement human's work. On the other hand, through relevant machine learning algorithms we are able to capture extra information. This is a very popular topic in computer science domain. More and more researchers are interested in mining implicated contents of expression by semantic or sentiment analysis by using these techniques.

1.1 Peer Review Process

Every successful publishment of a research papers must be passed through this process in order to guarantee its quality. The peer review process is able to assure that those fallacious or poor papers will be not published in the academic conference. It is also helpful to improve reputation in research domain. The way academic peer review works is that an author writes an article and sends it to a journal for review. Then, there are three or more scholars to review the article after reading it over. In this final project, each group is set as three people. These scholars in a group select an article to read and comment on it. The reviewers are chosen based on whether they have a good reputation in the specific field of the article or whether they are mentioned in the bibliography, or they could be suggested by authors sometimes. Once a list of reviewers is drawn up, the name of the author is removed by editor from the manuscript.

The editor makes decision about the manuscript when reviewers send back their comments: (1) Is it to be accepted? (2) Is it to be accepted but still need modifications? (3) Is it to be rejected? If the manuscript is accepted but required some modifications, the author has to make changes until the editor is satisfied. Then, the manuscript is published eventually. This is a rigorous process that from a manuscript submission to its publication in an academic journal eventually, it takes from six months to over a year. The whole process ensures the quality of each article and there is an iteration process in the main steps. The detailed procedure is described as in *Figure 1*.

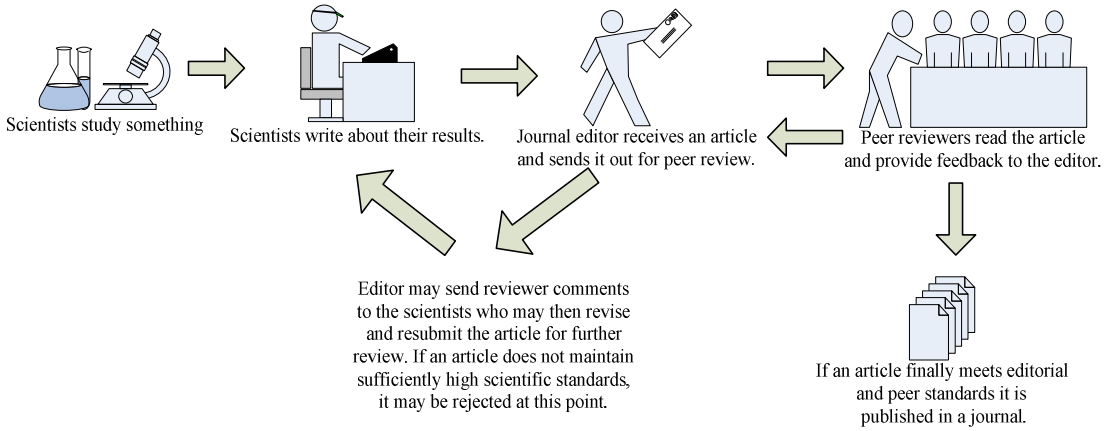


Figure 1: a primary life cycle of Peer Review Process

In our case, these three reviewers will give a brief summary of the main contributions of the paper, three strong and weak points of the paper, also if the paper has an experimental section, these three reviewers will write a comment on the repeatability of the experiments, and so on. Then, the most important is the detailed comments and an overall recommendation score among one to six. Our task is using the relevant or detailed comments to predict its overall recommendation score.

1.2 Aims and Objectives

This project is about text mining in reviews of international conference paper. The whole data set is given by project supervisor. Our motivation of this project is to use supervised machine learning techniques to improve on these handcrafted methods that classify reviews by human's mind, based on the data set of reviews of research papers submitted to a major international conference. Our primary aim of this project is to use the ratings as labelled targets and to train a variety of classifiers to predict these targets from textual features.

The main objective of this work is to use some advanced opinion extraction techniques to build a better model and also try to improve the performance of classifiers using state of art ROC analysis. The basic steps throughout the project are:

- Build a variety of training sets by opinion extraction from the data set.
- Train a variety of classifiers by using machine learning algorithms.
- Improve the performance of Multinomial Naive Bayes through weight learning by ROC analysis.
- Transfer the Classification problem into Ranking problem initially, and learn a threshold within the results by ROC analysis that is able to transfer the ranking problem back to Classification problem.
- Evaluate the classifiers by t-test and Friedman's test and also carry out post hoc Nemenyi test after Friedman's test.

The success of the project depends on whether the accuracy of classifier is higher than the baseline defined later, meanwhile identifying the best approaches.

1.3 Structure of Dissertation

The introduction stated the related information about the project topic and also presented a brief procedure of this project. In the following content, we introduce the organisation of this dissertation.

Chapter 2 describes the essential background knowledge about texting mining such as the techniques of Natural Language Processing (NLP) that we employed for opinion extraction and feature selection and the basic concepts of machine learning algorithms which we will use to train a variety of classifiers, introduces the inside structure of our data set and our workbench, and illustrates previous work which are related with our project.

Chapter 3 only focuses on pure classification algorithms, such as Naive Bayes, SVM, and Decision Tree. As a matter of fact, the performance of Naive Bayes algorithm is not very sensitive for the data set, but it is interesting that Multinomial Naive Bayes outperforms the other traditional classification algorithm. This point we will discuss in this chapter.

Chapter 4 describes an optimal factor, so-called weight learning, which is able to improve the accuracy of the Multinomial Naive Bayes classifier. We start from binary classification to multi-class classification. However, due to finding a global optimum in weight space is in NP-complete [8], we present Lachiche and Flach's algorithm [18] for producing an optimal solution. Indeed, we stop at three classes since time limitation and this is enough to state the optimisation problem.

Chapter 5 describes an interesting solution to address this classification problem. Since we use ratings as targets and the ratings of documents are from 1 to 6, it is able to treat it as a ranking problem at start. The rest thing we should consider is to generate an optimal threshold to make judgements. Thus, it returns to our classification problem.

Chapter 6 is about the experimental evaluation. We employs Friedman's test for multiple data sets [13]. After Friedman's test and if the null hypothesis is rejected, we can process with the post-hoc Nemenyi test to find out which classifier is differ. Furthermore, T-test is also taken to state whether the optimal process of weight learning does work.

Chapter 7 is a conclusion of our project.

Chapter 8 proposes possible ideas for future work.

2. Background

In this Chapter, we introduced the related background information about our project. The main contents include a description of what is the NLP techniques we used for modelling in this project; three mainstream classification algorithms that are NB, SVM and DT; ROC analysis for optimisation problem; a more detailed observation in our data set; also, some relevant previous work already done in this text mining area. Indeed, we only focus on those previous research papers which are about in review domains. This is more helpful for us to make a horizontal reference. Later, following the idea of those researchers, we focus on the opinion extraction and features selection by using NLP we introduced before in this project of conference papers reviews. This part plays a significant role in data mining because building up an efficient model affects the accuracy directly for the classifiers. In this chapter, we also illustrate the implemental environment such as programming toolkits and other applications.

2.1 Natural Language Processing

NLP is a main branch of artificial intelligence [7]. In theory, NLP is an attractive interaction between human and machine. Modern NLP techniques are based on machine learning. It is related to linguistics, computer science, and mathematics. When users input the reviews in text area, they are expressing natural language, and our task is how to deal with the language to mine the objective information in the language.

In this section, we prefer to introduce three advanced techniques of NLP: they are part-of-speech tagging [1], N-gram [4] and English Parser [20]. These techniques offer a path to extract opinion from reviews context in textual area, which is able to build up a more efficient model in order to achieve a higher accuracy by the features generated from them.

Part-Of-Speech: POS is a process of finding those phrases, sentence, or paragraph which corresponding to a particular grammatical pattern. A grammatical pattern contains some elements that are nouns, verbs, adjectives, adverbs, etc. We define the POS rules firstly to extract those words which satisfy the rules in relevant reviews. For example, a pattern of single word could be adjectives. Normally, adjectives are able to express human's emotion or objective opinion. Thus, the words in data set could be tagged by POS tagger first. The new features are produced with the characteristic or property of a certain word. Furthermore, also a pattern of two-words could be adjectives+nouns. The combination of property constructs a new phrase feature instead of single feature. Indeed, this method is widely implemented in text mining. A sentiment expression is based on words, and every word has its own property. Thus, we are able to decompose a sentence by POS to extract useful information efficiently and directly. However, the disadvantage we have to consider is that natural language is very complex. The sentiment expression sometimes is not the same meaning as it looks like. One of reasons is the ambiguity of natural language happens in different occasions. Another problem should be noticed is that Porter Stemming technique must be used carefully. If carrying out stemming operation in preprocessing stage, the characteristic or

property of a certain word will be totally changed. For instance, Porter Stemming technique will transfer “stemming”, “stemmer”, even “stem” itself into a reduced root format – “stem”, which results in all forms of a certain word are changed to be a unique form, no matter what base or root form. In fact, the way of POS working is to distinguish the characteristic or property for every different word forms, so stemming processing operation is unexpected for our original idea.

N-gram: N-gram approach is another essential part for features selection in text mining task. The same thing with POS is that it has a pattern as well and the information we extracted must follow the pattern, for example, unigram, bigram etc. But the main difference with POS is that N-gram is not based on language grammar strictly. It is a subsequence of a given sequence, and the N determines how long the subsequence is. For instance, unigram is size 1, bigram is size 2, and more than size 3 are so-called n-gram. An advantage of n-gram is that features can be presented as phrase form which is able to capture the context. For instance, a bigram includes a negation word like “not”, and of course, an expression with and without the negation is totally different. However, a big disadvantage of it is the sparseness of data. As a matter of fact, the larger N is, the more serious sparseness is. The effect is as the same as DNA pattern analysis. A structure of prefix tree can explain it well. For example, a subsequence of DNA such as “ATCATA”, if we use unigram A as a pattern, there are three times occurring in this subsequence. If stopping at bigram, the most frequent of pattern is AT. If trigram is set, the most frequent time must be only one here. It shows a monotonic decreasing of feature occurring in data set by increasing the N value. A serious sparseness cannot be accepted in text mining.

English Parser: English Parser is a parser working out the grammatical structure of sentences. In fact, it can parse relations from a sentence. For example, which groups of words will be collected together as phrase feature and also which words will be the subject or object of a certain verb. This technique has specified direction to text mining, especially for those texts which have a rigorous structure of English sentences. Fortunately, Stanford Parser [20] offers help for tanking parse, and this parser have a strong robustness and low sensitive for wrong sentence structure. But a weakness of Stanford Parser is that it cannot parse a long paragraph or a big chunk of text data, and it is easy to lead to an “out of memory” error in real-time.

We will detailedly illustrate these three techniques applied in feature extraction process later, and also provide their pros and cons by some real examples in our project.

2.2 Mainstream Algorithms for Classification

In this section, there three mainstream algorithms for addressing classification problem will be illustrated briefly, that are Naive Bayes, Support Vector Machine and Decision Tree. They are very classical and still popular all over the research field.

2.2.1 Naive Bayes Algorithm

In machine learning, generally we are interesting in that how to determine a best hypothesis in

hypothesis space H when given a training data D. The simplest solution is to define the so-called "best" hypothesis as a most probable hypothesis under the different prior probabilities when the data D and H are given. Bayes Theorem offers a method to compute this probability directly. The following is the famous Bayes Rule:

$$P(h \mid D) = \frac{P(h)P(D \mid h)}{P(D)} \tag{2.1}$$

where $P(h)$ is usually called prior probability of hypotheses, $P(D)$ represents the prior probability of being observed training set, $P(D|h)$ denotes the probability of being observed training data D when h is true. We are interesting in $p(h|D)$, namely, the probability of h is true when training data D is given. $P(h|D)$ is also called the posterior probability of h.

Naive Bayes algorithm is able to resolve the classification problem even to multi-class. An objective function $f(x)$ is found from a finite set V. The agent is provided a series of training set and new instances $<a_1, a_2 \ldots a_n>$ (attribution tuple), and then being asked for a prediction on objective function (classification). The new instances will be assigned to a maximum a posteriori value v_{map}.

$$v_{map} = \arg\max_{v_j \in v} P(v_j \mid a_1, a_2 \ldots a_n) \tag{2.2}$$

rewrite the *formula (2.2)* by *(2.1)*:

$$\begin{aligned} v_{map} &= \arg\max_{v_j \in v} \frac{P(v_j)P(a_1, a_2 \ldots a_n \mid v_j)}{P(a_1, a_2 \ldots a_n)} \\ &= \arg\max_{v_j \in v} P(v_j)P(a_1, a_2 \ldots a_n \mid v_j) \end{aligned} \tag{2.3}$$

also the probability of union $<a_1, a_2 \ldots a_n>$ is equal to the probability product for each attribute:

$$P(a_1, a_2 \ldots a_n \mid v_j) = \prod_i P(a_i \mid v_i) \tag{2.4}$$

substitute the $P(a_1, a_2 \ldots a_n|v_j)$ in *formula (2.3)* by *(2.4)*, and then we have Naive Bayes classifier:

$$v_{NB} = \arg\max_{v_j \in v} P(v_j) \prod_i P(a_i \mid v_j) \tag{2.5}$$

2.2.2 Support Vector Machine

SVM is a linear classification promotion. The different with ordinary linear classification showed in *Figure 2.1* is that SVM deals with a data point as a p-dimension vector, while an ordinary linear function only acts on 2-dimension. When those data points project to a higher dimension, we need an advanced "linear function" to separate those data points, which is so-called "hyperplane". As a supervised learning algorithm based on Structured Risk Minimisation principle from computational learning theory [31], SVM tries to find out a hyperplane with the maximum separation margin between two classes, and those the closest data points along the boundaries are so-called Supper Vectors (*Figure 2.2*).

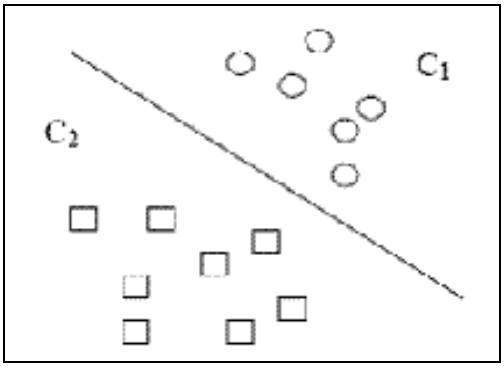


Figure 2.1: ordinary linear classification

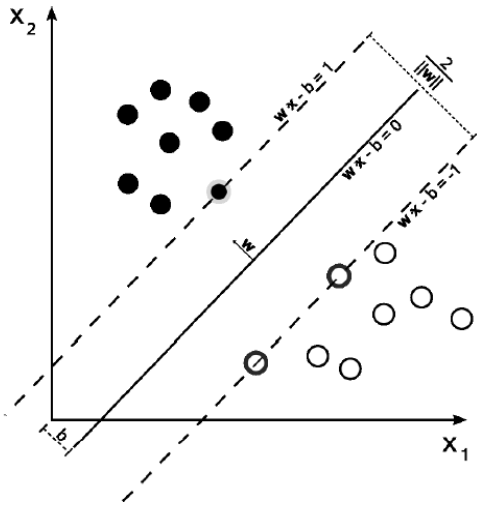


Figure 2.2 (from Wikipedia): the cycles on the dashed line are Support Vector

As the *Figure 2.1* shown, a linear function in 2-dimension can be presented as:

$$f(x) = wx + b \tag{2.6}$$

and when the data points in a high dimension, the *formula (2.6)* required to be changed a little:

$$f(x) = < w, x > + b \tag{2.7}$$

where $<w, x>$ denotes a dot product of vector w and vector x, and $w=\alpha_1 x_1+\alpha_2 x_2+\ldots+\alpha_n x_n$. Indeed, the w is not only affected by data vector, but also has a relationship with their labels. For example, there is a data vector in *Figure 2.2* keeps its position but changed its label, from a black cycle to a white cycle. This action leads to the whole hyperplane changed. Thus, the vector w can be written as $w=\alpha_1 y_1 x_1+\alpha_2 y_2 x_2+\ldots+\alpha_n y_n x_n$, also can be rewritten as:

$$w = \sum_{i=1}^{n} (\alpha_i y_i x_i) \tag{2.8}$$

Substitute *w* in *formula (2.7)* by *(2.8)*, we have:

$$\begin{aligned} f(x) &= <\sum_{i=1}^{n}(\alpha_i y_i x_i), x> + b \\ &= \sum_{i=1}^{n}\alpha_i y_i <x_i, x> + b \end{aligned} \tag{2.9}$$

Thus, the optimum problem can be written as:

$$\begin{aligned} &min \quad \frac{1}{2}\|w\|^2 \\ &s.t. \quad y_i[(wx_i)+b]-1 \geq 0 \qquad (i = 1,2,...,n) \end{aligned} \tag{2.10}$$

In fact, data vectors in high dimension are normally non-linear separable due to there are existing noisy data in training set, which seriously affect the classifier. So we allow our hyperplane to give up precise classification in order to achieve a lager margin. Thus, it changed from hard-margin $f(x) \geq 1$ to soft-margin $f(x) \geq 1-\zeta$ by adding a slack variable ζ where $\zeta \geq 0$.

2.2.3 Decision Tree

Decision Tree has received a great deal of attention over the past half century. It is intuitive and simply to be produced from data. It is a kind of Rule Learning, and sometimes has good performance towards a certain model. Decision Tree learning can be represented one or more if-then rule to increase the readability. It has been wisely extended to other field such as medical diagnosis, etc.

Decision Tree classify testing instances by sorts the instance from root node to a leaf node. A leaf node is the label of the new instance. Each node in the tree assigns a test for an attribute of an instance, and those descendant nodes of a certain node are corresponding to a possible value of this attribute. When given a new testing instance, the way of classification is starting from the root node meanwhile testing the specified attribute of this node, and then moving to a next node corresponding to the attribute value of this instance, and finally, repeat this process in a new subtree meanwhile treat the new node as a new root node.

A common method for decision tree generation is Inductive Dichotomizer 3 (ID3) [26]. The ID3 algorithm searches in a greedy fashion, for those attributes which create the maximum amount of information for the label of instances decision in training set. Due to using the greedy search, the algorithm never backtracks to considering the past choice.

With respect to how to select a good attribute as a node, ID3 measures the attribute value by a statistical attribution – Information Gain. The Information Gain (*formula 2.11*) is used to measure the ability of distinguishing a training instance for a given attribute.

$$Gain(S, A) \equiv Entropy(s) - \sum_{v \in Values(A)} \frac{|S_v|}{S} Entropy(S_v) \quad (2.11)$$

where *Values(A)* is a set of all possible value of an attribute A, S_v is a subset of value v corresponding to the attribute A in S, and *Entropy(S)* which is able to describe the purity of any sample set is defined as:

$$Entropy(s) \equiv \sum_{i=1}^{c} - p_i \log_2 p_i \quad (2.12)$$

where p_i is a proportion of class i in S.

A common problem in Decision Tree algorithm is that a hypothesis overfits the training set. This problem may be caused by random error or noisy data. In order to avoid overfitting problem, there several solutions can be applied:

- Stop increasing the tree at an early time, before the ID3 algorithm perfectly classified the training instances.
- Post-prune, namely prune the overfitting tree after creating the whole tree.

2.3 ROC Analysis

Receiver Operating Characteristic derives from statistic decision theory, and was introduced in signal detection theory to state how well a receiver is able to determine a signal from noise. Recently, this method was employed in machine learning and now is widely used as a tool for visualising and analysing many parts of machine learning algorithms. In fact, it is a graphical plot using for the optimise evaluation. Also, it is intuitive to represent a trade-off between true positive rate (TPR) and false positive rate (FPR).

A binary class ROC analysis curve is in a 2-dimension, where the TPR is as verticals (Y axis) and FPR is as horizontals (X axis). Both rates are estimated as below:

$$TPR = \frac{number\ of\ true\ positive}{number\ of\ positive}$$

$$FPR = \frac{number\ of\ false\ positive}{number\ of\ negative}$$

Also it can be shown in a 2x2 *Confusion Matrix* (*Table 2.1*):

	Positive' (Predict)	Negative' (Predict)
Positive (Actual)	True Positive	False Negative
Negative (Actual)	False Positive	True Negative

Table 2.1: a 2x2 Confusion Matrix

Thus, the TPR and FPR introduced above can be detailed described as:

$$TPR = \frac{true\ \ positive}{true\ \ positive + false\ \ negative}$$

$$FPR = \frac{false\ \ positive}{false\ \ positive + true\ \ negative}$$

Each prediction outcome or one instance of a confusion matrix represents a single point in the ROC curve, and all point can be connected from the upper right corner (coordinate: 1, 1) to lower left corner (coordinate: 0, 0). The best prediction point would be plotted in the upper left corner (coordinate: 0, 1), representing a perfect result, which means no false negatives and also 100% true positive rate. A random guess would be represented as a point along a diagonal line from the upper right corner to the left lower corner. This guess is as the same as a decision by flipping coins. Thus, the space which is above the diagonal line implicates good classification results, while the space that is below the diagonal line represent poor classification results. The *Figure 2.3(left)* shows some characteristics of a simple ROC space we described above.

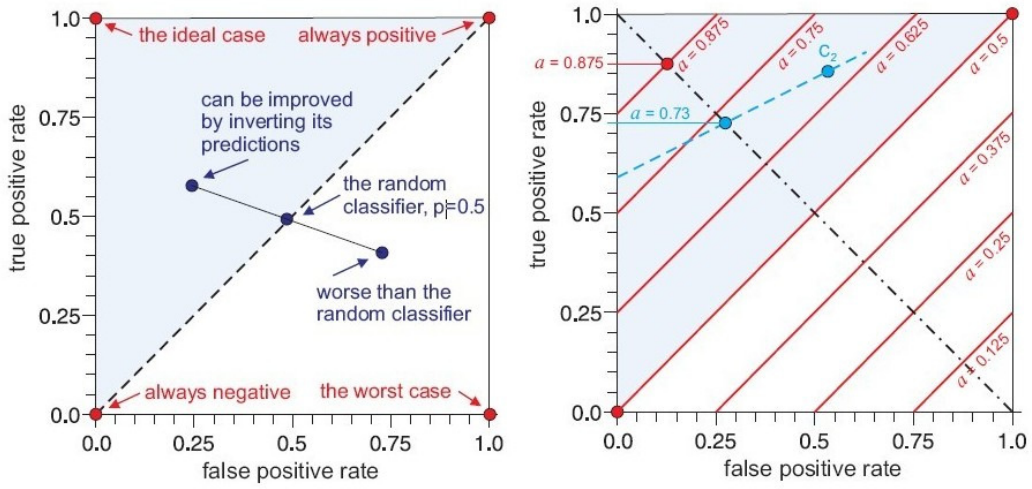
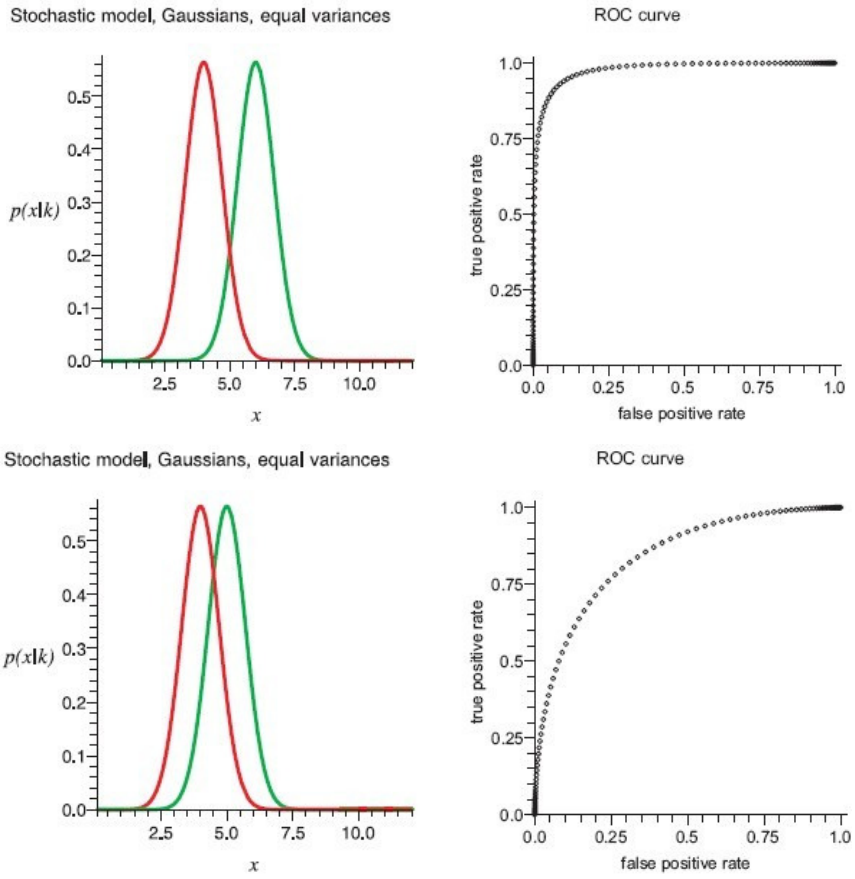


Figure 2.3: an intuitive description of ROC Space

The red lines corresponding to an accuracy value in *Figure 2.3* (right) is so-called iso-accuracy line. This line connecting all points in ROC space has the same accuracy. For instance, all points along the red line near coordinate (0, 1) have the same accuracy *a=0.875*, and the blue dashed line is also an iso-accuracy line illustrating a general classifier c2 which has accuracy *a=0.73*. An iso-accuracy line is a straight line with the slope *negative/positive*. Thus, there is a little trick to estimate accuracy. We can find the point of intersection of the iso-accuracy line and the black dashed line showing in *Figure 2.3* (right). A line parallel to X axis through the point of intersection will intersect Y axis, and the value in Y axis is the accuracy of a classifier. For example, the red line near coordinate (0, 1) have accuracy *a=0.875*, and the point of intersection in Y axis is *0.875* as well.

ROC space is able to show how different the data distribution of binary class. Here, we use a series of paired graphical plots to show the effect of data distribution projecting to a ROC space. Assuming there are two different classes based on the Gaussian distribution. The red curve and the green one represent two data classes respectively (*Figure 2.4*). We set the mean of red one as the same value each time and only change the mean of green one. The corresponding ROC curve will be visualised to describe the data distribution effects.



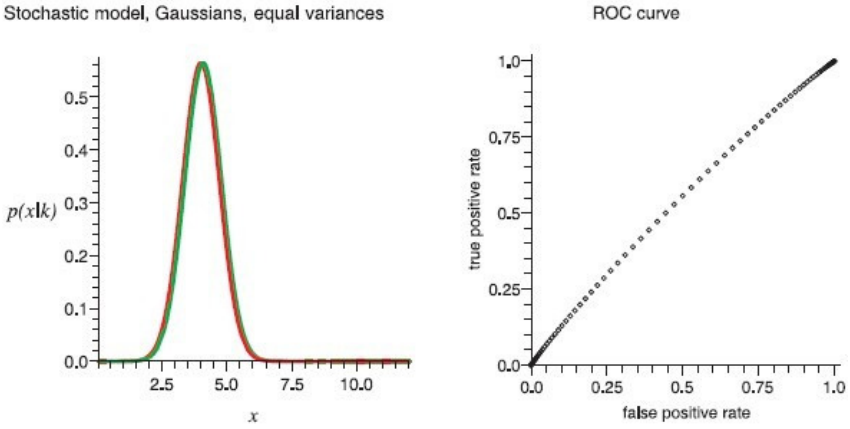


Figure 2.4: the left graphics are the distributions of two different classes. The right graphics are the corresponding ROC curves.

The parameters in each paired graphics are:

First: μ_{red}=4.0, μ_{green}=6.0, σ_{red}=σ_{green}=1.0
Second: μ_{red}=4.0, μ_{green}=5.0, σ_{red}=σ_{green}=1.0
Third: μ_{red}=4.0, μ_{green}=4.1, σ_{red}=σ_{green}=1.0

We assign the same mean value for red class and both standard deviation are the same. From the information given in *Figure 2.4*, we can conclude that if two classes are more overlapping, it will have worse disciminability, and also the performance in ROC curve will be approximate to a straight line. The purpose of ROC analysis is to discover a threshold within the intersection part of two distributions in order to achieve a higher true positive rate and a lower false negative rate. In ROC space, the point closest to the coordinate (0, 1) is an optimal choice.

2.4 Opinion Extraction & Feature Selection Applied in the Project

Following the information have been briefly mentioned in the *Section 2.1*, there are four solutions of NLP to implement opinion extraction and feature selection in this project. They are: Information Gain, N-gram, POS and English Sentence Parsing (or relations parsing).

The goal of opinion extraction process is to collect that the most useful information from a mess data set. On one hand, those extracted information are able to represent the reviewers' main idea, which is the main purpose of this project. On the other hand, it is also able to reduce noisy data from the data set. Feature selection process is also known as feature reduction in data mining. It would be helpful to choose those data which have strong identification ability for classification.

2.4.1 Information Gain

Information Gain is used to measure the identification ability of a feature. This function (*formula 2.11*) produces a value that a high value means a good feature. It is a basic solution for feature

selection in machine learning. Normally, an initial data set may contain a various forms for an English word. For instance, a base or root word, “train”, have some derived forms, such as “trained”, “training” or “trains”. If we deal with each form as a unique feature, generally these three features have a high information gain, but we do not want to treat them as three different features due to they have the same meaning in many situations. Deciding the class for a testing document based on a word form does not make any sense. If we do so, it is easy to result in overfitting. Thus, before using information gain to select features, we carry out data preprocessing. Here porter stemming and stop words list can be safely used, because when we only use information gain to collect useful features, there is no direct relationship with sentence structure and word property.

First of all, we import the Lucene Library and use Porter stemming function to map those words which have the same root or base form into a common term. After normalising process, we are able to reduce a number of attributes to avoid overfitting. For instance, after stemming words, those different forms of a word such as “trains”, “training”, “trained” and even “train”, will be forced to be changed as a root or base term – “train”. Second of all, we perform stop words removing by using the internal stop words list in Lucene. Finally, we keep 1000 top words of high information gain as the model. Each review includes a bag of features representing by single string vectors.

2.4.2 N-gram

Since the features in information gain model are single word, we will not use unigram as a model here as the same as information gain. Similarly, trigram and n-gram are abandoned due to the reason of data sparseness. A seriously sparse data model is not helpful for classification. For example, a model having serious data sparseness always depending on its prior probability in NB, which means the likelihood part is always zero if we use Laplace correction and calculate in log form. Thus, we decide to adopt bigram form.

Bigram is a combination of two isolated features which are known as phrase from. This method is good at capturing context. In more detail, comparing with the information gain model, Bigram is able to extract a negation phrase, “not-good”, while the information gain or unigram is only able to choose a single term “good” as a feature. It is common that the meaning of “not good” is opposite to “good”. The phrase “not-good” is the real opinion of reviewers. In single word model, the negation word “not” is usually filtered out by stop words list.

2.4.3 Part-of-Speech Tagger

POS approach is very popular in text mining. This method has better pertinency and maneuverability. Previous work has shown that adjectives are good indicators of subjective [12, 35, 36] since they can represent human’s sentiment. Based on this idea, we not only build up an isolated adjective, noun or verb model, but also make difference combinations of these properties to be a phrase form.

Here we show a list containing the main tagger we prefer using in this project to state the different taggers by using Stanford POS Tagger [16, 17].

JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
NN	Noun, singular or mass
NNP	Proper Noun, singular
NNPS	Proper Noun, plural
NNS	Noun, plural
RB	Adverb Most words that end in -ly as well as degree words like quite, too and very
RBR	Adverb, comparative Adverbs with the comparative ending -er, with a strictly comparative meaning.
RBS	Adverb, superlative
VB	Verb, base form subsumes imperatives, infinitives and subjunctives
VBD	Verb, past tense includes the conditional form of the verb to be
VBG	Verb, gerund or persent participle
VBN	Verb, past participle

Table 2.2: a list of taggers

The motivation of extracting those specified taggers is that nouns, adjectives, adverbs and verbs are the basic elements of a sentence. Each of them is able to expression emotion more or less. In fact, a combination has a strong ability or an accurate direction for opinion extraction. For instance, *JJ+NN*, *RB+VB*, *NN+VB*, *RB+JJ*, etc. A refinement operation from isolated features to phrase features approximates human's real emotion more than simplex words.

It is important to be noticed that stemming words now is forbidden to be used. There are couple of reasons for this warning. Firstly, the algorithm for tagging in POS tagger is based on the structure of sentence. Although it has a strong robustness for data, if we stem words first, the accuracy of correctly tagging action would be decreased. For example, considering the following sentence:

"This is a tagged tag"

after stemming process, it is changed to be:

"This be a tag tag"

Then we compare the tagging results before stemming and after stemming process:
Before: *This/*DT *is/*VBZ *a/*DT ***tagged/*JJ *tag/*NN
After: *This/DT be/*VBZ *a/*DT ***tag/*NN *tag/*NN

Thus, if we extract a phrase with the pattern *JJ+NN*, the information after stemming process would be lost. Similarly, if we only want to extract isolated adjective, the feature "tagged" is missing again.

2.4.4 Sentences Parser

This approach is to use a parser to parse a structure of a sentence based on dependency parse tree. Dependency parse tree provides an institutive and useful structure for the sentences by annotating edges with dependency types, such as subject, modifier, etc. Sanford Parser provides a function to extract the grammatical relations from free text sentences. We can choose a pattern to extract features from sentences by Sanford Parser. For instance, there is a typical sentence in our data set:

"The main problem with this paper is that relevant work is missing."

In fact, this sentence is also a full review comment of a particular instance in the data set. Obviously, it is a negative instance, but we are interested in how to know the class of this instance. Comparing with POS, we can easily know the advantages of the parser.

The most representative features that can show a negative sentiment in this sentence are "main problem" and "is missing". By using POS, it is easy to find the features "main problem" and "relevant work" due to the *JJ+NN* pattern, while the feature, "is missing", would be really missing. Generally, the pattern *VBZ+VVG* is not usually used to extract features. The word "missing" is actually an adjective, but the tagger deals with it as *VVG*. Also, the pattern *VBZ+JJ* is still unusual, since a single pattern JJ can illustrate the same meaning.

With respect to parser, we design three pattern forms for this sentence as "*mod*", "*ccomp*", and "*nsubj*". Depending on those three patterns, all main features which can fully represent the sentiment of the reviewer can be extracted. The typed dependencies after parsing are:

det(problem-3, The-1)
amod(problem-3, main-2)
nsubj(is-7, problem-3)
prep(problem-3, with-4)
det(paper-6, this-5)
pobj(with-4, paper-6)
complm(missing-12, that-8)
amod(work-10, relevant-9)
nsubj(missing-12, work-10)

cop(missing-12, is-11)
ccomp(is-7, missing-12)

Following the patterns we designed before, the features became:

amod(problem-3, main-2)
nsubj(is-7, problem-3)
amod(work-10, relevant-9)
nsubj(missing-12, work-10)
ccomp(is-7, missing-12)

These five features are able to illustrate a full meaning of the original sentence. Firstly, it is a "main problem" and "is" really a "problem". Then the "relevant work" is a "missing work" and it "is missing".

Dependency parse tree provides a syntax parsing process. In fact, it is stricter than POS and easier to comprehend. The example above also shows a good accuracy for modelling due to the four features seems like negative feature except "relevant work". However, the disadvantages of parser are that it requires an expensive time and memory cost, while POS is much more efficient. The time complexity of Parser is about $O(n^3)$ and usually require 1GB memory to run, often more, while POS tagger usually run at thousands of words per second and only requires 60MB-200MB memory [6].

As the same as the restriction for using porter stemming and stop words list in POS tagger, Stanford Parser does not intend to use data preprocessing as well. In our case, we did not want to apply porter stemming and stop words removing before using tagger and parser. To reduce the memory cost by parser, we impose Lingpipe library to normalise the documents. There is a function in Lingpipe can extract a full sentence from a chunk of text area. Because Stanford Parser cannot work on a chunk of text, thus normalising the reviews into the sentence level makes the working of the parser easier and more stable. Although some mistakes are still occurred during the process, we are able to escape the Java memory heap proble. The problem of sentences normalisation is that Lingpipe separates a sentence from a chunk of text by the period - "." and a capital alphabet which results in an error when there is a human's name or an ellipsis. For example, look at this sentence in our data set:

"They showed utility vs. NetKit (wvRN) on the Cora dataset but didn't take this further."

This is an original sentence from a review comment. Lingpipe deals with this whole sentence as two sentences due to the "vs. NetKit". The result after Lingpipe processing will be:

"*They showed utility vs.*"
"*NetKit (wvRN) on the Cora dataset but didn't take this further.*"

A more accurate method to extract sentences can be by handcraft. However, the size of review database is not small, so this unskilful approach has to be aborted. We allowed some mistakes existing in our new features model. In next step we will convert those features from the default binary format into a weighting format.

2.4.5 Term Frequency times Inverse Document Frequency

After extracting features, we still need to process the new data set, especially for n-gram model, POS tagger model and parser model. An algorithm we prefer to using is named Term Frequency times Inverse Document Frequency (TF-IDF). TF-IDF is a common weight technique using in text mining and information retrieval. It is a statistical method for measuring how significant a word is in the data set.

The weight is the score computed by TF-IDF algorithm for the word in the chunk. TF represents: the frequency times when a word occurs in a document:

$$TF = log(1 + f_{ij}) \tag{2.13}$$

the purpose of weighting 1 is to avoid the logarithmic function meaningless. where

$$fij = \frac{the\ times\ of\ the\ word\ occurs\ in\ a\ document}{the\ number\ of\ all\ words\ in\ this\ document} \tag{2.14}$$

means the frequency of the term *(word)* in a document.

IDF means: if a fewer documents which contain a target term, the result value of IDF is bigger, which means this term has a good separating capacity for classifying. If the number of documents which contains a term in a class is *m*, while the number of the other documents which contain the term in other classes is *k*, obviously, the totally number of documents which contain the term is *n*=*m*+*k*. We can clearly to see that when the m is bigger, the *n* is bigger, and according to the formula of IDF, the value of IDF is smaller, which results in the term has less separating capacity for classifying. The formula of IDF is:

$$IDF = fij \times log \frac{total\ number\ of\ documents}{the\ number\ of\ documents\ which\ contain\ the\ term} \tag{2.15}$$

The TF-IDF is used to indicate the correlation between a single term and particular document. It is useful to show which terms have a strong distinguishing ability.

2.5 Classifiers and SVM Ranker

In this section, we are going to indicate a general design for this project and introduce the relevant background information of algorithms we will use.

A main aim in this project is to examine whether if suffices to deal with sentiment classification as some special cases of review categorisation. We will start from binary classification to

multi-class classification (with the two labels being positive and negative sentiment, the three labels being positive, neutral and negative sentiment and finally, the six labels being score 1 to 6). We will design the experiments with three standard algorithms: NB, SVM and DT. In fact, the philosophies among these three classifiers are quit different. Additionally, we extend the Naive Bayes to a multinomial form, namely, Multinomial Naive Bayes.

With respect to classification problem, there are several choices we can treat review ratings as labels. To discover the difficulty in our data set, we prefer to test a general data distribution among different ratings. The score for each review states different sentiment of an independent reviewer. The sentiment distance between score 6 and score 1 should be further than the distance between score 3 and 4. First of all, we want to use those reviews which are belonging to score 1, 3, 4 and 6 to examine the difficulty in our project. Second of all, three standard algorithms NB, SVM and DT will be performed on a normal binary classes (score 1 to 3 are negative, score 4 to 6 are positive), a three classes (score 1 to 2 are negative, score 3 to 4 are neutral, and score 5 to 6 are positive), a multi-class (each score represents an independent class).

As a matter of fact, the experiments show Multinomial NB classifier outperforms to the others, and we do not stop at the current accuracy. The next step we want to optimise the accuracy by adding a weight value into the class score computed by Multinomial NB. To find an optimal weight [18] for each class, ROC analysis is our preferring choice. Due to finding a global optimum for multi-class problem in weight space is in NP-complete [8], we present the bottom up algorithm [18] for producing an optimal solution in three classes (negative, neutral and positive) classification. This is also able to extend to the classification of six classes, but since the time limitation, we stop at three classes, which is enough to show its performance.

With respect to ranking problem, we pay more attention to the ratings of review data set. Indeed, the scores can be ranked from high value to low. Our idea is to train a ranker to sort those documents correctly, which means we now consider a function that mapping from objects to ranks. The top part of model is placed by those documents belonging to score 6, and the rest parts are the documents with the decreasing score value. We impose SVMlight [32] application to train a ranker with pairwise learning algorithm. Because here we deal with our data set to be a ranking problem, initially we transform ranking to pairwise classification problem.

Firstly, we illustrate a definition about **Preference** [5]:
A preference is a relative statement about how two documents related. An ordering $\succ$ indicates a preference of two elements of the input space X

$$\{ x_i \succ x_j \} \ where \ x_i, x_j \in X \tag{2.16}$$

We treat a high score document as preference, and then these preferences can be used to train a ranker by pairwise algorithms [33]. The following (*Figure 2.5*) is a ranking SVM to show how it works:

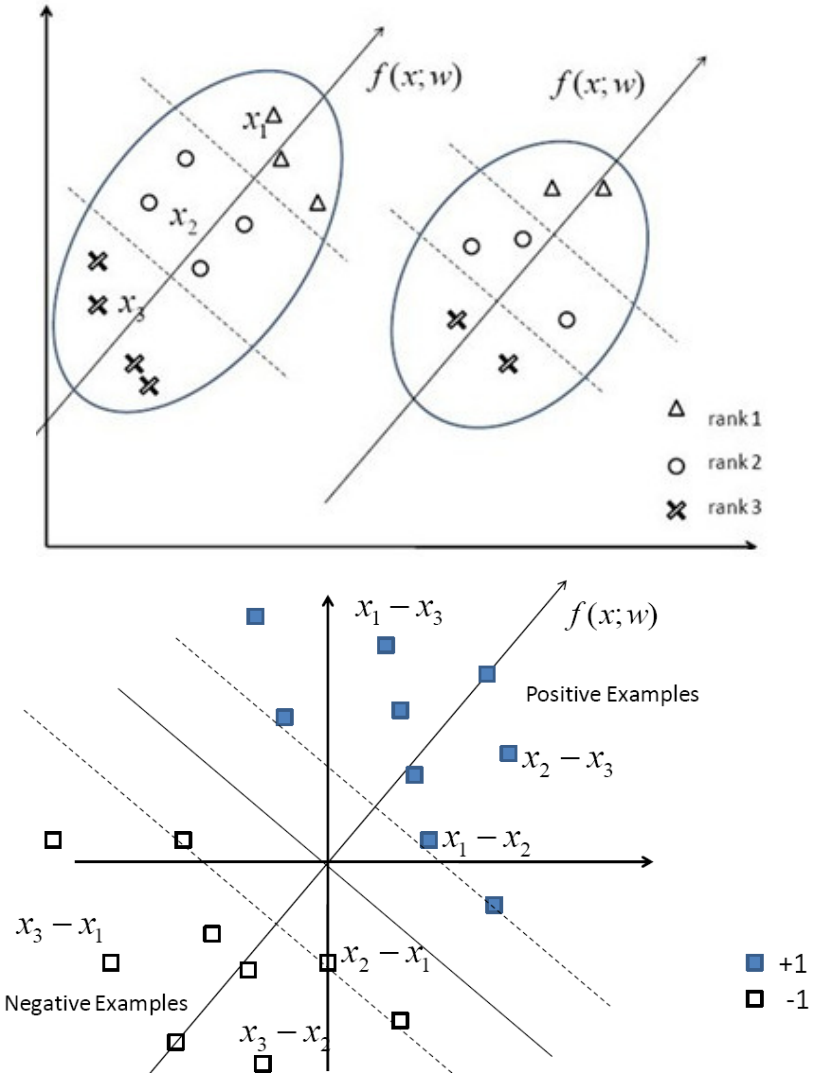


Figure 2.5 (from Hang Li, 2009): pairwise learning by SVM

As it shown in *Figure 2.5*, the top part is the feature vectors presented in a coordinate space. Different rank numbers are the different score intervals in our project. The bottom part in *Figure 2.5* shows the pairwise training feature vectors in ranking SVM. The pairwises are the difference between two single feature vectors in training set or input space *X*. Now, we need a ranking function *f: X→R* to address ranking problem. Thus, we have:

$$x_i \succ x_j \iff f(x_i; w) > f(x_j; w) \qquad (2.17)$$

Also we define an indicative function, *formula (2.18)*, to illustrate the label for the pairwise algorithm:

$$(x_i - x_j, y),\ y = \begin{cases} +1 & x_i \succ x_j \\ -1 & x_j \succ x_i \end{cases} \tag{2.18}$$

In fact, a ranking SVM is equal to a pairwise classification SVM. In ranking SVM, the pairwises are the training instances instead of a single feature vector in ordinary SVM. By using this algorithm, we can easily know which feature vector is preference to the others. A preferring feature vector implicates a high review score in our project.

Following the information given above, we can conclude the main difference between classification problem and ranking problem.

Classification: To a binary case, the classification is defined by a distribution P over a bag of features $X \times \{0, 1\}$, where $\{0, 1\}$ is the binary label space and the X is the observable feature space. Therefore, our aim is to train a classifier C: $X \rightarrow \{0, 1\}$ minimising the expected cost or maximising the expected accuracy on P given by [19]

$$e(c, P) = Pr_{(x,y)} \sim p[c(x) \neq y] \tag{2.19}$$

The classification regret of c on P is defined as

$$r(c, P) = e(c, P) - \min_{c^*} cost(c^*, P) \tag{2.20}$$

Or

$$r(c, P) = e(c, P) + \max_{c^*} acc(c^*, P) \tag{2.21}$$

Ranking: Let X denote the set of ordered pairs of distinct elements of X. Let π: $X \rightarrow \{0, 1\}$ be a ranking rule [30] or a preference function. Thus, we can say $\pi(x, x')=1$ if π prefers x to x', and 0 otherwise, denoting Θ as rank difference: $r_i \Theta r_j := i - j$. The loss function in ranking problem extended from *formula (2.17)* and *(2.18)* can be formalised [28]:

$$c_{pref}(x_1, x_2, y_1, y_2, \pi(x_1), \pi(x_2)) = \begin{cases} 1 & y_1 \Theta y_2 > 0 \wedge \pi(x_1) \Theta \pi(x_2) \leq 0 \\ 1 & y_2 \Theta y_1 > 0 \wedge \pi(x_2) \Theta \pi(x_1) \leq 0 \\ 0 & else \end{cases} \tag{2.22}$$

It is noticed that we can obtain m^2 samples drawn according to $p(x_1, x_2, y_1, y_2)$. It is essential that those samples do not offer m^2 iid samples of the function $c_{pref}(x_1, x_2, y_1, y_2, \pi(x_1), \pi(x_2))$ for any π.

2.6 A Closer Look at the Data Set

After introducing the background of relevant knowledge and techniques, we take a deeper observation towards our data set in order to capture more background information of the data set and make a future plan.

Our data set or say training set is about reviews of conference paper. There are several parts in this training set: “A Brief Summary of the Main Contributions of the Paper”, “Three Strong Points of the Paper”, “Three Weak Points of the Paper”, “Detailed Comments to the Authors”, “Overall Recommendation (Numeric)”, and also including other irrelevant information such as “Organization” or “E-mail”, etc.

The main goal of this project is to use the “Overall Recommendation (Numeric)” that is a numeric score from 1 to 6, as target and make use of the textual information to train a variety of classifiers to predict these targets, so the preferring algorithms are based on supervised machine learning framework due to that information data are labelled.

Since the targets are from score 1 to 6, where a higher score means a better paper, it is able to deal with it as two different solutions. The first solution is to treat it as a pure classification problem, a binary classification or multi-class classification. For the binary classification problem, scores from 1 to 3 are negative class and from 4 to 6 are positive class. While for the multi-class classification problem, similarly, the scores from 1 to 2 are negative class, 3 to 4 are neutral class and 5 to 6 can be positive class. A more complicated scenario would use these six scores as six different classes. The second solution is that it is possible to transfer it to a ranking problem due to the rankings of scores. A ranker can be trained to sort the instances and a threshold/thresholds can be set to determine the boundary of each class. Here we listed the documents distribution in each score:

Score	#Documents
1	91
2	423
3	530
4	355
5	171
6	42
Total	1612

Table 2.3: the numbers of documents in different scores

From the *Table 2.3*, it is clear to show the documents approximate to Gaussian distribution. The general documents occupy more of the position than those special ones which are really good or weak. If we separate the whole data set as two parts, positive (4-6) and negative (1-3), the training set is an unbalance set. Because this topic of project is about reviews of international conference paper and the reviewers are the scientists in research field, the contents of review are very different from others such as movie review, product review, etc. An important differentiator between them is that the reviews in our data set are much more rigorous than others. For example, those words such as “awesome”, “rubbish”, etc., which are able to express strong emotion are never appeared in our data set. This reason increases the challenge of this project.

2.7 Workbench

This project was implemented by using Java programming language on Eclipse version 3.2. Also, we used Matlab 7 for visualising the ROC curve. Additionally, there were several external libraries and toolkits we also used. We listed the name and using purpose here:

Stanford POS Tagger: for tagging the word with its characteristic or property.
Stanford Parser: for parsing the sentences, and yields the grammatical structure.
Lingpipe: for extracting every sentence in a paragraph.
Lucene: for data preprocessing.
LibSVM: for training a SVM classifier.
SVMLight: for training a SVM ranker.
Weka: for experiments based on the models.

2.8 Previous Work

The success of machine learning techniques applied in classification problem gives hope that it may be possible to adapt similar ideas to text mining or review mining in our case. In recent years, review mining is a popular topic in computer science and there are lots of interesting techniques for mining opinion or personal emotion.

2.8.1 Semantic Orientation Approach

Firstly, we introduce a very interesting idea different with using machine learning techniques for addressing the reviews classification problem, Semantic Orientation [11] (SO). SO is a terminology related with linguistics, logic, psychology, cognitive science and so on. Its research object is meaning of natural language which could be different linguistic unit like words, sentences, and so on. The purpose of semantics study is to mine the regularity or hidden interpretation of representation. Semantic Orientation is also about the evaluative character of a word.

There is a famous paper which is about semantic orientation applied in opinion extraction process written by Turney [25]. In this paper, the motivation of author was that use an algorithm to classify different domain reviews as *recommend* or *not recommend* automatically. Meanwhile, the author pointed out the method using by Hatzivassiloglou and McKeown [11] was good for isolated adjectives, but showed weakness for phases containing adjectives and adverbs. So alternatively, a new algorithm was: firstly, use a POS tagger to identify phrases which contain adjectives or adverbs. Then, estimate the semantic orientation of these phrases which are identified. Finally, average semantic orientation of these phrases extracted from the reviews to determine the class of a review.

The main idea is that embed a Pointwise Mutual Information and Information Retrieval (PMI-IR) algorithm for the second step. It measures the similarity of pairs of words or phrases. PMI between two words is defined as follows:

$$PMI(word_1, word_2) = \log_2\left[\frac{p(word_1 \,\&\, word_2)}{p(word_1)p(word_2)}\right] \quad (2.23)$$

where $p(word_1\&word_2)$ is the probability that both $word_1$ and $word_2$ are occurred in the same time. If the words are independent, the probability that they co-occur is given by the product $p(word_1)p(word_2)$. Thus, the ratio of $p(word_1\&word_2)$ and $p(word_1)p(word_2)$ is a measure of the degree of statistical dependence between the two words. Generally, one word is as an anchor word and the other is as the measured word. When we observe the other, the log function is the amount of information about the presence of one of the words.

The formula of semantic orientation is:

$$SO(phrase) = PMI(phrase, \text{“positive”}) - PMI(phrase, \text{“negative”}) \quad (2.24)$$

from the formula shown above, obviously, semantic orientation is positive if phrase is close to “positive” and negative if phrase is close to only “negative”.

Turney chose AltaVista Advanced Search Engine (*http://www.altavista.com*) as the experiment environment, since it had a large capacity of web pages, 350 million, and all the web pages were in English and there was a NEAR operator in AltaVista. NEAR operator limited the documents search, which contained the length of words within ten words of one another, in either order. To compare with his previous work [34], NEAR operator showed better performance than AND operator, because NEAR operator was better for measuring the strength of semantic association between words. If co-occur was an expression of NEAR, the estimate of semantic orientation can be derived from *formula (2.23)* and *(2.24)* shown above with minor algebraic manipulation:

$$SO(phrase) = \log_2\left[\frac{hits(phrase\ NEAR\ "positive")\,hits("negative")}{hits(phrase\ NEAR\ "negative")\,hits("positive")}\right] \quad (2.25)$$

where *hits* in *formula (2.25)* is the number of hits returned given the query. For instance, firstly, we input “awesome NEAR positive” into the query through AltaVista, and then it will show 868,000 results. Secondly, we input “awesome NEAR negative”, and then it will show 357,000 results. Finally, we input “positive” and “negative” as anchor words into the query respectively, the results are 1,330,000,000 and 755,000,000 respectively. Then substitute the number results to *formula (2.25)*, and we can know the synthesized vector of the word “awesome” which is positive or negative by calculating the function.

This method performed good results on automobiles reviews, with 84% accuracy, while with respect to movie reviews, there was only 66%. The reason of the low accuracy for movie reviews was that sometimes a positive phrase maybe “near” one star norm, while a negative phrase could be a high score review. For example, the word “unpredictable” always implies a bad review for other domains, especially for science aspect. However, this is an absolutely good vocabulary for a good movie review. Oppositely, a positive word such as “predictable” implies a bad meaning for a movie review. To sum up, semantic orientation shows good performance in some review

domains, but once involving sentiment of phrase, its accuracy is dropped down. The potential problem is that Turney only set two anchor words “excellent” and “poor” as anchor words to support the sentiment fuzzy architecture, which is considered roughly judgement words property.

Hu and Liu’s paper [22] states a process of products reviews mining by using SO as well. The performance results in their experiments show a good result for binary classification problem, which the average accuracy is up to 84%. The main difference between their method and Turney’s is that Hu and Liu’s model classifier was at the sentence level rather than documents level.

Semantic Orientation method is to compute a cosine angle of two words vector, target word and anchor word, in order to discover the sentiment of the target word whether it is “near” positive or negative. In other words, this approach offers a possible way to calculate the similarity of testing instances and target class.

2.8.2 Machine Learning Classifiers

Pang, Lee and Vaithyanathan’s [2] work on classification of reviews is perhaps the closest to ours. They employed machine learning technologies (Naive Bayes, Maximum Entropy and Support Vector Machine) to deal with the same issue. It considered the problem of reviews classification not by topic but by overall sentiment. The main difference between Turney and this experiment was that Turney implemented a specific pseudo-supervised learning technique (Beineke, Hastie and Vaithyanathan [23]) based on the mutual information between phrases, while they used completely prior-knowledge-free supervised machine learning method. The results of the three algorithms experimented by Pang, Lee and Vaithyanathan outperformed the human-generated baselines which was the accuracy 69%. Comparing among the three algorithms, SVM tended to perform the best, but the differences among them were not very large. Finally, they point out that in some special occasion, a human would easily distinguish the true sentiment of the review sometimes, while classifiers would be more difficult, due to some words illustrative of the opposite sentiment to that of the entire review.

There are two main issues when we applied machine learning algorithms; they are overfitting and data sparseness. However, there are also several solutions existing in order to avoid or reduce these problems. As described above, we can apply post-prune techniques or stop increasing the tree at an early time for decision three algorithms. With respect to the other two classifiers, we also use some preprocessing techniques to reduce overfitting problem such as port stemming and stop words removing when we select features. Now, we illustrated a detailed method to deal with data sparseness.

In [15, 27], when we use Machine Learning approaches to train a various classifiers, a method named data smoothing is helpful to avoid the probability of most sentences are equal to zero. There are two purposes of doing data smoothing, for example, when we consider the n-gram

algorithm: one is to make the sum of all n-gram probability equal to 1, and the other is to make all the n-gram probability not equal to 0. It also can be extend to other algorithms. At present, there are several data smoothing algorithms for data processing:

(1) Add-one smoothing.
(2) Add-delta smoothing.
(3) Witten-Bell smoothing.
(4) Good-Turing smoothing.
(5) Church-Gale smoothing.
(6) Jelinek-Mercer smoothing.
(7) Katz smoothing.

These data smoothing techniques are similar due to the same purposes. It employs a weight to both numerator and denominator in order to avoid zero happened. In [15], the best result is obtained by using Add-one smoothing (Laplace) showing in *Table 2.4*. It ensures any n-gram occurs at least once, which means those n-grams that not appeared in corpus are added 1. Thus, the probability of these n-grams is not zero. Therefore,

$$p_{add1}(w_1 w_2 \ldots w_n) = \frac{c(w_1 w_2 \ldots w_n) + 1}{N + V} \tag{2.16}$$

where N is the number of all n-gram (token), and V is the number of all possible different n-gram (type).

Witten-Bell smoothing is used in [15, 27] as well. The approach takes *1/(N+M)*, where N is the number of tokens, and M is the number of unique words. This method assigns that as the probability of a word which does not occur in the corpus, reassigning the remaining probabilities proportionally. The results in [15] showed the performance by using Naive Bayes with some data smoothing techniques:

Method	Test 1	Test 2
Unigram baseline	84.9%	82.2%
Naive Bayes	77.0%	80.1%
NB w/ Laplace	**87.0%**	80.1%
NB w/ Witten-Bell	83.1%	80.3%
NB w/ Good-Turing	76.8%	80.1%
NB w/ Bigrams+Lap	86.9%	81.9%

Table 2.4: Data smoothing results

From the experiment showing in *Table 2.4*, the results indicated the NB with Laplace correction show a better performance. Indeed, data smoothing has been wisely used in machine learning field. In this project, we will borrow this idea to address the corresponding problems in Naive Bayes family.

2.9 Summary

So far we have introduced background information about some NLP techniques for opinion extraction; three machine learning classifiers; ROC analysis for optimization problem; a closer observation for our data set; and previous work by other researchers. We also described a project workbench and NLP techniques for opinion extraction and feature selection we are going to adopt in our project, and meanwhile illustrated advantages and disadvantages of those techniques.

We will start this project from the next chapter. Through analysed their performance and kernel algorithms by some examples, we can construct different models and test them on different machine learning classifiers. Furthermore, to represent these features in a better way, we convert the default binary data format into a weighting data format by TF-IDF. In order to future understand the performances of different classifiers for the review data set, and the role of optimal operation, we are not only going to perform traditional machine learning algorithms including a multinomial naive bayes algorithm, but also let it be transferred into a ranking problem. Meanwhile, we want to employ ROC analysis to improve the accuracy. Finally, we will make use of Friedman test with the corresponding post-hoc Nemenyi tests for comparison of more classifiers over multiple data sets and utilize T-test to conclude the optimal operation on Multinomial Naive Bayes classifier.

3. Pure Classification Techniques

In this chapter, we introduce the winner classifier named multinomial NB, and then implement three standard machine learning algorithms and the winner on our review data. The documents in our data set are based on standard bag-of-features framework. Let $\{f_1, f_2,\ldots f_n\}$ be an isolated instance of m features that can appear in a review document. Let f_i be the individual feature representing with a TF-IDF score weighting. We not only focus on binary classification problem, but also extend them to a multi-class classification.

3.1 Multinomial Naive Bayes Classifier

Multinomial Naive Bayes (MNB) is a derivation version of Naive Bayes. A main difference between multinomial NB and NB is that the former counts a word frequency in a class. Multinomial NB classifier makes two assumptions.

The first assumption: it assumes that classification is independent of the positions of the words.
The second assumption: word appearance does not depend on position. Thus, we get:

$$P(X_i = w \mid c) = P(X_j = w \mid c) \tag{3.1}$$

for any positions *i, j*, work *w* and class *c*. Therefore, it just has one multinomial feature predicting for all words. $P(X_i=w|c_j)$ can be represented the fraction of times where word *w* appears across all documents of class c_j. The detailed algorithm is provided below [9]:

TRAINMULTINOMIALNB($\mathbb{C}, \mathbb{D}$)
1 $V \leftarrow$ EXTRACTVOCABULARY($\mathbb{D}$)
2 $N \leftarrow$ COUNTDOCS($\mathbb{D}$)
3 **for each** $c \in \mathbb{C}$
4 **do** $N_c \leftarrow$ COUNTDOCSINCLASS($\mathbb{D}, c$)
5 $prior[c] \leftarrow N_c / N$
6 $text_c \leftarrow$ CONCATENATETEXTOFALLDOCSINCLASS($\mathbb{D}, c$)
7 **for each** $t \in V$
8 **do** $T_{ct} \leftarrow$ COUNTTOKENSOFTERM($text_c, t$)
9 **for each** $t \in V$
10 **do** $condprob[t][c] \leftarrow \frac{T_{ct}+1}{\sum_{t'}(T_{ct'}+1)}$
11 **return** $V, prior, condprob$

APPLYMULTINOMIALNB($\mathbb{C}, V, prior, condprob, d$)
1 $W \leftarrow$ EXTRACTTOKENSFROMDOC(V, d)
2 **for each** $c \in \mathbb{C}$
3 **do** $score[c] \leftarrow \log prior[c]$
4 **for each** $t \in W$
5 **do** $score[c] \mathrel{+}= \log condprob[t][c]$
6 **return** $\arg\max_{c \in \mathbb{C}} score[c]$

Figure 3.1 (from [9]): Naive Bayes algorithm (multinomial model): Training and testing.

According to *formula (2.5)*, we extend it as:

$$\begin{aligned} c_{map} &= \underset{c \in C}{arg\,max}\, P(c \mid d) \\ &= \underset{c \in C}{arg\,max}\, P(c) \prod_{i=1}^{n} p(t_i \mid c) \end{aligned} \tag{3.2}$$

Each conditional parameter *P(t|c)* is a weight that indicates how good a term *t* is for class *c*. It is also called likelihood. Meanwhile, the prior *p(c)* is also a weight that illustrates the relative frequency of class *c*. Therefore, more frequent classes are more likely to be the correct class than infrequent class/classes. Those interpretations are the principle concepts of NB. Then, in multinomial NB, we estimate the conditional probability *p(t|c)* as the relative frequency of term *t* in documents which is belonging to class *c*. We have:

$$P(t \mid c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}} \tag{3.3}$$

where T_{ct} is the number of appearance of term *t* in training documents form class *c*, including duplications of a term in a document. From the assumptions we given above, T_{ct} is a count of occurrences in all positions *i* in the documents in the training set. Therefore, we do not calculate different estimates for the different positions.

There is a problem in *formula (3.3)* that if there is a term does not occur in our training set due to data sparseness, the whole *argmax* function will not be useful, which means the training data will be never large enough to indicate the frequency of rare events adequately. In order to address this problem, a solution named data smoothing we mentioned in *Section 2.8.2* has been proposed. Here, we use Laplace smoothing or add one as the solution. Thus, the *formula (3.3)* can be:

$$\begin{aligned} P(t \mid c) &= \frac{T_{ct} + 1}{\sum_{t' \in V} (T_{ct'} + 1)} \\ &= \frac{T_{ct} + 1}{\sum_{t' \in V} T_{ct'} + |V|} \end{aligned} \tag{3.4}$$

where $|V|$ is the number of different terms in the vocabulary. T_{ct} is the number of evaluated word occurring in a particular class. $\Sigma T_{ct'}$ is the total number of words with duplications in this class. We update this likelihood to instead of NB's.

3.2 From Binary to Multi-class Classification

Firstly, we examine the difficulties or characteristics in scientific reviews as mentioned in *Section 2.6*. The first task is to test the distances between each class. Thus, we separate the whole data set with their review score, and pick up those reviews which are in paired scores, (score 1 and 6), (score 3 and 4). Normally, it is harder for classifiers to distinguish the reviews which are in score 3 and 4 than score 1 and 6, because the opinions may be very different in score 1 and 6, while

similar in score 3 and 4. Score 1 represents a completely negative sentiment and score 6 illustrates an absolutely positive opinion. On the other side, both score 3 and 4 shows a neutral review. Based on this idea, there two simple experiments are carried out to show the evidences.

Data Set	Performance		
	Naive Bayes	Decision Tree	SVM
Data Set 1 (Score 1 and 6)	64.66%	64.66%	70.67%
Data Set 2 (Score 3 and 4)	53.67%	54.69%	62.60%

Table 3.1: accuracy for different paired scores

The information gain method is used to choose features in the two experiments. This is very a basic processing and the accuracy is only to indicate the different distances between paired scores. Here we implement data preprocessing such as porter stemming and stop words removing. The features are representing a unigram form. At the beginning, we only focus on the basic corpus whose attribution named "Detailed Comments to the Authors". From the results showing in *Table 3.1*, we can conclude that the distinction ability between score 1 and 6, and score 3 and 4 is significant. The reason is that discriminating the two classes which have opposite opinion is easier that the two that have similar views. The accuracies of two experiments are from 10-fold cross validation and they are able to prove that our assumptions are successful. On the other hand, the highest accuracy in data set 1 is only 70.67%, which also implies that the classification problem in this project is not easy and some advanced techniques should be used. This is the main motivation that N-gram, POS, and Parser are required on the agenda.

The next step, we append the other attributions with the basic corpus –"Detailed Comments to the Authors" so that increasing the information about the reviewer's opinion. The attribution named "A Brief Summary of the Main Contributions of the Paper" is helpful to increase the subjective sentences written by reviewers, although there are some objective descriptions still existing. Indeed, the sentences in this attribution can roughly states a basic idea from reviewers about the paper. For instance, the sentence:

"This paper mines useful features from user web sessions on search engines, and proposes a novel ClickRank framework to use those context features in getting better search result ranking."

Generally speaking, this sentence is a brief description about the paper from an objective view. However, the adjective word "useful" does actually imply a positive sentiment by reviewers.

The rest attributions in our data set we prefer to append are "Three Strong Points of the Paper" and "Three Weak Points of the Paper". Both of them are completely subjective representing by reviewers. But a problem is still unavoidable, that no matter what the score the review is, both negative and positive sentiment are included in these attributions. For example, although there is a paper given a score 6, which indicates a very nice paper and the sentiment of reviewer is

absolutely positive, the content in "Three Weak Points of the Paper" is still negative. This reason can be interpreted as scientific rigours or Einstein is not always born. Moreover, since the attribution of the review content in testing set is unknown, we cannot individually deal with the two opposite attribution, for example, just keep the content in "Three Strong Points of the Paper" attribution and not use the information in "Three Weak Points of the Paper" attribution for those reviews which have score 6. Thus, this dilemma becomes a big challenge for this project.

In order to illustrate the effect on appending attributions, we use binary classification case based on bigram form building by Stanford Parser which is introduced in *Section 2.4.4* to give the evidences:

Features	Basic Attribution				Append "Brief Summary"			
	NB	MNB	SVM	DT	NB	MNB	SVM	DT
subj	65.88	66.81	66.07	66.44	67.18	**73.14**	68.11	65.76
mod	66.87	69.10	68.11	65.76	66.44	**76.86**	67.80	64.33
comp	65.26	69.11	66.13	66.50	65.32	**69.54**	66.13	65.69
subj+mod	67.06	**75.87**	67.99	64.76	66.94	73.01	67.93	62.78
subj+comp	67.00	69.91	67.99	65.38	66.75	**77.23**	68.05	64.45
mod+comp	67.43	**75.68**	68.11	65.88	65.51	69.42	67.87	63.15
subj+mod+comp	67.31	65.76	67.87	64.27	66.44	**72.95**	67.93	62.84

Table 3.2: average 10-fold cross validation accuracies in percent. Boldface: the best performance for a give setting (row). Baseline is 64.76%

After appending the content in "A Brief Summary of the Main Contributions of the Paper" attribution in our data set, the accuracy has been slightly increased. Thus, we put those relevant attributions also including "Three Strong Points of the Paper" and "Three Weak Points of the Paper" together. The future experiments show this operation can be helpful a little for the accuracy. From *Table 3.2*, an additional information we can capture is that the multinomial naive bayes outperform the other algorithms.

Based on the experimental results, we treat all relevant attributions as a whole training set. The next step is to use this training set to build up a various models by different advanced natural language processing techniques, and then make use of those multiple data sets to test our algorithms.

Firstly, we used N-gram (bigram) method to extract features. In this part, we implement a bigram without data preprocessing and one with data preprocessing in order to show whether it has effect on this kind of feature extraction method. The result shows the data preprocessing is necessary for n-gram approach (*Table 3.3*). Secondly, we employ POS technique to yield either single feature or bigram form. This process is based on sentence level so that it is able to reduce the error rate of

features. If it is based on a chunk of text rather than the sentence level, the error may be occurred when the adjective appears in front of sentence, and the noun occurs in the following different sentence. Last but not least, we use Stanford Parser to extract the grammatical relations from sentences also based on sentence level. After using these three methods to extract features, TF-IDF is employed to weight the features, which can help us to choose those features that have strong ability for classification. The detailed results are listed in the Appendix A (Table 3A, 4A, 5A), we discuss a brief performance of classifiers on multiple data sets generated by different natural language processing methods.

Naive Bayes (Multinomial Naive Bayes)

N-gram method	Without preprocessing	With Preprocessing
Binary Classes	66.99% (63.34%)	68.92% (69.91%)
Three Classes	51.67% (51.99%)	53.91% (66.56%)
Six Classes	32.75% (38.28%)	34.37% (43.18%)

Support Vector Machine

N-gram method	Without preprocessing	With Preprocessing
Binary Classes	67.68%	67.93%
Three Classes	54.40%	54.96%
Six Classes	32.82%	33.68%

Decision Tree

N-gram method	Without preprocessing	With Preprocessing
Binary Classes	61.29%	60.98%
Three Classes	49.57%	50.74%
Six Classes	26.61%	32.20%

Table 3.3: experiments on three classifiers showing the effect of data preprocess when use N-gram

The numbers in *Table 3.3* describe that preprocessing is necessary for using N-gram method. The features' form is not sensitive to our model. It means there are few effects to feature extraction when we perform preprocessing, since it does not depend on any structure of sentence or the word characteristic or property. The results are the same as we expected before and also we have given the examples to show the reasons why preprocessing cannot be used in POS Tagger or relations Parser.

Back to our classification problem, first of all, we provide a list of methods that we used to extract features and we choose those patterns which are able to represent the human's sentiment from a common sense [25, 3].

NLP methods	Data Sets
N-gram	Bigram
POS Tagger	JJ, NN, VB, RB+JJ, JJ+NN, NN+VB, RB+VB, JJ+NN+VB, NN+VB+RB, JJ+ RB+VB, JJ+NN+VB+RB
Relations Parser	subj, mod, comp, subj+mod, subj+comp, mod+comp, subj+mod+comp

Table 3.4: data sets provided by different NLP methods

Second of all, according to the methods offering above in *Table 3.4*, we perform a binary scenario that separates the classes with positive label which scores from 4 to 6 and negative label that is from 1 to 3. The negative class contains 1044 instances and positive class includes 568 documents respectively. *Table 3.5* shows a simplified table of opinion extraction techniques with the corresponding data set which have the best accuracy in its family.

	Data Set	Accuracy
N-gram	N/A	69.91%
POS Tagger	RB+VB	61.41%
Parser	subj+comp	78.91%

Table 3.5: the best accuracy in different models

From the information given in *Table 3.5*, it is easy to be noticed that Parser method outperforms the others. An important reason is that paper review is very different to other review domains such as movies or electrical products since reviewers prefer to use objectives languages to describe the contributions of a paper. The best model by using POS tagger is RB+VB other than JJ, which shows scientists prefer objectivity. The fraction of adjectives is very low and *Table 3.6* shows it for each score. The results seem to be scientists like using more adjectives to states a very good paper and no interesting in those papers which have weak contributions. Thus, the performance of POS tagger could not be expected as the previous work done by Turney [25]. We thought that scientific reviews are totally different with entertainment since they have objectivity.

Scores	Vocabulary	Adjectives	Fraction of adjectives
1	10374	705	6.79%
2	52392	3436	6.55%
3	60155	4132	6.86%
4	37031	2448	6.61%
5	14842	993	6.69%
6	3375	240	7.11%

Table 3.6: adjectives in each score

To prove our prediction, we calculate the information gain for each feature, and meanwhile, we list top ten features which have the highest information gain value. These outstanding features are indicated in *Table 3.7*.

Top 10	Information Gain	Features
1	0.01046	this-paper
2	0.00748	liked-i
3	0.06656	encouraging-results
4	0.00649	Use-they
5	0.00546	Is-idea
6	0.00517	Is-contribution
7	0.00508	Convincing-it
8	0.00489	Unclear-it
9	0.00484	Like-i
10	0.00468	Described-approach, subgraphs-the, good-idea, good-give, presented-experiments

Table 3.7: features have high information gain

The *Table 3.8* indicates the best performance of three standard classifiers and also multinomial NB classifier with the corresponding data set. Similarly, those results are represented in three different classification tasks respectively.

Binary-classes classifiers	Data Set	Accuracy
Naive Bayes	subj+mod	70.35%
Multinomial Naive Bayes	subj+comp	**78.91%**
SVM	sub	68.05%
	JJ+NN+VB	
Decision Tree	subj+mod	65.76%

Three-classes classifiers	Data Set	Accuracy
Naive Bayes	subj+comp	67.00%
Multinomial Naive Bayes	subj+mod	**71.90%**
SVM	NN	55.02%
Decision Tree	NN+JJ	53.16%

Six-classes classifiers	Data Set	Accuracy
Naive Bayes	N-gram	34.37
Multinomial Naive Bayes	subj+mod	**48.82%**
SVM	subj+mod+comp	33.87%
	subj+mod	
	mod+comp	
	mod	
	subj	
Decision Tree	N-gram	32.20%

Table 3.8: the best accuracy yielded by classifiers in different models

According to the results in *Table 3.8*, obviously, multinomial NB is the winner. When extending to a multi-class scenario, SVM make use of voting to decide which label should be assigned to. Because we use LibSVM indicated in *Section 2.7*, the strategy of it is one-versus-one (OVO). It will design a SVM between any two samples, thus, it is required *n(n-1)/2* SVMs where *n* is the number of classes. When make a classification for an unknown sample, the class which has the most votes is the final decision of the testing sample. Moreover, the performance of Decision Tree is not good. In this case, we thought there are a couple of reasons:

- Decision Tree can easily overfit. Although we use pruning method to avoid this problem, but this is a grey area.
- Decision Tree can be extremely sensitive to small noisy or perturbations in the data. It means a slight change would result in a drastically outcome.
- Decision Tree also has problems out-of-sample prediction due to there is no smoothing.

Finally, we extend the binary classification problem to a three-class scenario and a six-class problem. In three-class case, we treat the whole training set as three parts: the scores 1 and 2 are labelled as negative, 3 and 4 are neutral class, and 5 and 6 are positive. To six-class classification problem, each score in the training set is as its label. Then, we implement the same process as binary case. The experimental results are shown in Appendix A (Table 3A, 4A, 5A). In *Table 3.9*, we conclude an overall performance of binary, three-class and six-class scenarios with its baseline in bracket. The highest accuracy among these is provided by Parser. Parser is the winner among the other opinion extraction techniques.

Binary Classes	78.91% (baseline 64.76%)
Three Classes	71.90% (baseline 54.90%)
Six Classes	48.82% (baseline 32.88%)

Table 3.9: the best accuracy with baseline

Comparing with [2], their baseline is ranged from 50% to 69%, and for binary classes problem, the accuracy can achieve to 82.9%. The results showing in *Table 3.9* are acceptable that the average of accuracy is higher than the baseline, which approximates to 16%. However, we do not just stop the accuracy here, so we intend to optimal the accuracy in next chapter

We also provide three confusion matrixes in next three figures to detailedly indicate the performance for each classification task:

```
  a   b   <-- classified as
928 116 |   a = negative
224 344 |   b = positive
```

Figure 3.2: confusion matrix for Binary

```
   a   b   c   <-- classified as
 308 196  10 |   a = negative
 118 737  30 |   b = neutral
  10  89 114 |   c = positive
```

Figure 3.3: confusion matrix for three-class

```
a   b   c   d   e   f   <-- classified as
0   2   0   4  39  46 |   a = one
1 206 128  81   5   2 |   b = two
0  86 311 123   6   4 |   c = three
2  22 114 209   6   2 |   d = four
0  10  41  74  39   7 |   e = five
0   0   1   5  14  22 |   f = six
```

Figure 3.4: confusion matrix for six-class

Three tasks	Precision	Recall
Binary	0.785	0.789
Three-class	0.719	0.719
Six-class	0.475	0.488

Table 3.10: Precisions and Recalls

Through the three confusion matrixes and the numbers in *Table 3.10*, we can conclude that the performance of each tasks go beyond the baselines. Because our data set is unbalance for each different class, the predictions are centred on "neutral", especially in our six-class task. The prediction for score 1 is completely failed. The classifier is apt to predict the worst samples as positive class. We use subj+mod that perform the best over the other data set as the model to train the classifier. It means that the model includes both objective and subjective opinions. The reason is complicated, and one the most possible cause maybe that: one hand, scientists would like to encourage the paper authors instead of blaming or shaming, although the paper is weak; on the other hand, the prediction is affected by the attribution named "Three Strong Points of the Paper", or even negation tags such as "no novel", "not clear", etc. The Parser cannot guarantee to extract all modifiers with negation 100% correctly. The classes two (b), three (c) and four (d) is very similar with each other due to the score distances between b and c, and c and d are not easy to discriminating. This result has been proved in the beginning of *Section 3.2*.

3.3 Summary

In this chapter, we found that the multinomial naive bayes outperforms the others. This classifier is able to count the evaluated word frequent in a class. Because this project is different to Spam email filter, the duplications of a feature do not require to be normalised. Every feature can represent the sentiment of a reviewer due to scientists will never produced an advertisement or other irrelevant information in textual area. According to the experiment showing in *Table 3.1*, it describes that it is easier to distinguish the classes which have longer score distance than those

classes that are similar. The following experiment is able to show that appending other textual areas or attributions are helpful to increate the quantity of features or change the weight of features, which makes a higher accuracy. Then, the experiment shows that data preprocessing is necessary when we using N-gram method to build up a data set. Furthermore, we discover a reason why the performance of POS tagger is not very good by implementing a statistical analysis in adjectives. Finally, we implement a large number of experiments from binary to multi-class classes based on the different data set we generated by using a series of opinion extraction techniques. It is noticed that no stemming and stop word list used in POS and Parser when building data set. Meanwhile, we point out that when deal with multi-class cases, SVM make use of voting algorithm to determine the label of a testing instant and the reason why the performance of Decision Tree is not good.

4. Weight Learning through ROC Analysis

In this chapter, we choose the data set which has the highest accuracy in binary classification as the experimental model. Because multinomial NB is significantly better than NB, we implement the optimal operation on this classifier. At the beginning, we start at the binary classification problem. Then, we perform the weight learning for a multi-class scenario. In fact, we use three-label as the example to illustrate the process how to find an optimal weight for each label due to the limited time. However, it is enough to show the solution for multi-class problem.

4.1 An Optimal Threshold in Binary Classification

With respect to a binary classification problem, normally, we use the following *formula (4.1)* to determine which class should be assigned for a testing instance by NB classifier:

$$f(c \mid d) = \begin{cases} \dfrac{f(c_1, d)}{f(c_2, d)} > 1 & c = c_1 \\ \dfrac{f(c_1, d)}{f(c_2, d)} < 1 & c = c_2 \end{cases} \tag{4.1}$$

When the score of class 1 is greater than the score of class 2, we can predict the testing document to be class 1, and otherwise. To the binary classification problem, we import a threshold in this case:

$$f(c \mid d) = \begin{cases} \dfrac{f(c_1, d)}{f(c_2, d)} > threshold & c = c_1 \\ \dfrac{f(c_1, d)}{f(c_2, d)} < threshold & c = c_2 \end{cases} \qquad where \;\; threshold = 1 \tag{4.2}$$

Then, we transfer the *formula (4.2)* with a weight for each class. It means we just want to know either $w_1 f(c_1,d) > w_2 f(c_2,d)$ or $w_1 f(c_1,d) < w_2 f(c_2,d)$. Thus, it is easy to use *formula (4.3)* as the decision criterion.

$$f(c \mid d) = \begin{cases} \dfrac{f(c_1, d)}{f(c_2, d)} > \dfrac{w_2}{w_1} & c = c_1 \\ \dfrac{f(c_1, d)}{f(c_2, d)} < \dfrac{w_2}{w_1} & c = c_2 \end{cases} \qquad where \;\; threshold = \frac{w_2}{w_1} \tag{4.3}$$

In order to find an optimal threshold, ROC curve is a useful approach to do this job. We provide an algorithm to find the threshold that is able to maximise the accuracy or minimise the cost.

```
algorithm findBestThreshold
Inputs: instances i, scores f(i)
Output: threshold t resulting in optimal
        cost/accuracy
  optimum = cost/accuracy assuming all
    instances i are classified as negative
  current = optimum
  sort scores f(i) in decreasing order
  best_threshold = highest score
  for each different score f
    current = update cost/accuracy assuming
      all instances s.t. f(i) > f
      are classified positive
    if current improves on optimum then
      optimum = current
      best_threshold = sqrt(f * next(f))
  return best_threshold
```

Figure 4.1 (from [18]): an algorithm to find an optimal threshold

As showing in *Figure 4.1*, this algorithm will return the best threshold from a list of instances *i* and their scores *f(i)*. Every tried threshold is always generated between two scores f_1 and f_2 by setting it to $sqrt(f_1 f_2)$. The reason of this setting is that the implementation of multinomial NB algorithm, maintains logarithms of the scores, it corresponds to average the log scores. If not, each score value will be too small. *Figure 4.2* is the ROC curve in our binary classification problem. Because this is a very simple method for binary class, we use it on checking the training error. There are a couple of reasons to implement it in this way. One hand, finding the optimal threshold for binary is an easy task due to the degree of freedom is only 1 = (2-1). It means that there is only one "weight" as the threshold will be found. On the other hand, we are not only able to show the performance of this optimal operation, but also to check the training error of data set.

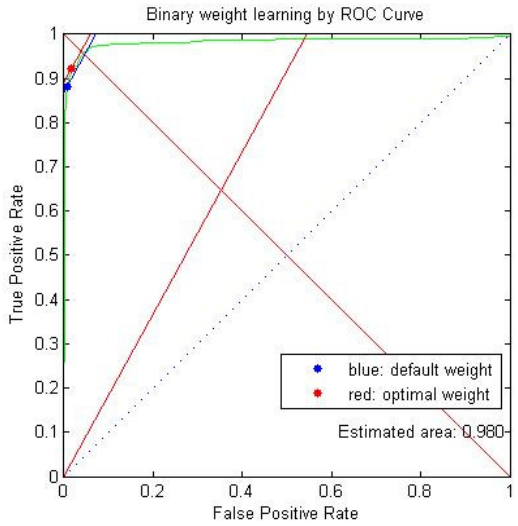


Figure 4.2: the best threshold on training set.

According to *Figure 4.2*, the training error rate is 0.03, when the threshold is set as default 1 (the blue point within the green curve). 0.980 is the Area Under Curve (AUC). Then, the best threshold is found as 1.001 that makes a better accuracy for testing the training set (the red point within the green curve). In this plot, the red line and blue line through the two points are the iso-accuracy line connecting all points in ROC space having the same accuracy. The other red line below the first red line that through the red point is another iso-accuracy line and the crossing point between it and the diagonal line is corresponding to baseline accuracy. This value can be found in y axis by drawing a line paralleling to x axis and through the crossing point. Therefore, *Figure 4.2* gives us a baseline for binary classification problem, and the training error in our data set, and also the performance of setting an optimal threshold for the binary scenario.

4.2 Greedy Fashion for Multi-class Classification

When dealing with multi-class problem, a lazy method is just to use sweeping or exhausted search to find optimal weights which can maximise the accuracy or minimise the cost. This method is not efficient that the time complexity is approximate to $O(i^n)$, where i is the number of instance, and n is the number of different classes. A more efficient approach [18] is to adapt the two-class optimisation to address more than two classes. Thus, the time complexity of each time is $O(i)$. We present the full algorithms in *Figure 4.3* and *Figure 4.4*.

```
algorithm setWeights
Inputs: instances i, actual_classes,
        ordering < on classes
Outputs: weights for all classes and
         corresponding cost/accuracy
  initialise all weights w(P) to 0
  w(1) = 1
  for P=2 to n
    I = []
    for each instance i
      find class Q<P maximising w(Q)*f(Q,i)
      store predicted_classes(i) = Q
      f(i) = f(P,i) / (w(Q)*f(Q,i))
      add i to I
    w(P) = findBestWeight(I,f,P)
  return weights w(P)
end algorithm
```

Figure 4.3(from [18]): the algorithm to set the class weight

The basic idea of weight learning in multi-class problem is to transfer it into pair classes. In *Figure 4.3*, it means that for any two classes P and Q, the only thing is that in this case showing in *Figure 4.3*, if $f(P, i)/f(Q, i) > w_Q/w_P$, we can ensure an instance will not be classified in class Q, but possible in class P. Thus, we only need to determine one weight each time and because the degree of freedom is (n-1), for our three-class scenario in next section, we are required to find 2 = (3-1) weights.

```
algorithm findBestWeight
Inputs: instances i, scores f, class P,
        actual_classes, predicted_classes
Output: weight for class P resulting in
        optimal cost/accuracy
  optimum = cost/accuracy assuming all i
    are classified in predicted_classes(i)
  current = optimum
  sort scores f in decreasing order
  best_weight = highest score
  for each different score f
    current = update cost/accuracy assuming
      all instances s.t. f(i) > f
      are classified in class P
    if current improves on optimum then
      optimum = current
      best_weight = sqrt(f * next(f))
  return best_weight
end algorithm
```

Figure 4.4(from [18]): the algorithm to find the best weight each time

Indeed, this algorithm uses a hill-climbing heuristic to find a good weight. It maps a multi-class to a binary problem. It starts by assigning w_1=1 and the other weighs for other classes are 0. Then, it tries to determine the weight for one class at each time. This algorithm is very efficient that at any stage there are at most *O(i)* possible weights that can influence the prediction. However, a disadvantage of this algorithm is that it cannot guarantee to find a global optimisation.

In our experiment, in order to avoid overfitting problem, we separate the whole data set as ten folders. We use 8 folders as the training set, one folder as the validation set which is used to find the optimal weights, and the last one folder as testing set that is used to test our weights. The detailed experiment results are in Appendix B. We also perform 10-fold cross validation in this situation. *Table 4.1* shows a brief comparison between the default weight and the optimal weight through 10-fold cross validation. They will be evaluated by using t-test in *Section 6.3*.

10-fold	Acc in Default Weights	Acc in Optimal Weights
1	0.56875	0.56875
2	0.575	0.575
3	0.59375	0.59375
4	0.53125	0.53125
5	0.6	0.6
6	0.65625	0.65625
7	0.625	0.61875
8	0.60625	0.60625
9	0.625	0.625
10	0.65625	0.65625

Table 4.1: performance comparison of default weights and optimal weights

Win	Draw	Loss
0	9	1

Table 4.2: reweight vs. default

Table 4.2 is a conclusion for *Table 4.1*. Unfortunately, this optimisation seems to be helpless in our data set. However, we want to refer the experiments implemented by Lachiche and Flach [18]. In their results, there are entirely 24 data sets using reweight approach, and half of them are 1BC and the rest are 1BC2 respectively. *Table 4.3* showing below are their results:

Measure	1BC	1BC2
Number of wins	7-5	6-6
Number of significant wins	4-1	2-2

Table 4.3 (from [18]): reweight vs. default

From the numbers in *Table 4.3*, it does not always improve the accuracy by using this method in multi-class problem. Thus, our outcome seems to be an unlucky one, but we will implement a t-test on those data to show whether it is significant difference. Additionally, although we separate the whole data set as ten parts (8 folders as the training set and the rest two ones are validation set and testing set respectively) to avoid overfitting, the problem is still existing so that the optimum on validation set does not correspond to the optimum on testing set.

5.3 Summary

In this chapter, we mainly implement a weight optimisation for each class in training set. Firstly, we use this method on binary problem and the performance is clearly improved. Then, we extend it to a multi-class problem, meanwhile introduce an algorithm to efficiently find the best weight. Finally, we implement several experiments and also use results done by Lachiche and Flach [18] to interpret the reason why our accuracy is not improved and also give the possible problems existing in our experiments.

5. From Classification to Ranking (and go back again)

In this chapter, we change our model of thinking to let it be more novel. Because the review scores are ranged from 1 to 6, it is definitely a ranking problem. It means a higher score is always better than a lower score. Therefore, initially we deal with our project as a ranking problem instead of the classification problem. Then, a trained ranker is able to sort the instances and we are just to set a boundary threshold to decide the class, which goes back to our project specification - classification problem.

5.1 Training a Ranker by Ranking SVM

At this stage, we use SVMlight with a pairwise algorithm as toolkit to train a ranker. Each bag-of-features is labelled with its score. Thus, the input file example can be created as:

3 qid:1 1:0 2:0 3:1 4:0.1 5:1 #1A
2 qid:1 1:0 2:0 3:1 4:0.1 5:1 #1B
1 qid:1 1:0 2:1 3:0 4:0.4 5:0 #1C
1 qid:1 1:0 2:0 3:1 4:0.3 5:0 #1D
1 qid:2 1:0 2:0 3:1 4:0.2 5:0 #2A
2 qid:2 1:1 2:0 3:1 4:0.4 5:0 #2B

The first value in each line means the relevant value or target value. The second column is a special attribute whose format is qid:#, which means the current instance belongs to a data file #. For example, the first instance above belongs to data file “movie reviews”, while the fifth instance belongs to data file 2 “paper reviews”. In our project, the value of qid is the same due to there is no other classes in our database. The rest attributions are about features that format is [feature_id]:[feature_value]. Thus, let us recall the relevant value in the first value. The relevant value means how relevant between this instance and its data file. Higher value means highly relevant degree. Therefore, we deal with those reviews whose score is 6 as the highest relevant value and we want to sort the values from high to low. SVMlight will use those target values to generate pairwise preference constrains as described in *Section 2.5*. A preference constraint is contained for all pairs of examples in the input file where the target value differs. As the given examples offered above, the following set of pairwise constraints is produced (examples are referred to the string after #):

1A>1B, 1A>1C, 1A>1D, 1B>1C, 1B>1D, 2B>2A

The SVMlight will skip the examples that have the same target values, for example, 1C and 1D, both of them have the same target relevant value 1. It is also noticed that ranks are comparable only between examples with the same value in qid attribution. The target values are only used to define the order of the examples. The ranker will be learned by these target values and also the feature attributions.

5.2 Rank Correction Coefficient

A trained ranker is able to sort the testing instance by scores computed by SVMLight. In the side of ranking problem, if the testing instance is an isolated sample, we can say the ranker is perfect, because there is no error in this ranking case. Of course, if it is a set of testing instances, a perfect ranking is not always successful. With respect to how to evaluate the ranker, we introduce a method called rank correction coefficient. It can be formulised as [28]:

$$cost(\ loss\ \ function\ ) = \begin{cases} y_1 - y_2 > 0 \wedge g(x_1) - g(x_2) < 0 & +1 \quad f(1) \\ y_2 - y_1 > 0 \wedge g(x_2) - g(x_1) < 0 & +1 \quad f(2) \\ otherwise & 0 \quad f(3) \end{cases} \tag{5.1}$$

Also, the error rate based on *formula (5.1)* can be defined as:

$$ER = \begin{cases} \dfrac{f(1) + f(2)}{f(1) + f(2) + f(3)} & if\ |i| \neq 1 \\ 0 & if\ |i| = 1 \end{cases} \tag{5.2}$$

Rank Correction Coefficient is used to take the ranker evaluation. We can make use of cost (loss function) or error rate to estimate the performance of a ranker. For a special case that the testing instance is just a single document, it is clear to see there is no error rate, which means for this case, the performance of the ranker we trained is a perfect one. Note that we can obtain i^2 samples drawn according to $p(x_1, x_2, y_1, y_2)$. It is essential that those samples do not yield i^2 idd samples of the function cost for any g.

In order to test our ranker, we used the 10-fold cross validation as the same as the experiment made in classification problem. The data set we prefer to use is the model that performs best accuracy in binary problem. Because training a ranker on our data set is expensively time cost process and the preliminary work of classification task has spent plenty of time, we deal with it as binary case that score 1-3 have target value 1 and score 4-6 have target value 2. Both score parts have the same qid value. The results of this experiment have shown as following (*Table 5.1*):

Cost (loss function)	56274
ER	21.71%

Table 5.1: cost and error rate

The results in *Table 5.1* illustrate an acceptable performance in this binary task. According to the *formula (5.1)*, if an absolute positive is sorted after a number of negative samples, the cost will be increased quickly. Indeed, not only did the ranker perform a sorted set of documents representing a monotonic relationship among instances, but also can be easily transferred into a classification problem, which is just required a threshold or thresholds. Since our final goal is to transfer it to classification task, the next section we will propose a solution to address this problem.

5.3 Transfer Ranking Problem back to Classification Problem

Considering a sorted set of documents from high to low scores and assuming this is a perfect ranking, it is easily available to determine a threshold that can classify those documents. For instance, there are six documents with score from 6 to 1, and then a perfect ranker is able to range the six documents from 6 to 1 without any error rate. Thus, we can set a threshold between score 3 and 4 to classify the whole testing documents. This approach is very efficient in classification stage, because we can ignore the specific distance between each paired documents. More specifically, we will treat the upper part or bottom part of threshold as a whole respectively. Therefore, the ranking problem is simpler than linear regression problem in this classification task. A roughly classifying process depending on a threshold can achieve our main aim of text classification.

We provide a solution that how to transfer the ranking problem back to classification task. The main task is how to find a suitable threshold that not only can deal with a set of documents, but determine the label for a single document. As a matter of fact, SVMlight provide a possible method to handle this question. The scores for sorting produced by SVMlight are helpful.

Firstly, we use SVMlight to train a ranker which can sort the document/documents with a particular score using for ranking. Furthermore, ROC curve is called for determining a threshold for the binary task. We denote each document in the testing set as *f(d, s)*. The algorithm for finding an optimal threshold can be presented as:

```
Input in ranking step: a testing set with labels
Output in ranking step: every document is followed with a score, denote f(d, s)
Deal with the output file: sort the documents by the scores s from high to low.
    Assume all sorted documents are negative
    Best threshold = the highest score
    Temp = accuracy
    For each different threshold
        Current = update the accuracy assuming all instances s(i)>threshold are classified in class positive
        If Current >= Temp
            Temp = Current
            Best threshold = threshold
    Return threshold
End algorithm
```

Figure 5.1: algorithm to find threshold

According to the algorithm showing in *Figure 5.1*, we implement the experiments step by step and the results shows bellow:

Threshold	Validation Set	Testing Set
0.914	72.5%	62.5%

Table 5.2: accuracy in validation set and testing set by setting an optimal threshold

The threshold value in *Table 5.2* is found by ROC curve. To avoid overfitting, we separate the whole data set as ten parts as the same as we did in weight learning task. There are 8 folders as training set, 1 folder is validation set which is used to find the optimal threshold, and the rest set is used to test the accuracy after the threshold set by validation set. The ROC curve in this task is plotted in *Figure 5.2*.

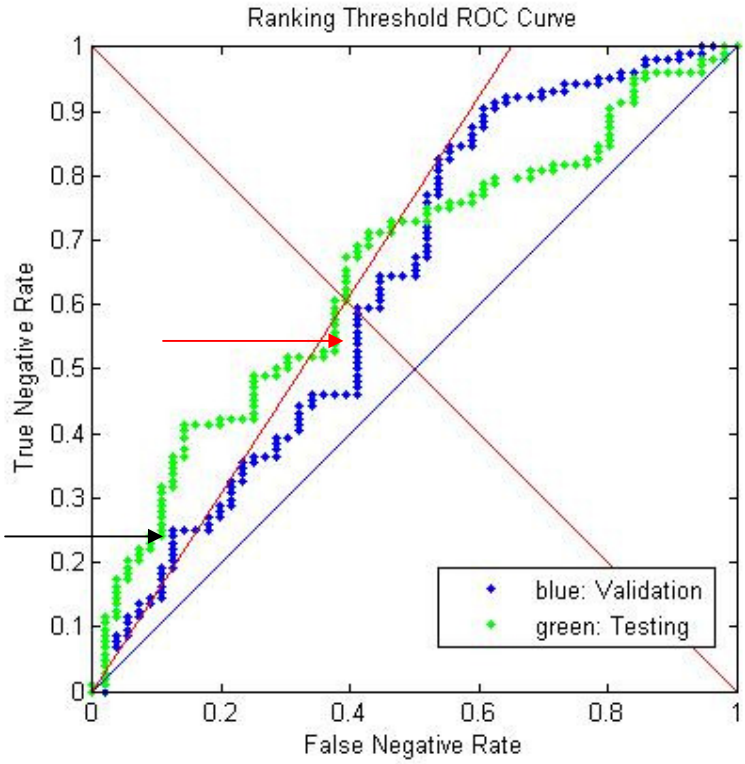


Figure 5.2: ROC curve for validation set and testing set

The red line starting from (0, 0) is baseline iso-accuracy line. The black arrow shows the point that the validation set help us to find the optimal threshold there with the responding to the point pointed by the red arrow in testing set. Indeed, it is able to guarantee to find the best threshold which achieved a best accuracy in validation set, but the corresponding threshold maybe not the

global optimum in testing set. Moreover, comparing with multinomial NB classifier, the performance of this method is worse than the comparer. The average accuracy by setting this threshold can achieve 67.50%, while the performance of traditional SVM algorithm whose accuracy is 67.99%. It is noted that this accuracy made by ranking method is based on 8 folders as training set, while the accuracy made by traditional SVM is based on 9 folders. Thus, this method still has bright future. Most noteworthy of that reducing the error rate in ranking stage is able to increase the accuracy. To train a ranker as good as possible would be meaningful for gaining a high accuracy when transfer it back to classification task.

In multi-class classification aspect, we also can use the algorithms described in weight learning section. Because those testing instances are sorted, it will not affect finding the following thresholds. The instances with a fixed label will not be changed in future. Therefore, a multi-class task also can be transferred into binary task as the same as we did in reweight task. Thus, this method is able to extend into multi-class classification problem.

5.4 Summary

In this chapter, we introduce an interesting method to implement a classification task. If the instances in data set are labelled with multiple scores that can be sorted, we can treat it as ranking problem first. Then, according to the sorted instances, a threshold can be used to determine their labels. We also point out that this approach is able to extend into multi-class classification problem by combining the algorithms (*Figure 4.3, Figure 4.4*) illustrated in reweight section.

6. Experimental Evaluation

Based on a large number of experimental results, it is necessary to make a conclusion of that which classifier or algorithm is significantly different over the others. In this project, we use Friedman test [21] to perform statistical comparisons of classifiers over multiple data sets which are generated by N-gram, POS and Parser. The classifiers include NB, multinomial NB, SVM and DT. After Friedman test, a popular Post-hoc test, Nemenyi test, will be proposed to show the degree of significant for our target classifier, multinomial NB. Those tests are very useful to compare more than two classifiers on multiple data sets. With respect to the optimal weight learning, we are going to use paired T-test to examine whether the optimum is significant or not.

6.1 Friedman Test over Multiple Data Sets

It is good at performing statistical comparisons of classifiers over multiple data sets. The idea is that it ranks the algorithms for each data set separately, for example, the best algorithm which outperforms the others awarding the first rank, the second best is ranked as 2. In the case of draw, average ranks are assigned, for example, algorithm A and algorithms B are the same performance, which are equal to rank 2, then the real rank value of them is 2.5 = (2+3)/2, where one is rank 2 and the other is rank 3, and the rest rank starts 4 rather than 3.

The basic idea of Friedman test is that let *r(j, i)* be the rank of the *j*-th of *K* algorithms or classifiers on the *i*-th of *N* data sets. For each algorithm, Friedman test computes the average ranks, denoting as:

$$\overline{R}_j = \frac{1}{N}\sum_i r(j,i) \tag{6.1}$$

Under the null-hypothesis which assumes that all the target algorithms are equivalent. Therefore, their average ranks $\overline{R}_j$ should be equal as well. The *formula (6.2)* is the Friedman statistic that is distributed according to X_F^2 with *k-1* degrees of freedom, when *K* and *N* are big enough (usually as a rule of a thumb, *K*>5 and *N*>10).

$$\chi_F^2 = \frac{12N}{K(K+1)}\left[\sum_j \overline{R}_j^2 - \frac{K(K+1)^2}{4}\right] \tag{6.2}$$

Alternatively, we substitute the average ranks by *formula (6.1)*, M (chi-square) from the *formula (6.3)* can be represented as:

$$M = \frac{12}{NK(K+1)}\sum_j R_j^2 - 3N(K+1) \tag{6.3}$$

As the same as *formula (6.2)*, *K* is the number of columns representing as algorithms, *N* is the number of rows often called blocks indicating as data sets. The main difference here is the *R*

which is the sum of the ranks in column *j* or algorithm *j*, instead of the average ranks of the *j*-th algorithm in *formula (6.2)*.

In *formula (6.3)*, if there is a significant difference between the sum of the ranks of each of the algorithms, then M will be large, or otherwise, and also if at least one algorithm shows significant difference then the value of M will be larger as well. The significance of M may then be looked up in tables. If M > critical value, then we can reject null hypothesis and conclude that the algorithm has a significant effect on the data sets.

Because we have used 4 algorithms to deal with the classification problem, it means the degree of freedom is 3 (K-1 = 4–1). The whole data sets are 89 and the data sets where there are at least two accuracies generated by any two algorithms must be higher than the baseline are 45, respectively. Thus, we implement Friedman test based on each situation. After computing the M value, we can look up the chi-square table to check whether the difference is significant or not.

H_0 (null hypothesis) all algorithms are equivalent.
H_1 (alternative hypothesis): all algorithms are not equivalent

#Data Sets	M Value
89	109.0213
45	95.9933

Table 6.1: M value for each different number of data sets

DF	0.05	0.01	0.001
3	7.81473	11.3449	16.266

Table 6.2: Chi-square critical value table

The two tables showing above indicate that the two different numbers of data sets are both greater than the critical value. Thus, we are able to reject the null hypothesis and choose the alternative hypothesis.

6.2 Post-hoc Nemenyi Test after Friedman Test

After performing Friedman test and if the null hypothesis is rejected, we can proceed with the corresponding post-hoc Nemenyi test [24]. This test is similar to the Tukey test for ANOVA and is used when all classifiers are compared to each other. The critical difference can be defined as:

$$CD = q_\alpha \sqrt{\frac{K(K+1)}{6N}} \tag{6.4}$$

It means if the corresponding average ranks differ by at least the CD, the performance of two classifiers is significantly different. The q_α table is presented below:

#classifiers	2	3	4	5	6	7	8	9	10
$q_{0.05}$	1.960	2.343	**2.569**	2.728	2.850	2.949	3.031	3.102	3.164
$q_{0.10}$	1.645	2.052	**2.291**	2.459	2.589	2.693	2.780	2.855	2.920

Table 6.3: critical values for the two-tailed Nemenyi test

We have 4 classifiers to compare, so when α = 0.05, the critical value = 2.569 and when α = 0.10, the critical value = 2.291. According to the *formula (6.4)*, we can easily calculate the critical difference by substituting the q_α. Thus, we have:

#data sets	89	45
$q_{0.05}$	0.4972	0.6992
$q_{0.10}$	0.4434	0.6235

Table 6.4: CDs

classifiers	NB	Multinomial NB	SVM	DT
89	3.0056	2.2135	1.4944	3.2978
45	2.8778	1.3556	1.9778	3.8111

Table 6.5: mean rank for each classifier based on the different numbers of data sets

The cells in *Table 6.5* indicate the average rank for each classifier. Based on the CDs showing in *Table 6.4*, the difference between the mean ranks of two classifiers can be computed to determine whether we can reject the null hypothesis. We usually use the Nemenyi test for pairwise comparisons, thus the *Table 6.6* shows the difference of pairwise:

pairwise	NB-MNB	NB-SVM	NB-DT	MNB-SVM	MNB-DT	SVM-DT
89	0.7921	1.5112	0.2922	0.7197	1.0843	1.8034
45	1.5222	0.9000	0.9333	0.6222	2.4555	1.8333

Table 6.6: pairwise comparisons

In *Table 6.6*, we can conclude that the value in cell which is greater than the corresponding CDs is powerful enough to detect the significant differences between the algorithms.

6.3 T-test on Weight Learning Results

Paired T-test is widely used for comparisons of two classifiers. This method can test whether the difference between two classifiers' results over many data sets is non-random. Here, we use the values provided in *Table 4.1*, and offer the parameters (*Table 6.7*) by computing the mean and standard variance for both algorithms. Then, after calculating the t value by substitute those parameters, we look up the p-value table to make a decision. The t value can be computed as:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{S_{\bar{x}_1 - \bar{x}_2}} \tag{6.5}$$

Algorithms	Acc in Default Weights	Acc in Optimal Weights	
Mean	0.6037	0.6031	
Standard variance	0.0392	0.0389	
t value			0.3434

Table 6.7: means and standard variances

The t value can be computed by the *formula (6.5)*. There are 10 values for each algorithm, therefore, the degree of freedom is (10-1) + (10-1) = 18. The corresponding confidence level shows below:

DF	0.2	0.1	0.05	0.01	0.005	0.001	0.0005	0.0001
18	1.33	1.73	2.10	2.88	3.20	3.92	4.23	4.97

Table 6.8: critical values in degree of freedom 18

From the critical values in different confidence levels, we can conclude that there is no significant different between optimum or non-optimum operations. The reason may be that uniform weights are optimal and that the predicted probabilities are calibrated, which is hard to believe for multinomial Naive Bayes.

6.4 Summary

In this chapter, we performed Friedman test over multiple data sets and the corresponding post-hoc Nemenyi test. In fact, we constructed two parts as testing data sets: the first one is the whole data sets we generated; the second one is refined that only includes those data sets which at least have two accuracies higher than its baseline. We can conclude that all algorithms are not equivalent. However, for the first part we also can conclude that there are significant difference between Multinomial NB and SVM, while for the second part, there is no power to prove that Multinomial NB and SVM is different.

After that, we performed t-test on reweighting results. Unfortunately, we cannot conclude that they have significant difference according to the calculated result.

7. Conclusion

The main aim of this project is to use machine learning algorithms to train several classifiers to predict the class of reviews. A general procedure of this project can be indicated in *Figure 7.1*. In the aspect of feature extraction, we used some natural language processing techniques such as N-gram, POS tagger and relations parser. According to the experimental results, we discussed the pros and cons of classifiers and the methods of feature extraction, meanwhile, analysed the implicated information of data based on statistical method. During the experimental stage, we found that the Multinomial Naive Bayes shows an outstanding performance. Thus, based on the best accuracy made by Multinomial NB, we implement a reweight process that to use an efficient algorithm to find a better weight for each class by ROC curve. Moreover, we proposed a novel and interesting approach to achieve the classification problem. We initially treated the project as a ranking problem due to the scores from 1 to 6 in data set. Then we used SVMlight to train a ranker that can sort the data sets by their score. Eventually, we transferred the ranking problem back to our classification problem by setting a threshold using ROC curve. At last but not least, we did the Friedman test over the multiple data sets we created during the experiments stage with the corresponding post-hoc Nemenyi test, and also the paired t-test was implemented to test whether the difference between the optimum and non-optimum operation of weight over various data sets is non-random.

Figure 7.1: the general process of project

8. Future Work

Although our project has achieved an acceptable accuracy for each classification task, there are still many aspects that could be improved. Firstly, since the feature extraction is based on sentence level, the correction of a sentence will deeply affect the features. Lingpipe application is helpful to extract sentences from a chunk of textual area, but there are still many errors. A solution that we manually correct the sentences after processing by Lingpipe may be useful. Secondly, SO method introduced in *Section 2.8.1* can be used to choose those features which are able to represent a closer meaning to the anchor words such as “good” or “bad”. Thirdly, some other data sets also can be used to implement the weight learning or transfer into ranking problem. Finally, the ranker can be trained by using soft-margin SVM, which can minimise the cost or error.

The big limitations of this work include the time and applications. We spend a lot of time parsing the relations from sentences. Each data set created by parser almost cost seven or eight hours. The time difficulty of using parser is eliminated by progress in hardware aspect.

Reference

[1] Brill, E. 1994. Some advances in transformation-based part of speech tagging. Proceedings of the Twelfth National Conference on Artificial Intelligence (pp. 722-727). Menlo Park, CA: AAAI Press.

[2] B. Pang, L. Lee, and S. Vaithyanathan, “Thumbs up? Sentiment classification using machine learning techniques” in Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 79–86, 2002.

[3] B. Pang and L. Lee, Opinion Mining and Sentiment Analysis, Foundations and Trends in Information Retrieval, Vol.2, Nos. 1-2 (2008) 1-135, 2008.

[4] Carpenter, B. (2005). Scaling high-order character language models to gigabytes. In Proceedings of the 2005 association for computational linguistics software workshop (pp. 1–14). Ann Arbor: ACL.

[5] Carterette, B.; Bennett, P. N.; Chickering, D. M.; and Dumais, S. T. 2008. Here or there: Preference judgments for relevance. In Proceedings of the European Conference on Information Retrieval (ECIR).

[6] Cer, D., M.-C. de Marneffe, D. Jurafsky, and C. Manning. 2010. Parsing to stanford dependencies: Trade-offs between speed and accuracy. In LREC 2010.

[7] Charniak, Eugene, Introduction to artificial intelligence, page 2. Addison-Wesley, 1984.

[8] Chris Bourke, Kun Deng, Stephen D. Scott, Robert E. Schapire, N. V. Vinodchandran: On reoptimizing multi-class classifiers. Machine Learning 71(2-3): 219-242 ,2008.

[9] Christopher D. Manning, Prabhakar Raghavan & Hinrich Schutze, Introduction to Information Retrieval, Cambridge Univeristy Press, 2008.

[10] E. G. V´azquez, A. Y. Escolano, and J. P. Junquera P. G. Ria˜no. Repeated measures multiple comparison procedures applied to model selection in neural networks. In Proc. of the 6th Intl. Conf. On Artificial and Natural Neural Networks (IWANN 2001), pages 88–95, 2001.

[11] Hatzivassiloglou, V., & McKeown, K.R. 1997. Predicting the semantic orientation of adjectives. Proceedings of the 35th Annual Meeting of the ACL and the 8th Conference of the European Chapter of the ACL (pp. 174-181). New Brunswick, NJ: ACL.

[12] Hatzivassiloglou, V., & Wiebe, J.M. 2000. Effects of adjective orientation and gradability on sentence subjectivity. Proceedings of 18th International Conference on Computational Linguistics. New Brunswick, NJ: ACL.

[13] J Demšar, Statistical comparison of classifiers over multiple data sets. Journal of Machine Learning Research, 7:1–30, 2006.

[14] J. Pizarro, E. Guerrero, and P. L. Galindo. Multiple comparison procedures applied to model selection. Neurocomputing, 48:155–173, 2002.

[15] K. Dave, S. Lawrence, and D. M. Pennock, “Mining the peanut gallery: Opinion extraction and semantic classification of product reviews,” in Proceedings of WWW, pp. 519–528, 2003.

[16] Kristina Toutanova and Christopher D. Manning. 2000. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora

(EMNLP/VLC-2000), pp. 63-70.
[17] Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In Proceedings of HLT-NAACL 2003, pp. 252-259.
[18] Lachiche, N., & Flach, P. (2003). Improving accuracy and cost of two-class and multi-class robabilistic classifiers using ROC curves. In Proceedings of the 20th international conference on machine learning (pp. 416–423).
[19] Mara-Florina Balcan, Alina Beygelzimer, John Langford, and Gregory B. Sorkin, Robust reductions from ranking to classification, Tech. report, IBM Research Report, November 2006.
[20] Marie-Catherine de Marneffe, Bill MacCartney and Christopher D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In LREC 2006.
[21] M. Friedman. A comparison of alternative tests of significance for the problem of m rankings. Annals of Mathematical Statistics, 11:86–92, 1940.
[22] M. Hu and B. Liu, "Mining and summarizing customer reviews", International Conference on Knowledge Discovery and Data Mining, Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 168 – 177, 2004.
[23] P. Beineke, T. Hastie and S. Vaithyanathan, "The sentimental factor: improving review classification via human-provided information", Annual Meeting of the ACL, Proceedings of the 42nd Annual Meeting on Association for Comutational Linguistics, Barcelona, Spain, article no: 263, 2004.
[24] P. B. Nemenyi. Distribution-free multiple comparisons. PhD thesis, Princeton University, 1963.
[25] P. Turney, "Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews", in Proceedings of the Association for Computational Linguistics (ACL), pp. 417–424, 2002.
[26] Quinlan, J. R. Induction of decision trees. Machine Learning, 1(1). 81-106, 1986.
[27] Q. Ye, Z. Zhang and R. Law, "Sentiment classification of online reviews to travel destinations by supervised machine learning approaches", Expert Systems with Applications, Volume 36, Issue 3, Part 2, pages 6527 – 6535, April 2009.
[28] R. Herbrich, T. Graepel, and K. Obermayer. Large Margin Rank Boundaries for Ordinal Regression. Advances in Large Margin Classifiers, 115-132, Liu Press, 2000.
[29] Richard Mankiewicz, The Story of Mathematics (Princeton University Press), p.158.
[30] S. Cl´emen¸con, G. Lugosi, N. Vayatis. Ranking and Scoring Using Empirical Risk Minimization, Proceedings of the Eighteenth Annual Conference on Computational Learning Theory (COLT), 1–15, 2005.
[31] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In In European Conference on Machine Learning (ECML), 1998.
[32] T. Joachims, Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.
[33] T. Joachims, Optimizing Search Engines Using Clickthrough Data, Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD), ACM, 2002.

[34] Turney, P.D. 2001. Mining the Web for synonyms: PMI-IR versus LSA on TOEFL. Proceedings of the Twelfth European Conference on Machine Learning (pp. 491-502). Berlin: Springer-Verlag.
[35] Wiebe, J.M. 2000. Learning subjective adjectives from corpora. Proceedings of the 17th National Conference on Artificial Intelligence. Menlo Park, CA: AAAI Press.
[36] Wiebe, J.M., Bruce, R., Bell, M., Martin, M., & Wilson, T. 2001. A corpus study of evaluative and speculative language. Proceedings of the Second ACL SIG on Dialogue Workshop on Discourse and Dialogue. Aalborg, Denmark.

Appendix A

Model Name	NB	multinomialNB	LibSVM	Decision Tree
Tagged-JJ	58.50%	60.55%	67.80%	60.79%
Tagged-NN	59.43%	58.68%	67.80%	57.94%
Tagged-VB	59.42%	58.31%	67.80%	60.92%
Tagged-RB-JJ	58.87%	58.75%	67.74%	60.86%
Tagged-JJ-NN	59.86%	56.89%	67.87%	60.98%
Tagged-NN-VB	58.80%	58.56%	67.87%	57.13%
Tagged-RB-VB	58.93%	58.75%	67.93%	61.41%
Tagged-NN-VB-RB	59.24%	57.63%	68.00%	58.75%
Tagged-JJ-NN-VB	59.12%	58.00%	68.05%	57.63%
Tagged-JJ-RB-VB	59.37%	56.95%	67.93%	60.79%
Tagged-JJ-NN-VB-RB	59.31%	56.95%	68.05%	57.07%
Bigram without preprocess	62.16%	59.18%	67.62%	60.86%
Bigram with preprocess	67.31%	66.19%	67.87%	64.70%
Bigram-sub	65.88%	66.81%	66.07%	66.44%
Bigram-mod	66.87%	69.10%	68.11%	65.76%
Bigram-comp	65.26%	69.11%	66.13%	66.50%
Bigram-sub-mod	67.06%	75.87%	67.99%	64.76%
Bigram-sub-comp	67.00%	69.91%	67.99%	65.38%
Bigram-mod-comp	67.43%	75.68%	68.11%	65.88%
Bigram-sub-mod-comp	67.31%	65.76%	67.87%	64.27%

Table 1A (Binary): only include the content of “Detailed Comments to the Authors”

Model Name	NB	multinomialNB	LibSVM	Decision Tree
Bigram without preprocess	63.03%	60.42%	67.62%	59.93%
Bigram with preprocess	67.37%	67.56%	67.80%	61.72%
Bigram-sub	67.18%	73.14%	68.11%	65.76%
Bigram-mod	66.44%	76.86%	67.80%	64.33%
Bigram-comp	65.32%	69.54%	66.13%	65.69%
Bigram-sub-mod	66.94%	73.01%	67.93%	62.78%
Bigram-sub-comp	66.75%	77.23%	68.05%	64.45%
Bigram-mod-comp	65.51%	69.42%	67.87%	63.15%
Bigram-sub-mod-comp	66.44%	72.95%	67.93%	62.84%

Table 2A (Binary): include the content of “Detailed Comments to the Authors” attribute and “A Brief Summary of the Main Contributions of the Paper”

Model Name	NB	multinomialNB	LibSVM	Decision Tree
Tagged-JJ	58.50%	60.55%	67.80%	60.79%
Tagged-NN	59.43%	58.69%	67.80%	57.94%
Tagged-VB	59.43%	58.31%	67.80%	60.92%
Tagged-RB-JJ	58.87%	58.75%	67.74%	60.86%
Tagged-JJ-NN	59.86%	56.89%	67.87%	60.98%
Tagged-NN-VB	58.81%	58.57%	67.87%	57.13%
Tagged-RB-VB	58.93%	58.75%	67.93%	61.41%
Tagged-NN-VB-RB	59.24%	57.63%	67.99%	58.75%
Tagged-JJ-NN-VB	59.12%	58.00%	68.05%	57.63%
Tagged-JJ-RB-VB	59.37%	57.26%	67.87%	59.24%
Tagged-JJ-NN-VB-RB	59.31%	56.95%	68.05%	57.07%
Bigram without preprocess	67.00%	63.34%	67.68%	61.29%
Bigram with preprocess	68.92%	69.91%	67.93%	60.98%
Bigram-sub	66.53%	75.25%	68.05%	64.76%
Bigram-mod	65.63%	75.43%	67.80%	65.01%
Bigram-comp	65.51%	68.92%	67.80%	64.64%
Bigram-sub-mod	70.35%	75.50%	67.93%	65.76%
Bigram-sub-comp	67.31%	78.91%	64.76%	67.99%
Bigram-mod-comp	65.77%	76.36%	67.87%	64.82%
Bigram-sub-mod-comp	70.04%	70.47%	63.65%	67.93%

Table 3A (Binary): include the content of “Detailed Comments to the Authors” and “A Brief Summary of the Main Contributions of the Paper” and “3 strong and weak points”

Model Name	NB	multinomialNB	LibSVM	Decision Tree
Tagged-JJ	45.29%	43.36%	54.90%	48.08%
Tagged-NN	40.76%	54.28%	55.02%	46.09%
Tagged-VB	41.44%	49.01%	54.78%	49.13%
Tagged-RB-JJ	42.25%	53.16%	54.90%	45.60%
Tagged-JJ-NN	41.56%	47.70%	54.59%	45.84%
Tagged-NN-VB	40.45%	48.26%	54.59%	46.77%
Tagged-RB-VB	38.77%	51.30%	54.47%	48.39%
Tagged-NN-VB-RB	38.15%	47.83%	54.59%	48.33%
Tagged-JJ-NN-VB	40.69%	43.98%	54.53%	41.54%
Tagged-JJ-RB-VB	39.33%	45.29%	54.53%	46.15%
Tagged-JJ-NN-VB-RB	39.02%	44.29%	54.59%	45.04%
Bigram without preprocess	51.67%	51.99%	54.40%	49.57%
Bigram with preprocess	53.91%	66.56%	54.96%	50.74%
Bigram-sub	48.88%	66.32%	54.90%	47.64%

Bigram-mod	51.55%	67.43%	54.90%	48.26%
Bigram-comp	52.17%	62.41%	54.90%	51.67%
Bigram-sub-mod	52.23%	67.93%	54.90%	46.22%
Bigram-sub-comp	67.00%	60.98%	54.90%	48.33%
Bigram-mod-comp	51.43%	67.62%	54.90%	45.91%
Bigram-sub-mod-comp	51.05%	66.01%	54.90%	46.40%

Table 4A (Three-class): include the content of "Detailed Comments to the Authors" and "A Brief Summary of the Main Contributions of the Paper" and "3 strong and weak points"

Model Name	**NB**	**multinomialNB**	**LibSVM**	**Decision Tree**
Tagged-JJ	26.55%	21.65%	32.94%	26.48%
Tagged-NN	26.80%	30.89%	32.57%	27.42%
Tagged-VB	25.74%	21.53%	33.19%	26.67%
Tagged-RB-JJ	25.50%	22.64%	32.94%	27.54%
Tagged-JJ-NN	27.85%	30.09%	32.75%	26.74%
Tagged-NN-VB	26.74%	30.96%	32.69%	26.36%
Tagged-RB-VB	24.50%	26.67%	32.94%	25.50%
Tagged-NN-VB-RB	26.49%	30.96%	32.69%	27.79%
Tagged-JJ-NN-VB	27.36%	31.33%	33.06%	27.92%
Tagged-JJ-RB-VB	25.93%	29.78%	32.88%	26.18%
Tagged-JJ-NN-VB-RB	27.05%	31.51%	32.69%	28.35%
Bigram without preprocess	32.75%	38.28%	32.82%	26.61%
Bigram with preprocess	34.37%	43.18%	33.68%	32.20%
Bigram-sub	32.01%	43.98%	33.87%	29.22%
Bigram-mod	29.40%	44.54%	33.87%	30.58%
Bigram-comp	30.46%	31.95%	33.75%	30.46%
Bigram-sub-mod	31.64%	48.82%	33.87%	29.09%
Bigram-sub-comp	32.01%	44.35%	33.75%	28.35%
Bigram-mod-comp	29.65%	45.72%	33.87%	29.28%
Bigram-sub-mod-comp	32.20%	44.79%	33.87%	29.28%

Table 5A (six-class): include the content of "Detailed Comments to the Authors" and "A Brief Summary of the Main Contributions of the Paper" and "3 strong and weak points"

Appendix B

Default weights			
Negative Weight	**Neutral Weight**	**Positive Weight**	**Accuracy**
1	1	1	0.56875
1	1	1	0.575
1	1	1	0.59375
1	1	1	0.53125
1	1	1	0.6
1	1	1	0.65625
1	1	1	0.625
1	1	1	0.60625
1	1	1	0.625
1	1	1	0.65625

Table 1B: default weight without optimum

Reweight by ROC curve			
Negative Weight	**Neutral Weight**	**Positive Weight**	**Accuracy**
1	1.001299980025966	1.0370491213052542	0.56875
1	1.0280982443327096	1.0279980544728673	0.575
1	1.0053499689162972	1.0270999561873226	0.59375
1	0.9913999546096419	1.024449237395392	0.53125
1	0.9910499987387115	1.0160493491952052	0.6
1	0.9981499987476833	1.0364996092618655	0.65625
1	1.0008497889293877	1.0368999951779343	0.61875
1	0.9998996799679456	1.0461	0.60625
1	1.014549900202055	1.0173494581509344	0.625
1	1.00564996892557	1.0281996061076857	0.65625

Table 2B: weight learning by ROC curve