Semi-Supervised Learning for Text Classification

B. Tech Project Report

By

Anmol Chaudhary (1310110059)

Iti Chauhan (1310110151)

Toshan Majumdar (1310110343)

Supervisor: Dr. Dolly Sharma

Assistant Professor

Department of Computer Science and Engineering

Submitted in the partial fulfillment of requirements for B. Tech. in Computer Science and Engineering

SHIV NADAR UNIVERSITY

Department of Computer Science and Engineering,

School of Engineering, Shiv Nadar University,

Gautam Buddha Nagar, U.P., India, 201314

www.snu.edu.in

Approval Sheet

This report titled Semi-Supervised Learning for Text Classification by Anmol Chaudhary, Toshan Majumdar and Iti Chauhan is approved for the degree of B. Tech in Computer Science and Engineering.

Project Advisor(s)

Name	Dr. Dolly Sharma
Signature	________________________________
Date	________________________________

Declaration Sheet

We declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Name: Anmol Chaudhary Signature___________________________

Date: _____________________________

Name: Iti Chauhan Signature___________________________

Date _____________________________

Name: Toshan Majumdar Signature___________________________

Date _____________________________

Abstract

Semi-supervised learning for text classification, which employs a small labelled data and a large unlabelled data is seen as a promising technique to reduce the labour-intensive and time consuming effort of labelling training data in order to build accurate classifiers since unlabelled data is easy to get. Meanwhile unlabelled data may be easier to collect, there have been only few ways to utilise them. Semi-supervised learning addresses this problem by using a large amount of unlabelled data with the labelled data, to build better classifiers. Because semi-supervised learning requires less human effort and produces better results, it is of high interest in both practice and theory.

Our research project is an implementation of several semi-supervised learning techniques based on the Expectation Maximisation model and using Bag of Words and Bag of Centroids using K-means for feature selection. The main idea is to understand the dependence of the classifier's performance on the type and quality of data and on the feature selection technique employed. We test the various semi-supervised text classification techniques with several different sized data sets. The classifiers are evaluated on the basis of several performance metrics. Experiment results show our implementations are efficient and comparable.

Acknowledgements

We would like to thank our project guide and mentor **Dr. Dolly Sharma** who helped us clarify our doubts and led us in the right direction throughout the entire project phase. The constant support and constructive criticism provided by her helped us shape our ideas and knowledge base into a well-researched project with insightful results.

We would also like to thank the entire computer science faculty at SNU especially **Dr. Debopam Acharya, Dr. Anirban Mondal** and **Mr. Rajeev Kumar Singh** for their useful insights and advice during the course of the project work as well as their teaching prowess which helped us nurture a deep understanding of the basic concepts of computer science, a strong foundation on which we based our entire project on.

The final year project was a truly informative and enjoyable experience right from the conceptualization and selection of the topic till the compilation of results and project documentation. This was only possible through the support of the various people involved as well as the coordination and effort of the core team members.

We hope that this project report is an accurate description of the work done towards Semi-Supervised Learning for Text Classification and truly represents the depth of research and effort taken by the core team members.

We cannot thank enough our families and friends whose unconditional love and support made our journey at Shiv Nadar University so wonderful.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

SSL	Semi-Supervised Learning
SVM	Support Vector Machine
TSVM	Transductive Support Vector Machine
LDS	Low Density Separation
TP	True Positives
TN	True Negatives
FP	False Positives
MLL	Marginal Log Likelihood
CLL	Conditional Log Likelihood
MNB	Multinomial Naïve Bayes
BOC	Bag of Centroids
AUC	Area Under ROC Curve
EM	Expectation-Maximization
MLP	Multilayer Perceptron
NLTK	Natural Language Toolkit
BOW	Bag of Words
TF-IDF	Term Frequency – Inverse Document Frequency
NB	Naïve Based
PEBL	Psychology Experiment Building Language

Chapter 1

Introduction

Today we are at the defining moment in the evolution and growth of the Internet. The Internet is a busy place. Every second, approximately 6,000 tweets are tweeted; more than 40,000 Google queries are searched and more than 2 million emails are sent, according to Internet Live Stats, a website of the international Real Time Statistics Project. With about 1 billion websites, the Web is home to many more individual Web pages, there is an immense unprocessed data available on the internet.

This un-structured and unlabeled data is not of much usage and provides no meaningful information to the users. An effort to add value to this raw data requires expertise in data mining. Since, most of the data mining techniques mainly are oriented towards well-labeled data, which is neither easily available nor free of cost. Labeled data comes with its own construction and maintenance cost. And the time and labor required to process and label data also makes it a disadvantage to use it excessively for data mining and analytics. Therefore, the supervised technique (working with only labeled data) did not prove to be very convenient. While searching for alternatives to overcome these advantages data scientist has come up with a new learning technique called: The Semi-Supervised Learning Technique.

1.1 Background

1.1.1 Semi-Supervised Learning Technique

Semi-supervised learning is a class of supervised learning tasks and techniques that also make use of unlabeled data for training – typically a small amount of labeled data with a large amount of unlabeled data. Semi-supervised learning falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data). Many machine-learning researchers have found that unlabeled data, when used in conjunction with a small amount of labeled data, can produce considerable improvement in learning accuracy. The acquisition of labeled data for a learning problem often requires a skilled human agent (e.g. to transcribe an audio segment) or a physical experiment (e.g. determining the 3D structure of a protein or determining whether there is oil at a particular location). The cost associated with the labeling process thus may render a fully labeled training set infeasible, whereas acquisition of unlabeled data is relatively inexpensive. In such situations, semi-supervised learning can be of great practical value. Semi-supervised learning is also of theoretical interest in machine learning and as a model for human learning.

The semi-supervised learning technique is a newly emerging in the field of artificial intelligence therefore it held our interest towards it to a great extent. Semi-supervised learning is motivated by the fact of building better systems with greater accuracy by making use of less expensive and more logical training set. In our final year project, we aim to study the vitalities of text classification, find the right dataset with labeled and unlabeled data, which will allow us to develop semi-supervised learning techniques and algorithms that will be an improvement of any supervised learning algorithm. Finally, we would like to compare the results of our semi-supervised approach to the already existing approaches.

1.1.2 Types of Semi-Supervised Learning

A variety of methods have been developed for transductive SSL. These methods can be grouped as: EM with generative mixture models, bootstrapping methods discriminative models (Transductive Support Vector Machines (TSVM)) and data based methods, including Manifold Regularization, Information Regularization, and Low Density Separation(LDS). Specifically, TSVM extends the maximum margin principle of SVM to unlabeled data. It combines the regularization of SVMs on the labeled points with the cluster assumption on the unlabeled points, to enforce the decision boundary to lie in low density regions.

In general, the essential distinction between transductive learning and inductive learning is that transductive learning produces labels only for the available unlabeled data; while inductive learning not only produces labels for the unlabeled data, but also learns a classifier that can be used to predict labels for new data. In this sense, some SSL algorithms, though named as “transductive”, have an inductive nature.

1.1.3 Text Classification

Text Classification is a data mining technique which assigns one or more classes to a document according to their content, the classes selected from a previously established taxonomy. As most information (over 80%) is stored as unstructured text, text mining is believed to have a high commercial potential value.

The text mining studies are gaining more importance recently because of the availability of the increasing number of the electronic documents from a variety of sources which include unstructured and semi structured information. The main goal of text mining is to enable users to extract information from textual resources and deals with the operations like, retrieval, classification (supervised, unsupervised and semi supervised) and summarization, and utilization of natural Language Processing (NLP), data Mining, and Machine Learning techniques.

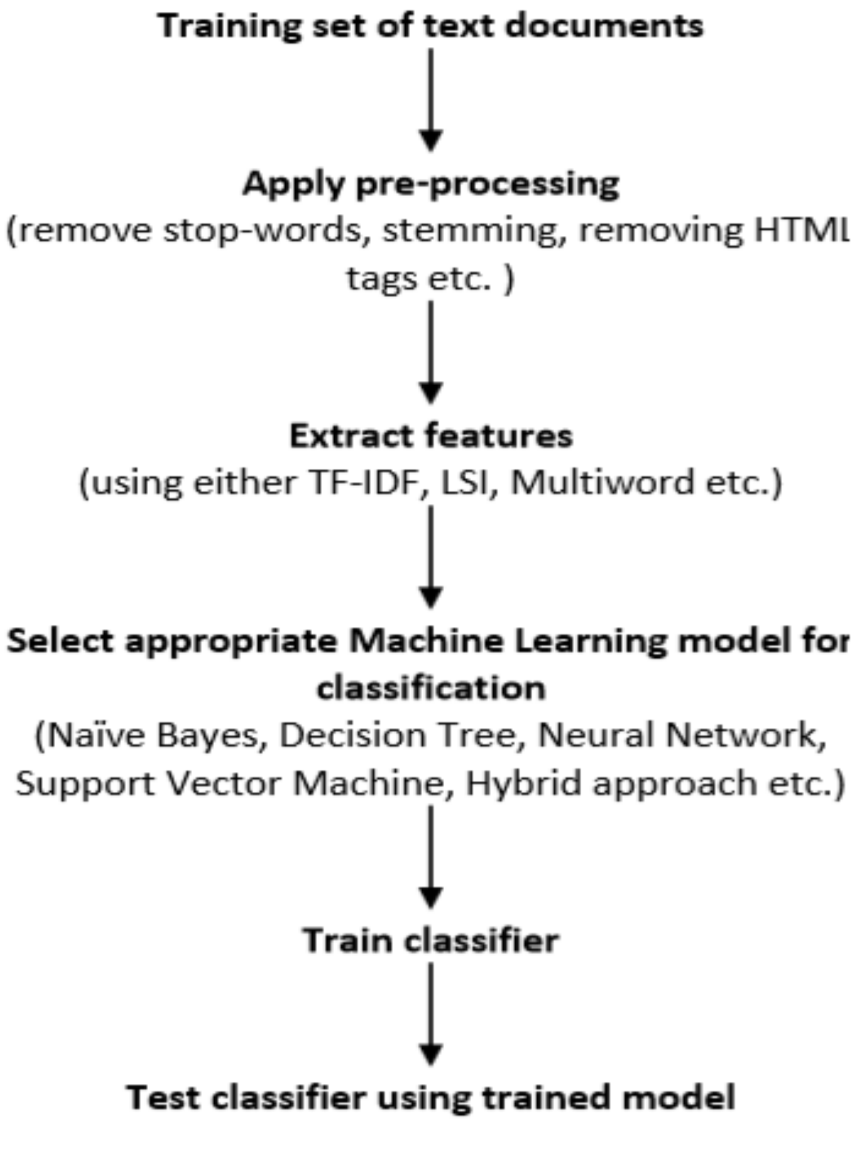

Training set of text documents
Apply pre-processing
(remove stop-words, stemming, removing HTML tags etc.)
Extract features
(using either TF-IDF, LSI, Multiword etc.)
Select appropriate Machine Learning model for classification
(Naïve Bayes, Decision Tree, Neural Network, Support Vector Machine, Hybrid approach etc.)
Train classifier
Test classifier using trained model

Fig 1.1.1 Text classification strategy [2]

1.2 Fundamentals

Pre-processing:

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing.

Indexing

The indexing techniques in multidimensional databases are becoming more and more important. The data warehouse environment has become a popular technology and a lot of research work has been done in order to improve the query performance and to provide faster response times.

Feature selection

In machine learning and statistics, feature selection, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. The central premise when using a feature selection technique is that the data contains many features that are either redundant or irrelevant, and can thus be removed without incurring much loss of information.

Performance measures

Important for understanding the quality of the model/ technique. For selecting the most acceptable technique. Evaluation Method is yardstick to examine the efficiency and performance of any model Evaluation methods for data mining techniques like classification and association rule mining.

1.2.1 Evaluation method

It is common to call true positives (TP) ‘hits’, true negatives (TN) ‘correct rejections’, false positives (FP) ‘false alarms’, false negatives ‘misses’.

For a perfect model, true positives and true negatives are filled out, the other fields would be zero.

ACCURACY

Accuracy is how efficiently the model categorizes the positives and negatives out of all the documents

Accuracy = (TP + TN)/ (TP+TN+FP+FN)

PRECISION

Precision is the fraction of retrieved documents that are relevant to the query

Precision = TP/(TP+FP)

RECALL

Recall is the fraction of the documents that are relevant to the query that are successfully retrieved

Recall = TP/(TP+FN)

SENSITIVITY

Sensitivity is the proportion of actual positives which are correctly identified as positives by the classifier

Sensitivity = TP / (TP+FN)

SPECIFICITY

Specificity is the proportion of actual negatives which are correctly identified as negatives by the classifier

Specificity = TN / (TN+FP)

F1- SCORE

The F1-Score (also F-Score or F-measure) is a measure of test's accuracy. It considers both the precision and the recall of the test to compute the score.

$$\mathrm{F1} = 2 * \left(\frac{1}{\frac{1}{recall}+\frac{1}{precision}}\right) = 2 * \frac{precision*recall}{precision+recall}$$

1.3 Related Work

As the field of semi-supervised learning is evolving rapidly, it is difficult to summarize the field in one single paper, however we would like to acknowledge few of the papers which acted as our motivation to work further in our project and in a way gave our research a direction.

Semi-supervised Learning with Weakly-Related Unlabeled Data: Towards Better Text Categorization. By Liu Yang, Rong Jin, Rahul Suthankar. Published in the year 2009.

Work description:

This paper, aims at identifying a new data representation, that is on one hand informative to the target class (category), and on the other hand consistent with the feature coherence patterns exhibiting in the weakly related unlabeled data. Since all pre-existing semi-supervised algorithms typically make the cluster assumption, putting the decision boundary in low density areas without crossing the high density region. Hence, the key challenge was to leverage the seemingly unrelated unlabeled data to improve the classification accuracy of the target classes.

The way of approaching towards semi-supervised learning with weakly related unlabeled data, is to first derive a new data representation from unlabeled data, and then apply supervised learning technique to the derived new data representation, the construction of a good data representation combined with the training of a maximum margin classifier under a unified framework. In particular, the data representation generated by this method exploits both labeled and unlabeled data, which differentiates the proposed framework from self-taught learning.

The databases used were the two most standard datasets, the Reuters-21578 dataset and the WebKB dataset. For computational simplicity, 1000 documents are randomly selected from the TREC AP88 dataset and are used as an external information source for both datasets. The main focus is on binary classification. For each class, 4 positive samples and 4 negative samples are randomly selected to form the training set; and the rest of the data serve as the testing set.

SSLW explicitly estimates the optimal word correlation matrix for the target document categorization problem we take into account a word-correlation matrix when computing the kernel similarity matrix, and we search for an optimal word-correlation matrix, towards maximizing the categorization margin. While analyzing this research work we found out that SSLW is more effective than self-taught learning in using unlabeled data to improve classification, due to the fact that Self-taught learning does coding and classification in two separate stages, while SSLW achieves these two purposes simultaneously.

Large Scale Text Classification using Semisupervised Multinomial Naive Bayes. By Jiang Su, Jelber Sayyad-Shirabad, Stan Matwin. (2011)

Work description:

This paper proposes an improved, accurate extension of the Multinomial Naïve Bayes Classification technique called Semi supervised frequency estimate. The objective function of this new estimate is to maximize the marginal log likelihood (MLL), which is quite different from the goal of classification learning, i.e. maximizing conditional log likelihood (CLL). This new model uses the estimates of word probabilities, obtained from unlabeled data, and class conditional probability given a word, learned from labeled data, to learn parameters of an MNB model. Analysis shows that both SFE and EM learn the same word probability estimates from unlabeled data, but SFE obtains better CLL values than EM in labeled training data.

The main concept of semi-supervised frequency estimate is to augment an MNB model by using parameters from unlabeled data while maintaining the CLL score in the well labeled document, since the objective function is no longer MLL. The new frequency estimation method combines the word frequency obtained from the unsupervised learning with the class prediction for that word obtained from the supervised learning.

The datasets used were eight large multi-class datasets to conduct the empirical study for text classification. All these datasets have been widely used in large scale text classification tasks, and are publicly available. Most of the multi-class datasets have imbalance class distribution, and thus we use AUC to compare the prediction performance of learning algorithms. Multi-class AUC is calculated using the average of AUC scores on all pairs of two class AUC. However, accuracy comparisons were also included for a more comprehensive study.

The result devised from this paper show that SFE significantly and consistently improves the AUC and accuracy of MNB, while EM+MNB can fail to improve the AUC of MNB. We also showed that SFE consistently produces better conditional log likelihood (CLL) values than both EM+MNB and MNB trained on the same initial labeled training set, while EM+MNB can result in a model with worse CLL than MNB. Moreover, analysis and empirical results show that SFE has a much lower computational cost than EM+MNB, which makes it a better choice in the presence of very large unlabeled datasets. [2]

Semi Supervised Text Classification with Universum. By Gargi Joshi.(2015)

Work description:

Universum learning, a learning technique conceptually different from SSL or transduction, has become a new research topic in machine learning. When the number of labeled examples is insufficient to estimate the parameters of classification functions, the Universum can be used to approximate the prior distribution of the classification functions. The experimental results can be explained using the concept of Universum introduced by Vapnik, that is, Universum examples implicitly specify a prior distribution on the set of classification functions.

The Universum is a collection of examples that do not belong to any category of interest, but do belong to the same domain as the problem. Weston devised an algorithm called U-supporting vector machine (SVM) to demonstrate that using Universum as a penalty term of the standard SVM objective function can enhance categorization performance. Intuitively, the Universum examples should be close to the text categorization, since they do not belong to any class. The proposed system inspires to design a semi-supervised learning algorithm with Universum to enhance semi-supervised learning categorization performance, particularly under conditions of sufficient unlabeled examples or data sets.

The experiments use four data sets with several combinations, and the experimental results indicate that the proposed algorithm can benefit from Universum examples.

The first approach toward semi supervised text categorization is multi-view learning. The main idea is to represent each document by multiple views and exploit unlabeled documents through the correlation among different views.

The second approach exploring unlabeled documents is to develop semi supervised learning techniques that learn a classification model for text categorization from a mixture of labeled and unlabeled documents

Finally, in addition to SSL and active learning, another approach toward text categorization with small-size samples is to transfer the knowledge of a related text categorization task to the target text categorization task, which is closely related to transfer learning, domain adaptation, or transfer leaning from weakly-related unlabeled documents.

Semi-supervised learning using frequent itemset and ensemble learning for text classification. By Ishtiaq Ahmed, Rahman Ali. Published in year 2014.

Work description:

This research work proposes a novel semi-supervised learning method which makes use of frequent itemset and ensemble learning (FIEL). In this approach, Apriori algorithm has been used for finding the frequent itemset while Multinomial Naive Bayes, Random Forest and LibSVM are used as base learners for ensemble learning which uses majority voting scheme.

A large number of SMS are generated throughout the world whose manual filtration and classification, as spam or ham, is impossible task. Therefore, text classification is used to automatically process and classify SMS data into ham or spam.
The architecture of this system was divided into four components: Preprocessing, Feature Construction, Labeling Unlabeled Dataset and Ensemble Classification.

Three datasets were used for performing this research: UCI repository called as (1) ‘‘SMS Spam Collection Data Set’’ (2) SMS Spam Corpus v.0.1 Small (3) SMS Spam Corpus v.0.1Big

FIEL needs only a limited number of positive dataset, i.e., ham SMS and different amounts of unlabeled SMS. Initially, the frequent pattern mining algorithm (Apriori) is run on ham dataset to find the frequent item based on the positive features under certain minimum support. Then, by running Apriori again under the same min support on the unlabeled SMS, more than one item based on frequent unlabeled features were found. Finally, Multinomial Naive Bayes, Random Forest and LibSVM were used as base classifier to construct an ensemble classifier model. To ensemble the classifiers, major voting scheme to predict the label of the corresponding SMS was used. Extensive experiments have been performed, which show that the proposed technique produces accurate classifier given that only positive class is known under certain minimum support. The conclusion is that FIEL algorithm has produced extremely good results when the positive dataset is low enough, compared to SPY-EM and PEBL approaches, with higher stability in terms of F score. Choosing appropriate minimum support for different corpora would be an interesting research direction for the future as exploiting different min support produces varying results.

Active + Semi-Supervised Learning = Robust Multi-View Learning. By Ion Muslea, Steven Minton, Craig A. Knoblock

Work Description:

This paper introduces a new multi-view algorithm, Co-EMT, that interleaves active and semi-supervised learning. It shows that Co-EMT clearly outperforms the other algorithms both on the parameterized problems and on two real world domains. The experiments suggest that the robustness of Co-EMT comes from active learning compensating for the view correlation.

The database used was 16 of 20 news groups postings from the Mini Newsgroups dataset,7 which is a subset of the well-known 20-Newsgroups domain. Each newsgroup consists of 100 articles that were randomly chosen from the 1000 postings included in the original dataset. We divided the 16 news groups in four groups of four. The examples in each such group are used as either positive or negative examples in one of the two views.

The accuracy of the algorithms is estimated based on four runs of 5-fold cross-validation; consequently, each training and test set consist of 640 and 160 examples, respectively. For the three semi-supervised algorithms, the 640 training examples are split randomly into two groups: 40 of them are used as labeled examples, while the remaining 600 are unlabeled. To keep the comparison fair, Co-EMT and Co-Testing start with 10 randomly chosen labeled examples and query 30 of the 630 unlabeled ones, for a total of 40 labeled examples. For performing the experiment Naive Bayes was used as the underlying algorithm. Co-EMT obtains the lowest error rates in the correlation-incompatibility space.

Semi-Supervised Text Classification Using EM By Kamal Nigam,Andrew McCallum,Tom Mitchell, Published in the year 2005

Work Description:

This paper explores the effectiveness of using a combination of labeled and unlabeled data to train classifiers by estimating parameters of a generative model through iterative Expectation-Maximization (EM) techniques, when applied to the domain of text classification. This model is an extremely simplistic representation of the complexities of written text.

This chapter explains and illustrates three key points about semi-supervised learning for text classification with generative models.

First, despite the simplistic representation, some text domains have a high positive correlation
between generative model probability and classification accuracy. In these domains, a straightforward application of EM with the naive Bayes text model works well.

Second, some text domains do not have this correlation. Here we can adopt a more expressive
and appropriate generative model that does have a positive correlation. In these domains, semi-supervised learning again improves classification accuracy.

Finally, EM suffers from the problem of local maxima, especially in high dimension domains such as text classification. We demonstrate that deterministic annealing, a variant of EM, can help overcome the problem of local maxima and increase classification accuracy further when the generative model is appropriate.

The EM technique as applied to the case of labeled and unlabeled data with naive Bayes yields a straightforward and appealing algorithm. First, a naive Bayes classifier is built in the standard supervised fashion from the limited amount of labeled training data. Then, classification of the unlabeled data with the naive Bayes model is performed, noting not the most likely class but the probabilities associated with each class. Then, a new naive Bayes classifier using all the data (labeled and unlabeled)using the estimated class probabilities as true class labels was rebuild. This means that the unlabeled documents are treated as several fractional documents according to these estimated class probabilities. We iterate this process of classifying the unlabeled data and rebuilding the naive Bayes model until it converges to a stable classifier and set of labels for the data.

$$\begin{aligned} l(\theta|X,Y) \quad = \quad & \log(\mathrm{P}(\theta)) + \sum_{x_i \in X_u} \log \sum_{j \in [M]} \mathrm{P}(c_j|\theta)\mathrm{P}(x_i|c_j;\theta) \\ & + \sum_{x_i \in X_l} \log\left(\mathrm{P}(y_i = c_j|\theta)\mathrm{P}(x_i|y_i = c_j;\theta)\right). \end{aligned}$$

1.4 Gap-Analysis

Though many semi-supervised learning methods have been proposed in recent years, there is no dominating method in this area. Which points out that the reason for this is that semi-supervised learning methods need to make stronger model assumptions than supervised learning methods. [4]

All successful semi-supervised algorithms typically make the cluster assumption, which puts the decision boundary in low density areas without crossing the high density regions. It is observed that the cluster assumption is only meaningful when the labeled and unlabeled data are somehow closely related, hence in such case, the cluster assumption may not be valid; and consequently how to leverage this type of unlabeled data to enhance the classification accuracy becomes a challenge. [8]

Another most commonly used technique the Multinomial Naïve Bayes, the EM (Expectation Maximization) may decrease the performance of MNB when the dataset contains multiple subtopics in one class. Thus, there is still a need for a semi-supervised learning method that is fast, simple to use, and can consistently improve the prediction performance of MNB.

The traditional semi-supervised methods do not require large amount of labeled data but may not produce good results if they are provided with only positive and unlabeled data.

SSL rely on the assumptions that the views (disjoint subsets) are compatible and uncorrelated. As these assumptions are unlikely to hold in practice, it is crucial to understand the behavior of multi-view algorithms on problems with incompatible, correlated views the existing semi-supervised algorithms are not robust over the whole spectrum of parameterized problems.

In-depth research has shown that the existing algorithms, such as Multinomial Naïve Bayes, SVM, PEBL, sometimes produce unstable results due to their dependence on certain factors. There exists a huge scope of improvement.

Also the performance of semi-supervised learning methods may be data dependent. The high dimensional data poses computational constraints, while the sparse data means that a document may have to be classified based on the values of a small number of features. Thus, finding an algorithm which is both efficient and can generalize well is a challenge for this application domain. [23]

Chapter 2

Design and Implementation

2.1 System Overview

In our project, we worked on semi-supervised learning techniques based on the EM model. The experiments consisted of training various classifiers with some labelled data and then predicting the expected labels or classes for the unlabelled data. Finally, the classifier was re-trained using both the labelled and the unlabelled data with the predicted labels, and the classes were predicted on the test data. Different datasets were used to experiment and to show how the dataset and the type of data were on of the biggest factors in the performance of a classifier. To analyse the results, we will look at each dataset and draw a comparison between the classifiers used.

2.1.1 Part 1: Techniques Used

2.1.1.1 Classifier Design and Implementation

In real world applications, handling just the labelled data isn't sufficient to obtain accurate parameter estimates. For semi-supervised learning, we discuss how classifiers based on EM utilise both the labelled and unlabelled data to improve parameter estimates. For all the datasets used, we have applied the EM technique to the labelled and unlabelled data. This technique along with the range of classifiers that we have experimented with, yields a straightforward and appealing algorithm.

First, the classifier is built in the normal supervised fashion from the labelled training data that we have. After training the classifier with the labelled data, we predict the expected classes or labels for the unlabelled data. The predicted labels are written in a csv file and saved. We manually combine the labelled data and the unlabelled data with the predicted labels into a single csv file for further training. Then, we rebuild a new classifier and train it with all the labelled training data that we have now i.e. labelled data and the unlabelled data with the predicted expected labels.

Multinomial Naive Bayes

Multinomial Naïve Bayes model specifies that a document is represented by the set of word occurrences from the document. The order of the words is lost, however, the number of occurrences of each word in the document is captured. When calculating the probability of a document, one multiplies the probability of the words that occur. The individual word occurrences to be the "events" and the document to be the collection of word events.

Multinomial Naïve Bayes assumes that all attributes of the dataset are independent of each other given the context of the class. This is the so-called "naive Bayes assumption". Because of the independence assumption, the parameters for each attribute can be learned separately, and this greatly simplifies learning, especially in this case, with the IMDB Large Movie Dataset, where the number of attributes is large.

While some simple document classification tasks can be accurately performed with vocabulary sizes less than one hundred, many complex tasks on real-world data from the Web, UseNet and newswire articles do best with vocabulary sizes in the thousands. Naive Bayes has been successfully applied to document classification in many research efforts. [2]

In the multinomial model, a document is an ordered sequence of word events, drawn from the same vocabulary V. We assume that the lengths of documents are independent of class.2 We again make a similar naive Bayes assumption: that the probability of each word event in a document is independent of the word's context and position in the document. Thus, each document di is drawn from a multinomial distribution of words with as many independent trials as the length of di. This yields the familiar "bag of words" representation for documents. Define Nit to be the count of the number of times word wt occurs in document di. Then, the probability of a document given its class from. The following equation is simply the multinomial distribution.

$$\mathbf{P(d_i|c_j;\theta) = (P(\ |d_i|)|d_i|!\pi_{t=1}{}^{|v|}\ P(w_t|cj;\theta)^{Nit}\ )/_{Nit!}}$$

The parameters of the generative component for each class are the probabilities for each word, written $\theta w_t|cj = P(\ w_t|c_j;\theta)$, where $0 \leq \theta w_t|c_j \leq 1$ and $\sum_t \theta w_t|c_j = 1$.

Bernoulli Naïve Bayes

Bernoulli Naïve Bayes model specifies that a document is represented by a vector of binary attributes indicating which words occur and do not occur in the document. When calculating the probability of a document, one multiplies the probability of all the attribute values, including the probability of non-occurrence for words that do not occur in the document. The document becomes the "event," and the absence or presence of words become attributes of the event. This describes a distribution based on a multi-variate Bernoulli event model. This approach is more traditional in the field of Bayesian networks, and is appropriate for tasks that have a fixed number of attributes. Bernoulli Naïve Bayes assumes that the probability of each word occurring in a document is independent of the occurrence of other words in a document. Thus, the probability of a document given its class is simply the product of the probability of the attribute values over all word attributes.

This model classifies new test documents using Bayes' rule to turn the generative model around and calculate the posterior probability that a class would have generated the test document in question. Classification then becomes a simple matter of selecting the most probable class. equivalent to viewing the distribution over documents as being described by a Bayesian network, where the absence or presence of each word is dependent only on the class of the document. This model does not capture the number of times each word occurs, and that it explicitly includes the non-occurrence probability of words that do not appear in the document.

In the multi-variate Bernoulli event model, a document is a binary vector over the space of words. Given a vocabulary V, each dimension of the space t, $t \in \{1,...,|V|\}$, corresponds to word w_t from the vocabulary. Dimension t of the vector for document d_i is written B_{it}, and is either 0 or 1, indicating whether word w_t occurs at least once in the document. With such a document representation, we make the naive Bayes assumption as stated above.[16]

$$\mathbf{P(d_i|c_j;\theta) = \pi_{t=1}^{|v|} (B_{it}P(w_t|c_j;\theta) + (2)\ (1-B_{it})(1-P(w_t|c_j;\theta)))}$$

Gaussian Naïve Bayes

Gaussian Naive Bayes is a generalization of Naive Bayes Networks, which are a special case of probabilistic networks that allows treating continuous variables. It solves the main problems in assessment procedures: low complexity and high accuracy. The Assessment Tool based on Gaussian Naive Bayes presents significant better results when compared with an Assessment Tool based on Naive Bayes for the same case. Besides, the Assessment Tool based on Naive Bayes presented better computational performance in terms of CPU time. [7]

The present assessment method can also be used for complex scenarios, such as driving, flight training and several skills training based on virtual reality. Similarly, longer virtual reality training sessions could be simulated since the hardware used could provide enough performance.

Use of Gaussian distribution for X (training data) and to compute its parameters from D (sample data). Using some mathematical simplification, it is possible to reduce computational complexity of the equation:

$$\log [P(w_i \mid X_1, X_2, \ldots, X_n)] = \log [(1/S)\, P(w_i)\, \Pi_n^{k=1} P(X_k \setminus w_i)] = \log (1/S) + \log P(w_i) + \sum_n^{k=1} \log[P(X_k \setminus w_i)]$$

As S is a scale factor, it is not necessary be computed in classification rule for GNB.

Based on the same space of decision with classes, a GNB method computes conditional class probabilities and then predicts the most probable class of a vector of training data X, according to sample data D. The parameters of GNB method are learning from data and the conditional probabilities are estimated and the final decision about vector of training data X is done.

Linear SVM

Linear SVM is the newest *extremely fast* machine learning (data mining) algorithm for solving *multiclass* classification problems from *ultra large* data sets that implements an original proprietary version of a *cutting plane algorithm* for designing a *linear support vector machine.* Linear SVM is a *linearly scalable routine* meaning that it creates an SVM model in a CPU time which scales *linearly* with the size of the training data set. The comparisons with other known SVM models clearly show its superior performance when high accuracy is required.

Linear SVM is efficient in dealing with extra-large data sets(say, several millions training data pairs). It also is a solution for multiclass classification problems with any number of classes, capable of working with high dimensional data (thousands of features, attributes) in both sparse and dense format. Do not require expensive computing resources (personal computer is a standard platform). And therefore is ideal for contemporary applications in digital advertisement, e-commerce, web page categorization, text classification, bioinformatics, proteomics, banking services and many other areas. [18]

Random Forest

Random Forest classifier is one of the best available classifiers – able to classify large amounts of data with accuracy. Random Forests are an ensemble learning method (also thought of as a form of nearest neighbour predictor) for classification and regression that construct a number of decision trees at training time and output the class that is the mode of the classes output by individual trees. The basic principle of Random Forest is that a group of "weak learners" can come together and result in a possible "strong learner". Random Forests are good at predictions considering they do not overfit because of the law of large numbers.

Random Forest grows many classification trees. If the number of cases in the training set is N, sample N cases at random. The sample will be the training set for growing the tree. [11]

If there are M input variables, a number mM is specified such that at each node, m variables are selected at random out of the M and the best split on these m variables is used to split the node. The value of m is held constant during the forest growing. As m goes down, both inter-tree correlation and the strength of individual trees go down. So an optimal value of m had to be discovered. The greater the inter-tree correlation, the greater the random forest error rate, so one pressure on the Random Forest model is to have the trees as uncorrelated as possible. [23]

Every tree that is built, is grown to the maximum extent possible and no pruning is done. Random Forest has a range of benefits. It handles thousands of input variables without any variable deletion. It even performs well in the cases of missing data by providing effective methods for estimating missing data and by maintaining accuracy. It computes proximities between pairs of cases that can be used in clustering, locating outliers, or (by scaling) give interesting views of the data. Random Forest helped in semi-supervised learning as all these abilities can be extended to unlabelled data as well.

Logistic Regression

Logistic Regression comes from the field of statistics and finds great application in machine learning as well. It is one of the best methods for binary classification problems thus making it one of the most used classifiers in our project. Logistic regression uses an equation as the representation, very much like linear regression. [44]

Input values (x) are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) to predict an output value (y). A key difference from linear regression is that the output value being modelled is binary values (0 or 1) rather than a numeric value.

Below is an example logistic regression equation:

$$\mathbf{y = e^{(b0 + b1*x)} / (1 + e^{(b0 + b1*x)})}$$

$$\mathbf{\ln(p(X) / 1 - p(X)) = b0 + b1 * X}$$

Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data. Logistic Regression models the probability of the default class (e.g. the first class).

For example, in the IMDB Large Movie Review Dataset, we were classifying text as positive(1) or negative(0) from the movie review, then the first class could be positive and the logistic regression model could be written as the probability of positive given the movie review, or more formally:

P(Class = (Positive | Movie Review)

Written another way, we are modelling the probability that an input (X) belongs to the default class (Y=1), we can write this formally as:

$$\mathbf{P(X) = P(Y=1|X)}$$

Logistic regression is a linear method, but the logistic function transforms the predictions. The impact of this is that we can no longer understand the predictions as a linear combination of the inputs as we can with linear regression. In practice we can use the probabilities directly. Because this is classification and we want a crisp answer, we can snap the probabilities to a binary class value, for example:

$$\mathbf{0 \text{ if } P(Negative) < 0.5}$$

$$\mathbf{1 \text{ if } P(Positive) >= 0.5}$$

Multilayer Perceptron Classifier

The field of artificial neural networks is often just called neural networks or multi-layer perceptrons after perhaps the most useful type of neural network. A perceptron is a single neuron model that was a precursor to larger neural networks.

It is a field that investigates how simple models of biological brains can be used to solve difficult computational tasks like the predictive modelling tasks we see in machine learning. The goal is not to create realistic models of the brain, but instead to develop robust algorithms and data structures that we can use to model difficult problems.

The power of neural networks come from their ability to learn the representation in the training data and how to best relate it to the output variable that has to be predicted. In this sense neural networks learn a mapping. Mathematically, they are capable of learning any mapping function and have been proven to be a universal approximation algorithm.

The predictive capability of neural networks comes from the hierarchical or multi-layered structure of the networks. The data structure can pick out (learn to represent) features at different scales or resolutions and combine them into higher-order features. For example, from lines, to collections of lines to shapes.

Once a neural network has been trained it can be used to make predictions. Predictions on test or validation data in order to estimate the skill of the model on unseen data can be made. and also can be deployed operationally to make predictions continuously.

The network topology and the final set of weights is all that you need to save from the model. Predictions are made by providing the input to the network and performing a forward-pass allowing it to generate an output that you can use as a prediction.

$$y = \varphi(\sum_{i=1}^{n} w_i x_i + b) = \varphi(\mathbf{w}^T \mathbf{x} + b)$$

where w denotes the vector of weights, **x** is the vector of inputs, **b** is the bias and $\boldsymbol{\varphi}$ is the activation function.

2.1.2 Part 2: Data Preprocessing

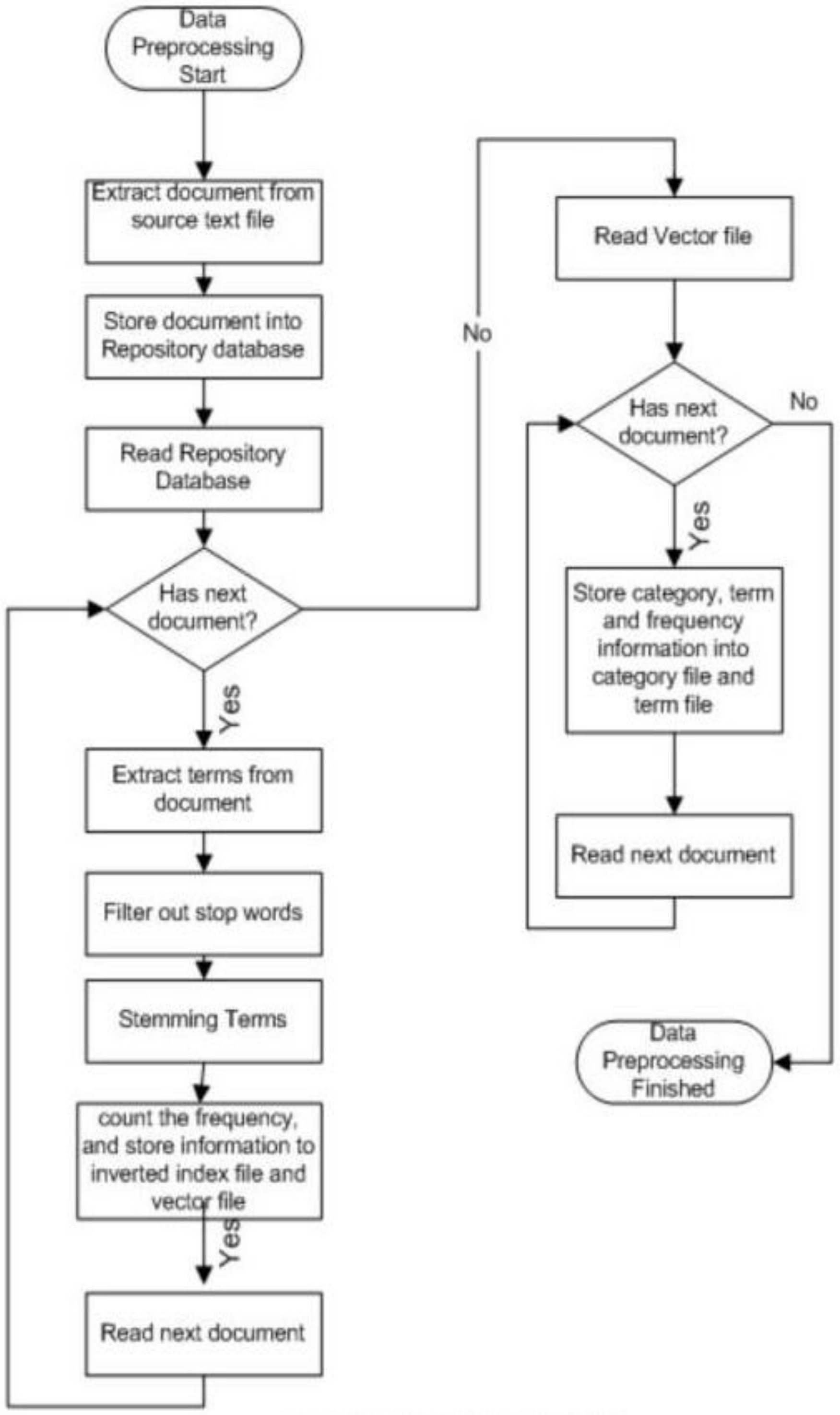

Data Preprocessing Start
Extract document from source text file
Store document into Repository database
Read Repository Database
Has next document?
Yes
No
Extract terms from document
Filter out stop words
Stemming Terms
count the frequency, and store information to inverted index file and vector file
Yes
Read next document
Read Vector file
Has next document?
No
Yes
Store category, term and frequency information into category file and term file
Read next document
Data Preprocessing Finished

Fig 2.1.1 Flow Chart of Data Preprocessing [59]

2.1.2.1 Data Preparation

Data preparation remains the most significant step of any data science process. During the course of our research project, this step took a considerable amount of time as we understood the importance of this part. We have chosen 3 datasets to compare the results and consistency of the algorithms among the datasets.

IMDB Large Movie Review Dataset

Dataset obtained from Kaggle.com containing 25,000 highly polar movie reviews for training and 25,000 for testing, with an additional 50,000 unlabeled reviews. The sentiment of reviews is binary, meaning a review with an IMDB rating of <5 has sentiment score of 0, while the one with a sentiment score >=7 has a sentiment score of 1. No movie has more than 30 reviews and no movie in the training data is repeated in the testing data. There were 3 Tab Separated Values(.tsv) files which and organized into two columns i.e. Review and Sentiment.

Amazon Reviews: Unlocked Mobile Phones

Obtained from Kaggle datasets, it contains around 400,000 reviews of unlocked mobile phones having information like reviews, ratings, price and their relationships. The fields in the dataset are Product Title, Brand, Price, Rating, Review Text out of which only the Product Name, Review and Rating fields were used and the remaining removed during cleaning. The ratings range from 1-5, with a 3 meaning neutral, later removed to improve polarity. Other steps to improve polarity were: reviews with rating of 1,2 changed to 0, whereas reviews with a rating of 4,5 changed to 1. To avoid correlation, we chose only 20 reviews for each product. We picked 4000 unlabeled reviews, 2000 reviews and their ratings to create our training data and 1000 test data. Finally, the dataset was converted into Comma Separated Values(.csv) files to simplify the reading process in python.

SMS Spam Collection Dataset

Obtained from UCI's Machine Learning Depository, this dataset, a collection of both ham and spam messages containing 5574 instances which was further divided into 500 labelled data and 4500 unlabeled data, the remaining used as testing data. Even though being comparatively smaller dataset, the variations in size might help us to better understand its impact on quality and accuracy of insights. The datasets were later converted to Comma Separated Values(.csv) files after removing commas by manual preprocessing.

2.1.2.2 Reading the Datasets

We used the pandas package, which provides fast, flexible and expressive data structures designed to make working with structures (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive.

To read a file, first we need to import the pandas package and then use the read_csv method to read .csv or .tsv files.

2.1.2.3 Preprocessing Operations in Python

Preprocessing the data is the process of cleaning and preparing the text for classification. Raw data may sometimes contain a lot of noise and uninformative parts such as HTML tags, scripts, etc. On words level, many words, such as stop words, in the text don't have an impact on the general orientation of it. Keeping those words makes the dimensionality of the problem high and hence the classification more difficult since each word in the text is treated as one dimension. Reducing the noise in the text should help improve the performance of the classifier and speed up the classification process, thus aiding in real time sentiment analysis. We carried out multiple steps in this process.

Removal of HTML Tags

This was done using the BeautifulSoup library of Python. Using the get_text method in BeautifulSoup library, we can extract the text without any tags or markup. Though, for a simple application on a dataset such as the IMDB Large Movie Review Dataset, regular expression could have be used to remove tags and markup, but it is not considered a reliable practice and so we use BeautifulSoup, which makes for a very powerful library.

Removal of Symbols

When considering how to clean text, it is very important to think about the data problem we are trying to solve. In this case, we have removed both numbers and punctuation for the ease of use. We will use the re package, which is used to deal with regular expressions. This package comes pre-installed in python. Therefore, we have to just import it and use it. We do a basic find and replace using regular expression, which searches for any patterns or letters except alphabets and replaces them with a blank space.

Conversion to Lower case and Tokenization

We convert the text from the last step into lowercase by using the lower() method.

Given a defined document unit or some text, tokenization is the task of chopping it up into pieces, called tokens. We use the split() method to split the sequence of words into tokens.

Stop Word Removal

Finally, we need to deal with some extremely common words that would appear to be of little value in helping select documents matching a user need are excluded from the vocabulary entirely. In English, they include words such as “a”, “and”, “the”, etc. These are called stop words and python comes with a built in stop words list.

First, we had to install the library plus the packages that come with it. Then we import the words list from the python Natural Language Toolkit(NLTK). We use nltk to get a list of stop words. To remove the stop words from the text, we go through each tokenised word and check if it exists in the list of stop words. If it does, it is removed from the new list of words.

We needed to improve the speed of this part of the code as it is called tens of thousands of times. Therefore, we converted the stop word list into a different data type, a set. In python, searching sets is much faster than searching lists. Because we will have to run these codes on multiple reviews or texts, we combined all these pre-processing steps under a single callable function with only the review as the parameter to it.

2.1.3 Feature Selection

2.1.3.1 Bag of Words

The raw data, which is a sequence of symbols can’t be fed to the algorithms as most of them expect numerical feature vectors with a fixed size rather than the raw text documents with variable length. In order to address this, we used the utilities provided by scikit-learn for the most common ways to extract numerical features from text content. We used CountVectorizer to create our Bag of Words. Vectorization is the general process of turning a collection of text documents into numerical feature vectors. This specific strategy is called Bag of Words or "Bag of n-grams" representation. In the IMDB dataset, we have a very large number of reviews, which will give us a large vocabulary. To limit the size of the feature vectors, we should choose some maximum vocabulary size. Keeping in mind that the stop words have already been removed, we are taking the 4000 most frequent words.

First, we initialize the CountVectorizer object and set its parameters. It comes with its own options to automatically do pre-processing, tokenization, and stop-word removal, but here we have specified it has None, and have built our own pre-processing function for both learning and experimental purposes.

2.1.3.2 Bag of Centroids using K-means

In this method, we have employed Google's trained word2vec model, which works in a way that is similar to deep approaches such as recurrent neural nets or deep neural nets, but it implements certain algorithms, such as hierarchical softmax, that make it computationally more efficient. In our code, word2vec creates clusters of semantically similar words. This process of grouping vectors is called "vector quantization". To accomplish this, we are using K-means as a clustering algorithm. We first set K or the number of clusters. Running our code several times on different values of K, ranging from 5-500 words per cluster, we noticed that small clusters, with an average of only 5 words or so per cluster, gave better results than a large cluster with a higher number of words. Another thing that was worth noticing was the variation in cluster creation time with changes in the value of K. Clustering with large K was very time consuming and so our code took from 550 to 650 seconds to run the K-means.

First, we initialize a k-means object and use it to extract centroids. Then we use fit_predict to fit the K-means on the word vectors and then consequently predict the various word vectors into different clusters. The clusters built may differ in every compilation as Word2Vec relies on random number seed. The clusters are then printed and while some of them make a lot of sense, some clusters lack quality. At any rate, now we have a cluster assigned to each word vector and now we can define a function to convert reviews into Bag of Centroids. It is almost similar to the Bag of Words but uses semantically related clusters instead of individual words. To build the Bag of

Centroids, we define a function in which, the number of clusters is equal to the highest cluster index in the word/centroid map. The function gives us a numpy array for each review, each with a number of features equal to the number of clusters. In this way, we create bags of centroids for our training and test set as well.

2.2 Program Implementation

2.2.1 Data Preprocessing

2.2.1.1 Reading the Database

```
#Import the pandas package
import pandas
#Call read_csv to read a file, pass the file name, the delimiter and header parameters
variable = pandas.read_csv("Name of the file", header=0, delimiter="(, for .csv or \t for
.tsv)", quoting=3 )
#print the shape of the read file to check if all the rows have been read
print variable.shape
```

Table 2.2.1 Reading the Database

2.2.1.2 Data Preparation:

```
#install BeautifulSoup4 using pip

sudo pip install BeautifulSoup4

#import BeautifulSoup

#initiate BeautifulSoup

variable = BeautifulSoup(<Review>)

#remove the HTML tags by using get_text

variable.get_text
```

Table 2.2.2 Removing HTML Tags

```
#import the regular expression package, re

import re

# Use regular expressions to do a find-and-replace

variable = re.sub("[^a-zA-Z]",          # The expression to search for
                  " ",                  # The expression to replace it with
                  <Review>)             # The text to search
```

Table 2.2.3 Removing numbers and punctuations

```
#split() method to split the sequence of words into tokens after converting the sequence into
lower case

Tokens = variable.lower().split()
```

Table 2.2.4 Tokenization

```
#Import nltk

# Import the stop word list
from nltk.corpus import stopwords
#Check if the word exists in the English list of stop words. If it does, remove it
words = [w for w in words if not w in stopwords.words("english")]
```

Table 2.2.5 Stop Word removal

2.2.3 Feature Selection

Creating features from Bag of Words (BoW) using CountVectorizer: The fit_transform() does two things: First, it fits the model and learns the vocabulary; second, it transforms our training data and into feature vectors. The input to fit_transform() should be a list of strings. Finally, to make our job easier, we convert the fitted data to a numpy array.

```
vectorizer = CountVectorizer(analyzer = "word",   \
                             tokenizer = None,    \
                             preprocessor = None, \
                             stop_words = None,   \
                             max_features = 4000)
```

Table 2.2.6 Bag of Words using CountVectorizer

Bag of Centroids using K-means

```
def bag_of_centroids(wordlist, word_centroid_map):  #Function to create bag of centroids
# Set number of clusters is equal to the highest cluster index in the word/centroid map.
# Initialize the bag of centroids vector
        bag_of_centroids = np.zeros(num_centroids,<dtype>)
# Run K-means
        K =  Kmeans(<no. of clusters>)
        K_variable = k.fit_predict(word_vectors)
```

Table 2.2.7 Bag of Centroids

2.2.4 Generate Classifier

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier    #import Library

rf_estimator = RandomForestClassifier()                #initiate the RandomForestClassifier

rf_estimator.fit(<review>,<label>)                     # Training RFC using BOW

result = rf_estimator.predict(<test data>)             # .predict() to make label predictions
```

Table 2.2.8 Random Forest

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. estimators.predict() performs the collection of fitted sub-estimators.

Gaussian Naïve Bayes

```
from sklearn.ensemble import GaussianNB                #import Library

gnb_estimator = GaussianNB()                           #initiate the GaussianNB classifier

gnb_estimator.fit(<review>,<label>)                    #training GNB using BOW

result = gnb_estimator.predict(<test data>)            # .predict() to make label predictions
```

Table 2.2.9 Gaussian Naïve Bayes

Logistic Regression

```
from sklearn.linear_model import LogisticRegression as LR   #import Library

lr = LR()                                                  # initiate the LR classifier

lr.fit(<review>,<label>)                                   # Training LR using BOW

 result = svc_estimator.predict(<test data>)               # .predict() to make label predictions
```

Table 2.2.10 Logistic Regression

Chapter 3

Experimental Result

3.1 Experimental Data

While searching and gathering data sets various criteria were considered such as size of datasets, whether they were open source or proprietary and the readability and understandability.

The following data sets were used:

IMDB Large Movie Review Dataset

Dataset obtained from Kaggle.com containing 25,000 highly polar movie reviews for training and 25,000 for testing, with an additional 50,000 unlabeled reviews. The sentiment of reviews is binary, meaning a review with an IMDB rating of less than 5 has sentiment score of 0, while the one with a sentiment score greater than equal to7 has a sentiment score of 1. No movie has more than 30 reviews and no movie in the training data is repeated in the testing data. There were 3 Tab Separated Values (.tsv) files which and organized into two columns i.e. Review and Sentiment.

Amazon Reviews: Unlocked Mobile Phones

Obtained from Kaggle datasets, it contains around 400,000 reviews of unlocked mobile phones having information like reviews, ratings, price and their relationships. The fields in the dataset are Product Title, Brand, Price, Rating, Review Text out of which only the Product Name, Review and Rating fields were used and the remaining removed during cleaning. The ratings range from 1-5, with a 3 meaning neutral, later removed to improve polarity. Other steps to improve polarity were: reviews with rating of 1,2 changed to 0, whereas reviews with a rating of 4,5 changed to 1. To avoid correlation, we chose only 20 reviews for each product. We picked 4000 unlabeled reviews, 2000 reviews and their ratings to create our training data and 1000 test data. Finally, the dataset was converted into Comma Separated Values (.csv) files to simplify the reading process in python.

SMS Spam Collection Dataset

Obtained from UCI's Machine Learning Depository, this dataset, a collection of both ham and spam messages containing 5574 instances which was further divided into 500 labelled data and 4500 unlabeled data, the remaining used as testing data. Even though being comparatively smaller dataset, the variations in size might help us to better understand its impact on quality and accuracy of insights. The datasets were later converted to Comma Separated Values(.csv) files after removing commas by manual preprocessing.

3.2 Experimental Results and Analysis

3.2.1 Part 1: Performance measure of each Classifier

This section discusses the performance of the classifiers as measured by the AUC against the different datasets + feature selection technique used for text classification. In figure 3.2.1, we can clearly notice how MultinomialNB has performed better with BOW than with BOC.

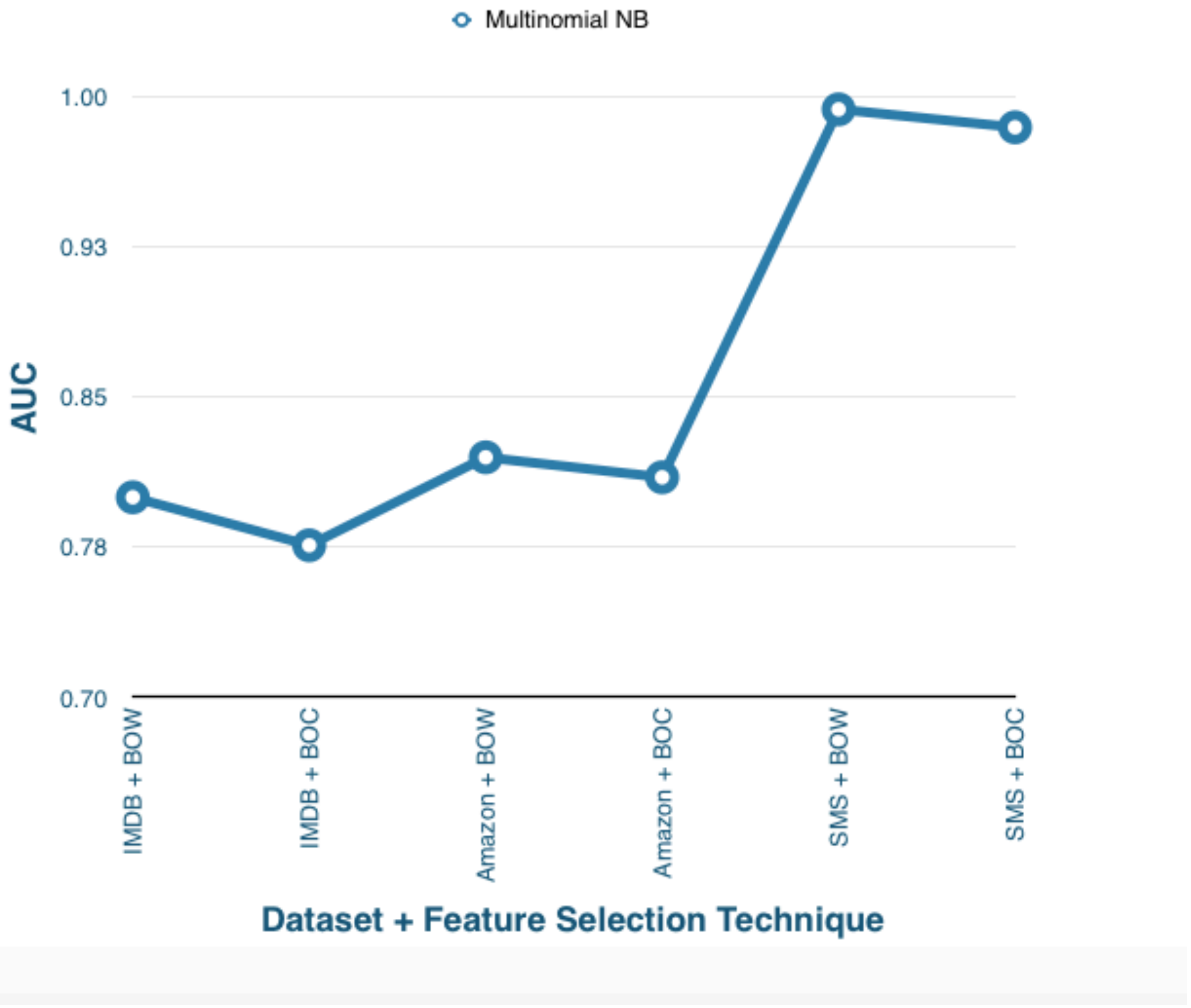


Figure 3.2.1 Performance of MultinomialNB

While other classifiers produced better results with the smaller SMS Spam Collection dataset, Random Forest performed best with the IMDB Large Movie Review Dataset. With larger number of features, Random Forest's performance improves. As the number of features decrease in the Amazon dataset, its AUC scores drop. Using the SMS dataset, there is a further decrease in the number of features and with that we notice a decrease in the AUC scores as well.

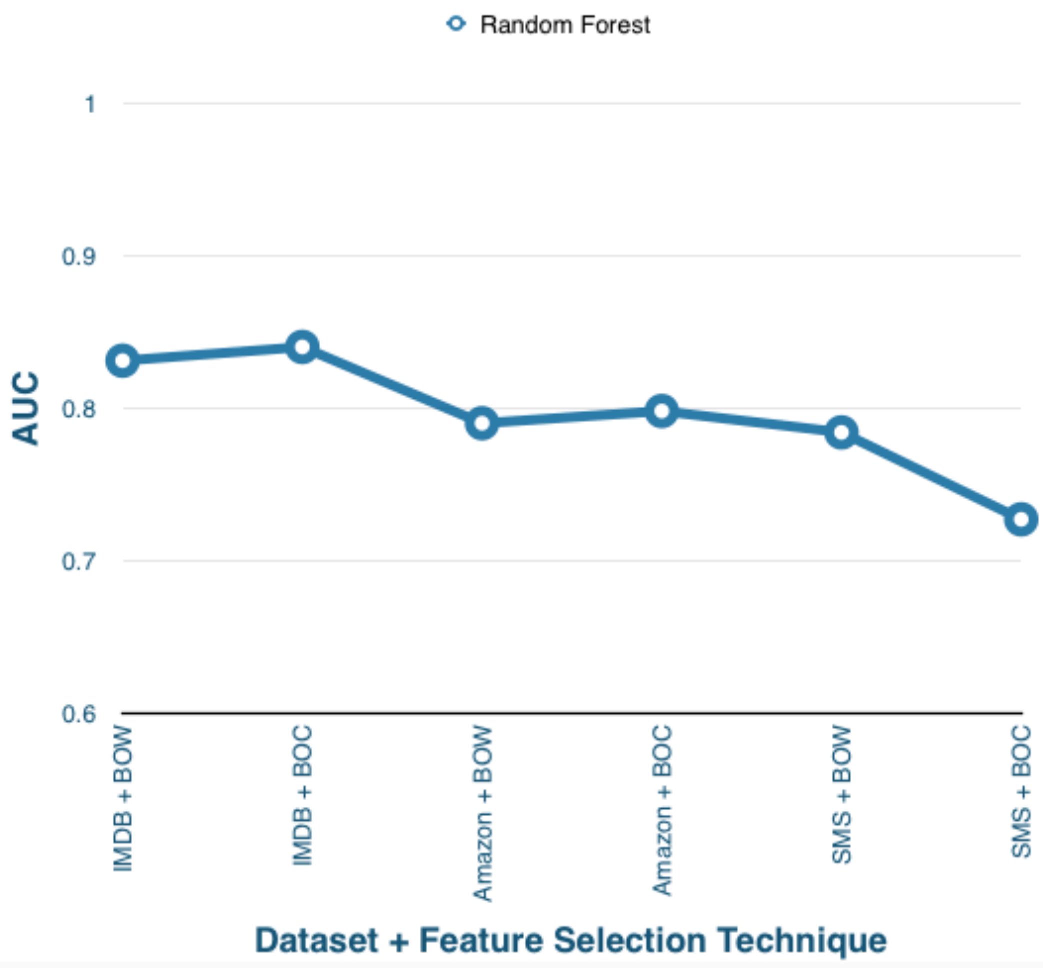


Figure 3.2.2 Performance of Random Forest

The simple neural network designed with one hidden layer of 15 neurons, with the help of MLPClassifier produced significantly better results with the SMS Spam Collection dataset. An AUC of around 84%, while using the IMDB dataset, is quite promising considering the scope of improvement in the design complexity of the neural network and the relatively lesser work done with neural networks in the domain of text classification.

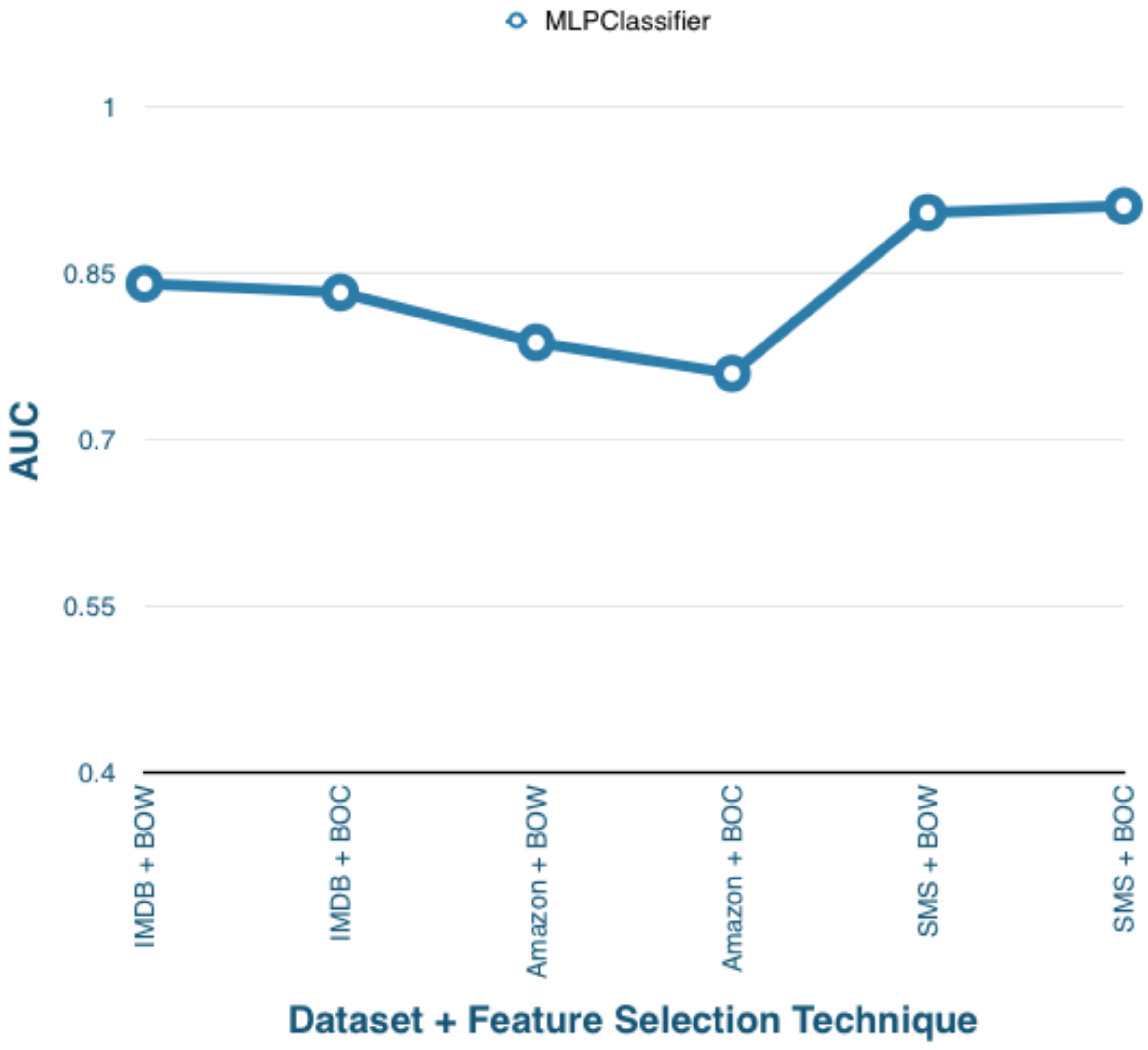


Figure 3.2.3 Performance of MLPClassifier

LinearSVC reached an AUC of around 85% on the IMDB dataset and did much better with the SMS dataset by achieving an AUC of 90.5% and a precision of 97%. It can be seen that LinearSVC performs almost the same with both BOW and BOC models and has only slight variations.

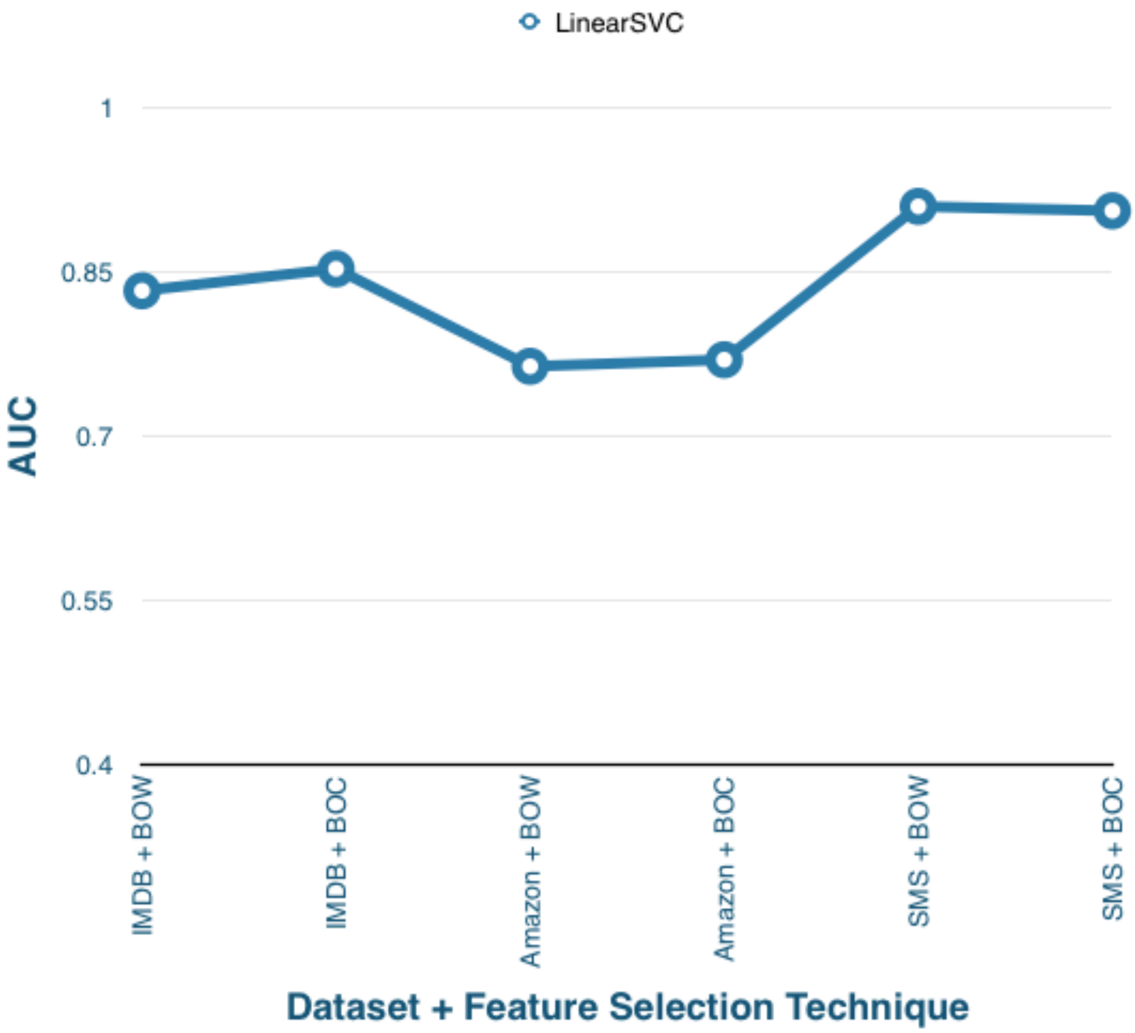


Figure 3.2.4 Performance of LinearSVC

LogisiticRegression's performance remained fairly consistent for the IMDB and SMS dataset, but its performance dipped a little for the Amazon dataset. LogisticRegression with both BOW and BOC produced almost similar results in all the three datasets. It performs consistently over IMDB and SMS datasets with an AUC of 86%.

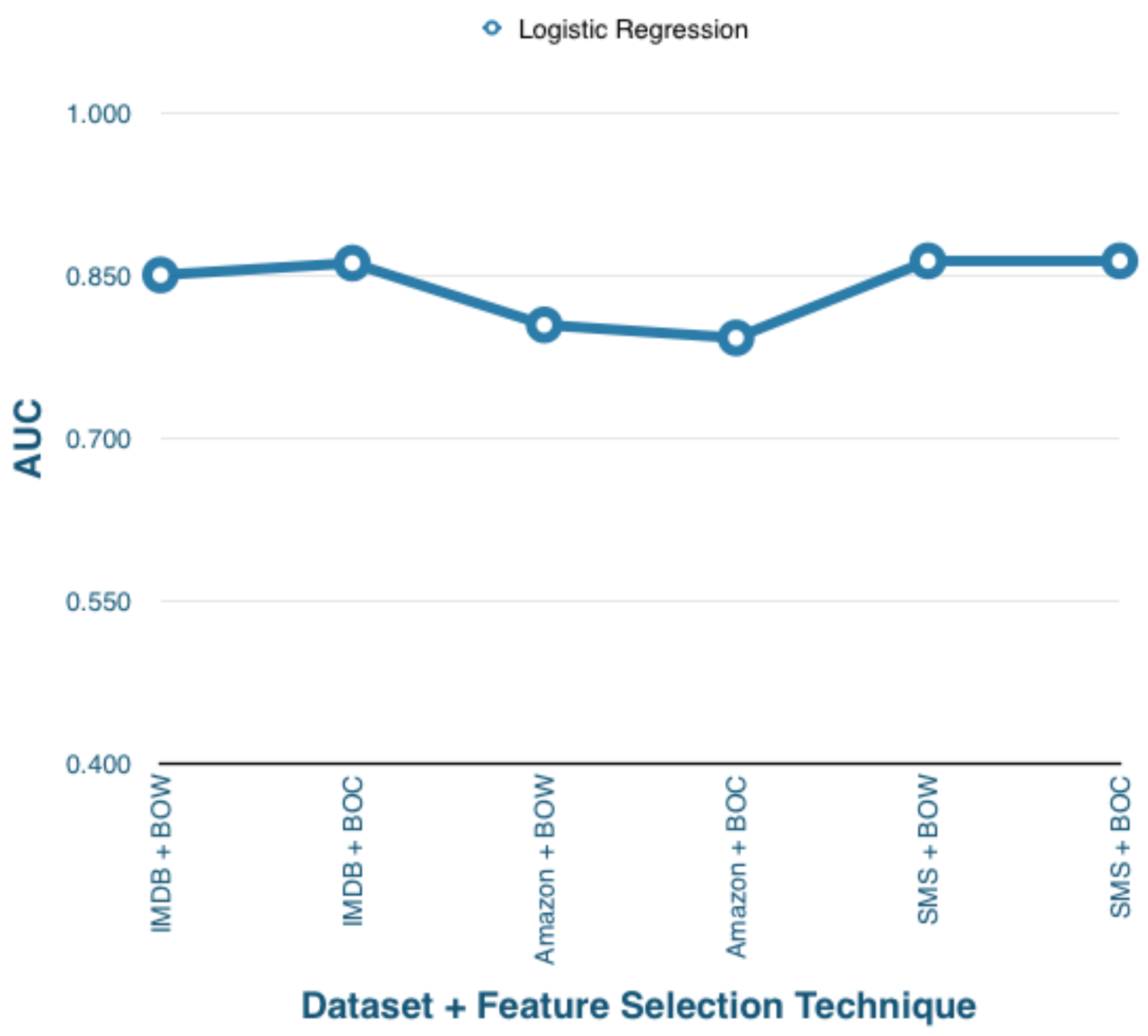


Figure 3.2.5 Performance of Logistic Regression

The highlight of BernoulliNB's performance was the poor AUC score of 0.647 with the BOC model on the Amazon dataset. It performed decently well otherwise, scoring an 86% AUC for the SMS dataset.

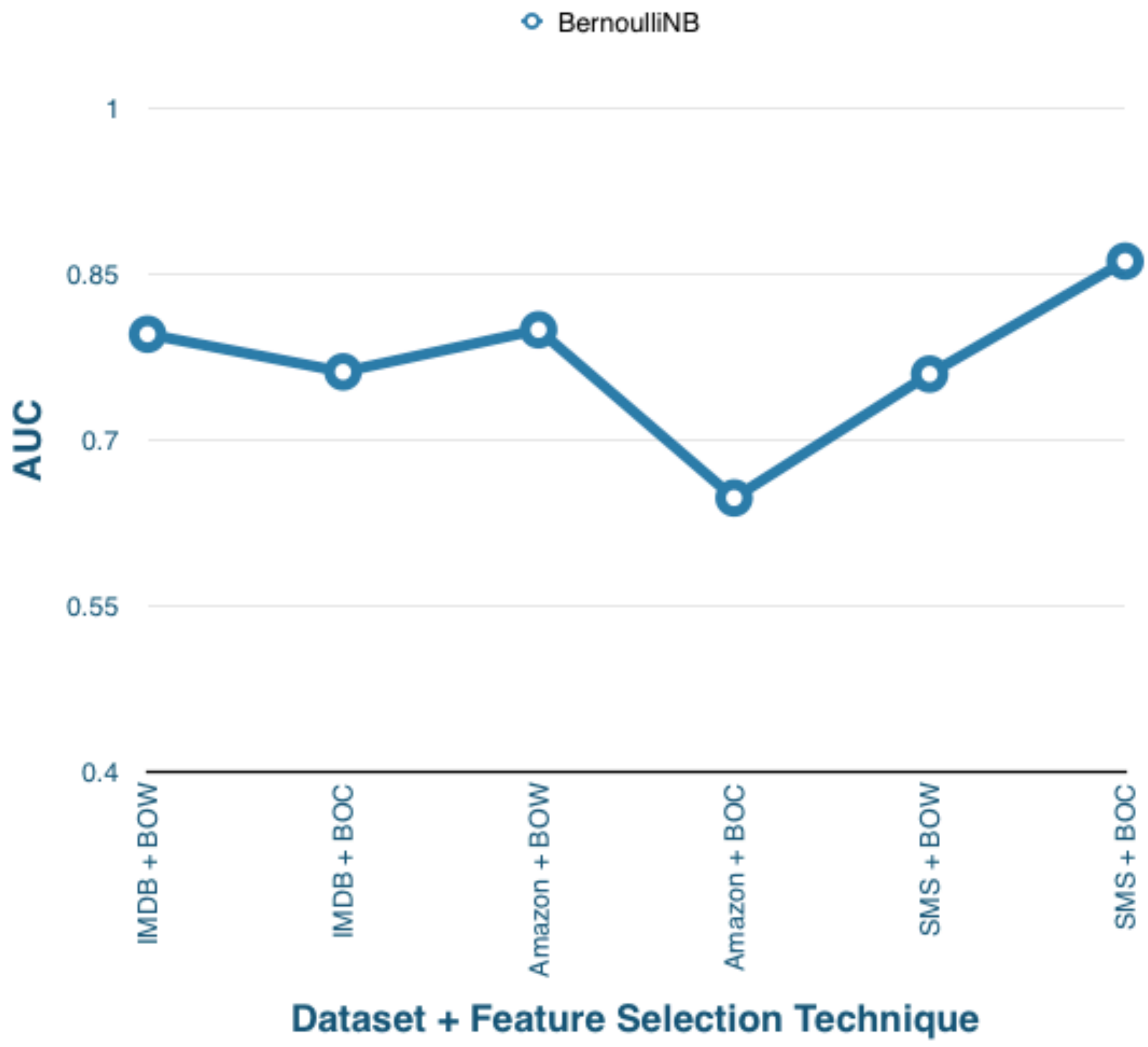


Figure 3.2.6 Performance of BernoulliNB

GaussianNB's scores were consistent for the IMDB and Amazon datasets. Using BOW and BOC on the SMS dataset brought about steep changes in the scores, with BOW greatly increasing the score and BOC bringing it down dramatically

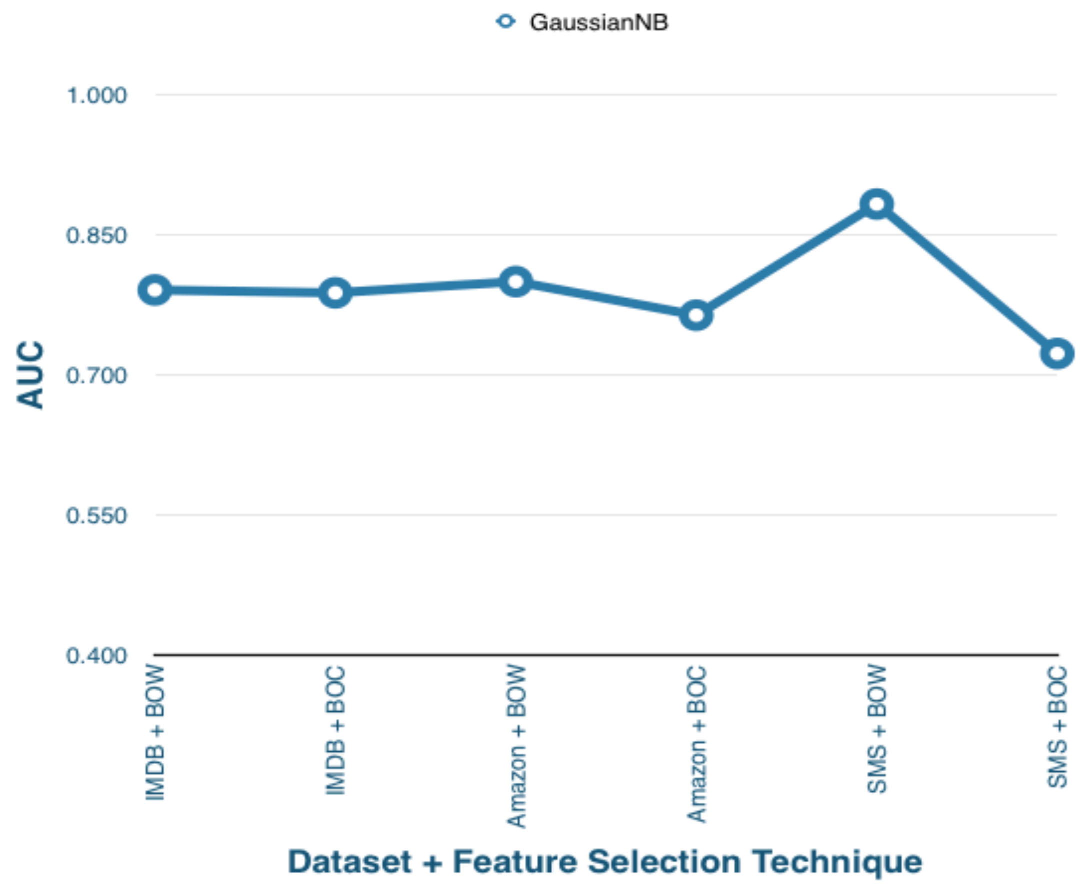


Figure 3.2.7 Performance of GaussianNB

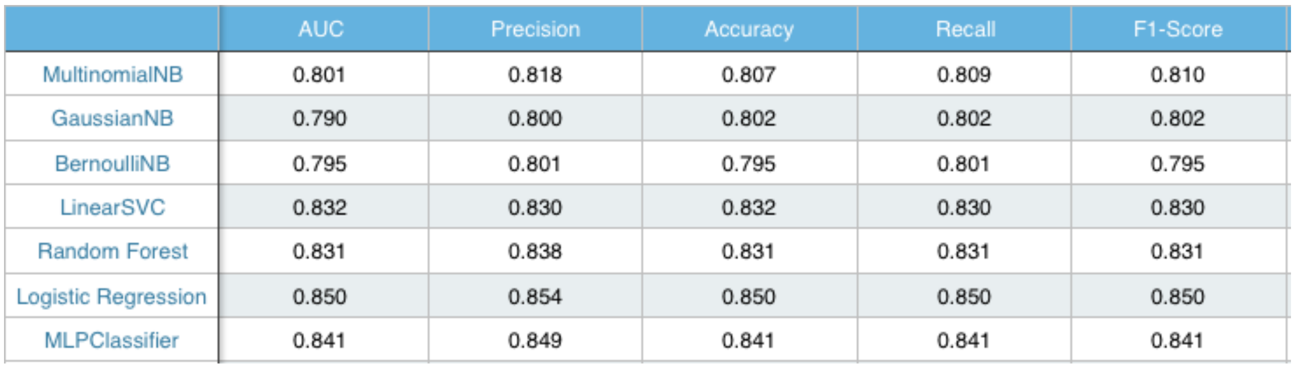
3.2.2 Part 2: Comparison between classifiers

In our project, we worked on semi-supervised learning techniques based on EM model. The experiments consisted of training various classifiers with some labelled data and then predicting the expected labels or classes for the unlabelled data. Finally, the classifier was re-trained using both the labelled and the unlabelled data with the predicted labels, and the classes were predicted on the test data. Different datasets were used to experiment and to show how the dataset and the type of data were on of the biggest factors in the performance of a classifier. To analyse the results, we will look at each dataset and draw a comparison between the classifiers used. The metric scores are an average of the scores produced for the individual labels.

3.2.2.1 IMDB Large Movie Review

3.2.2.1.1 Bag of Words

The Bag of Words model learns a vocabulary from all of the documents, then models each document by counting the number of times each word appears. In the IMDB data, we have a very large number of reviews, which will give us a large vocabulary. To limit the size of the feature vectors, we should choose some maximum vocabulary size. In our experiments, we have used the 4,000 most frequent words considering that stop words have already been removed. Table 11 shows the various metric scores of all the classifiers that we used with the Bag of Words model to classify movie reviews from the IMDB Large Movie Review Dataset.

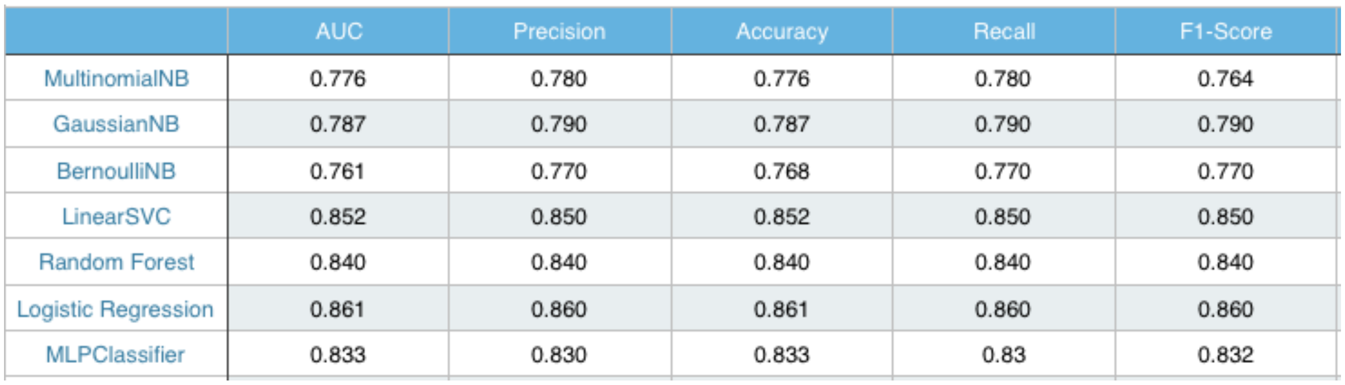


	AUC	Precision	Accuracy	Recall	F1-Score
MultinomialNB	0.801	0.818	0.807	0.809	0.810
GaussianNB	0.790	0.800	0.802	0.802	0.802
BernoulliNB	0.795	0.801	0.795	0.801	0.795
LinearSVC	0.832	0.830	0.832	0.830	0.830
Random Forest	0.831	0.838	0.831	0.831	0.831
Logistic Regression	0.850	0.854	0.850	0.850	0.850
MLPClassifier	0.841	0.849	0.841	0.841	0.841

Table 3.2.1 Metric Scores using BOW model

3.2.2.1.2 Bag of Centroids using K-means

For this, we used Google's pre-trained Word2Vec model, which creates clusters of semantically related words. After a lot of experimentation, based on the accuracy achieved, we decided to set the value of K equal to one-fifth of the vocabulary size. Working with such a large value of K led to longer run times but better accuracies. Table 12. below depict the various metric scores for the classifiers used with the Bag of Centroids model to classify movie reviews from the IMDB Large Movie Review Dataset.



	AUC	Precision	Accuracy	Recall	F1-Score
MultinomialNB	0.776	0.780	0.776	0.780	0.764
GaussianNB	0.787	0.790	0.787	0.790	0.790
BernoulliNB	0.761	0.770	0.768	0.770	0.770
LinearSVC	0.852	0.850	0.852	0.850	0.850
Random Forest	0.840	0.840	0.840	0.840	0.840
Logistic Regression	0.861	0.860	0.861	0.860	0.860
MLPClassifier	0.833	0.830	0.833	0.83	0.832

Table 3.2.2 Metric Scores using BOC model

Figure 10 is a plot between the AUC and the classifiers using both the Bag of Words and Bag of Centroids model. While BOC performs substantially well with some algorithms, it fails to considerably improve the AUC for others. In some cases, even dropping the AUC score. The classifiers perform quite well, with LogisticRegression giving the best AUC of 0.861 and Precision of 0.860. For this large dataset, both the feature selection techniques have led to almost equivalent results. In the classifiers, such as LinearSVC, Random Forest, Logistic Regression and MLPClassifier that have given better results, LinearSVC, Random Forest, Logistic Regression have shown an improvement with the use of the Bag of Centroids model.

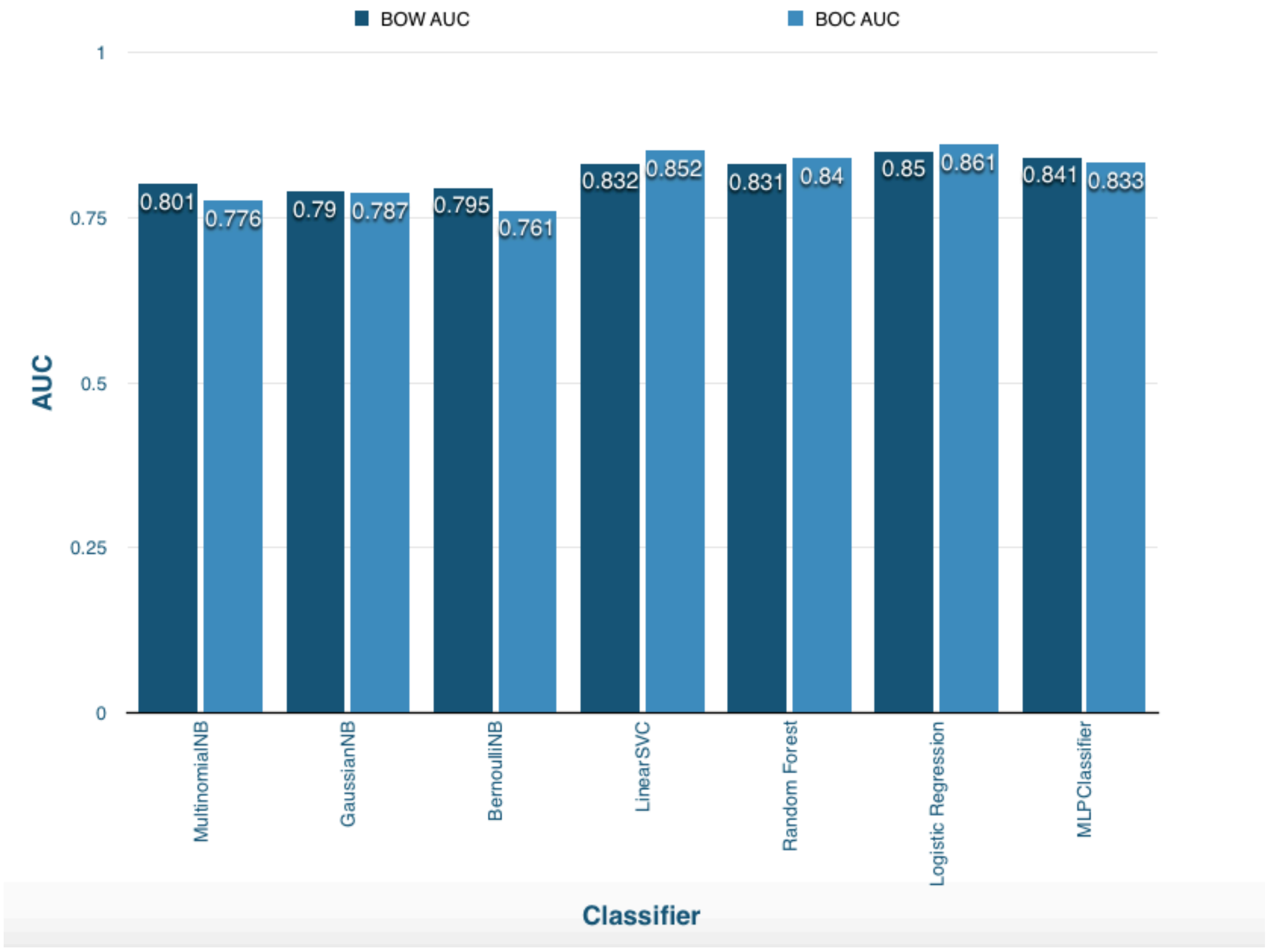


Figure 3.2.8 IMDB: Comparison between BOW and BOC mode

3.2.2.2 Amazon Reviews: Unlocked Mobile Phones

3.2.2.2.1 Bag of Words

We had to prepare this dataset for semi-supervised learning by preparing the unlabelled and testing data. Amazon Reviews dataset had product reviews, which were classified using the algorithms shown in Table 3.2.3 below. These algorithms have used the Bag of Words model for feature extraction.

	AUC	Precision	Accuracy	Recall	F1-Score
MultinomialNB	0.820	0.840	0.819	0.820	0.820
GaussianNB	0.799	0.830	0.771	0.771	0.780
BernoulliNB	0.799	0.830	0.771	0.771	0.780
LinearSVC	0.763	0.810	0.810	0.810	0.810
Random Forest	0.790	0.836	0.834	0.834	0.834
Logistic Regression	0.804	0.860	0.864	0.864	0.864
MLPClassifier	0.788	0.820	0.815	0.820	0.815

Table 3.2.3 Metric Scores using BOW model

3.2.2.2.2 Bag of Centroids using K-means

Table 3.2.4 below show the metric scores of the various classifiers using BOC model for the classification of product reviews from the Amazon Reviews: Unlocked Mobile Phones dataset.

	AUC	Precision	Accuracy	Recall	F1-Score
MultinomialNB	0.810	0.840	0.828	0.830	0.830
GaussianNB	0.763	0.820	0.818	0.810	0.818
BernoulliNB	0.647	0.730	0.569	0.570	0.580
LinearSVC	0.769	0.810	0.805	0.810	0.810
Random Forest	0.798	0.840	0.835	0.840	0.840
Logistic Regression	0.792	0.830	0.835	0.840	0.830
MLPClassifier	0.760	0.800	0.793	0.790	0.793

Table 3.2.4 Metric Scores using BOC model

Figure 3.2.9 is a comparison between the classifiers using BOW and BOC. In comparison to the scores of the classifiers used for classifying IMDB dataset, results of the classifier, used for classifying product reviews from Amazon, are very different. MultinomialNB gives the best result for this dataset using both BOW and BOC. BernoulliNB plummeted to 0.647 using BOC but with the BOW model, it gained substantially.

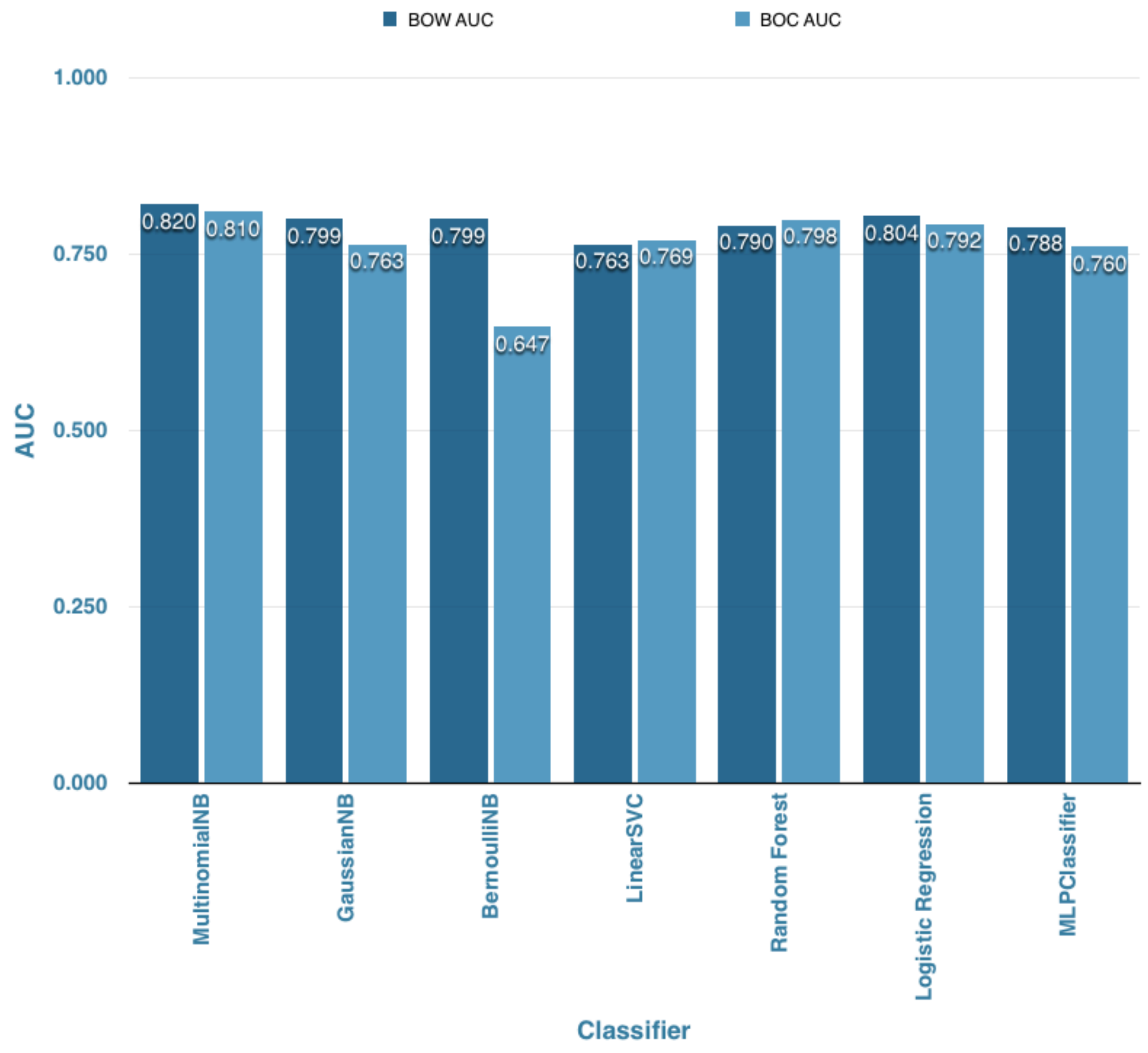


Figure 3.2.9 Comparison between the classifiers using BOW and BOC

3.2.2.3 SMS Spam Collection

From UCI's machine learning repository, this dataset was prepared by us for the implementation of semi-supervised learning techniques. This dataset produced exceptional Precision scores for almost all the classifiers while the AUC score sometimes was far from the precision or the accuracy achieved. In this dataset, the testing data was much smaller than the other datasets.

3.2.2.3.1 Bag of Words

Using this model, we achieved an AUC of 99.4% with MultinomialNB.

	AUC	Precision	Accuracy	Recall	F1-Score
MultinomialNB	0.994	0.990	0.990	0.990	0.990
GaussianNB	0.882	0.920	0.882	0.882	0.890
BernoulliNB	0.759	0.930	0.932	0.930	0.920
LinearSVC	0.909	0.980	0.975	0.980	0.975
Random Forest	0.784	0.950	0.941	0.941	0.930
Logistic Regression	0.863	0.960	0.963	0.960	0.960
MLPClassifier	0.905	0.970	0.969	0.970	0.970

Table 3.2.5 Metric Scores using BOW model

3.2.2.3.2 Bag of Centroids (Vector Quantization) using K-means

Bag of Centroids helped in producing great accuracies too. MultinomialNB again predicted with 99% precision and accuracy and with an AUC of 98.5%.

	AUC	Precision	Accuracy	Recall	F1-Score
MultinomialNB	0.985	0.990	0.990	0.990	0.990
GaussianNB	0.722	0.860	0.686	0.690	0.740
BernoulliNB	0.861	0.870	0.864	0.860	0.864
LinearSVC	0.905	0.970	0.969	0.970	0.969
Random Forest	0.727	0.930	0.926	0.930	0.910
Logistic Regression	0.863	0.960	0.963	0.960	0.960
MLPClassifier	0.911	0.960	0.963	0.960	0.960

Table 3.2.6 Metric Scores using BOC model

The results on this dataset were marginally better, and that could be because of multiple factors such as the quality of the data, number of word vectors, size of the dataset, etc. Between BOW and BOC, BOW performed better than BOC with every classifier except for MLPClassifier, LinearSVC and LogisticRegression, in which case the results were almost similar.

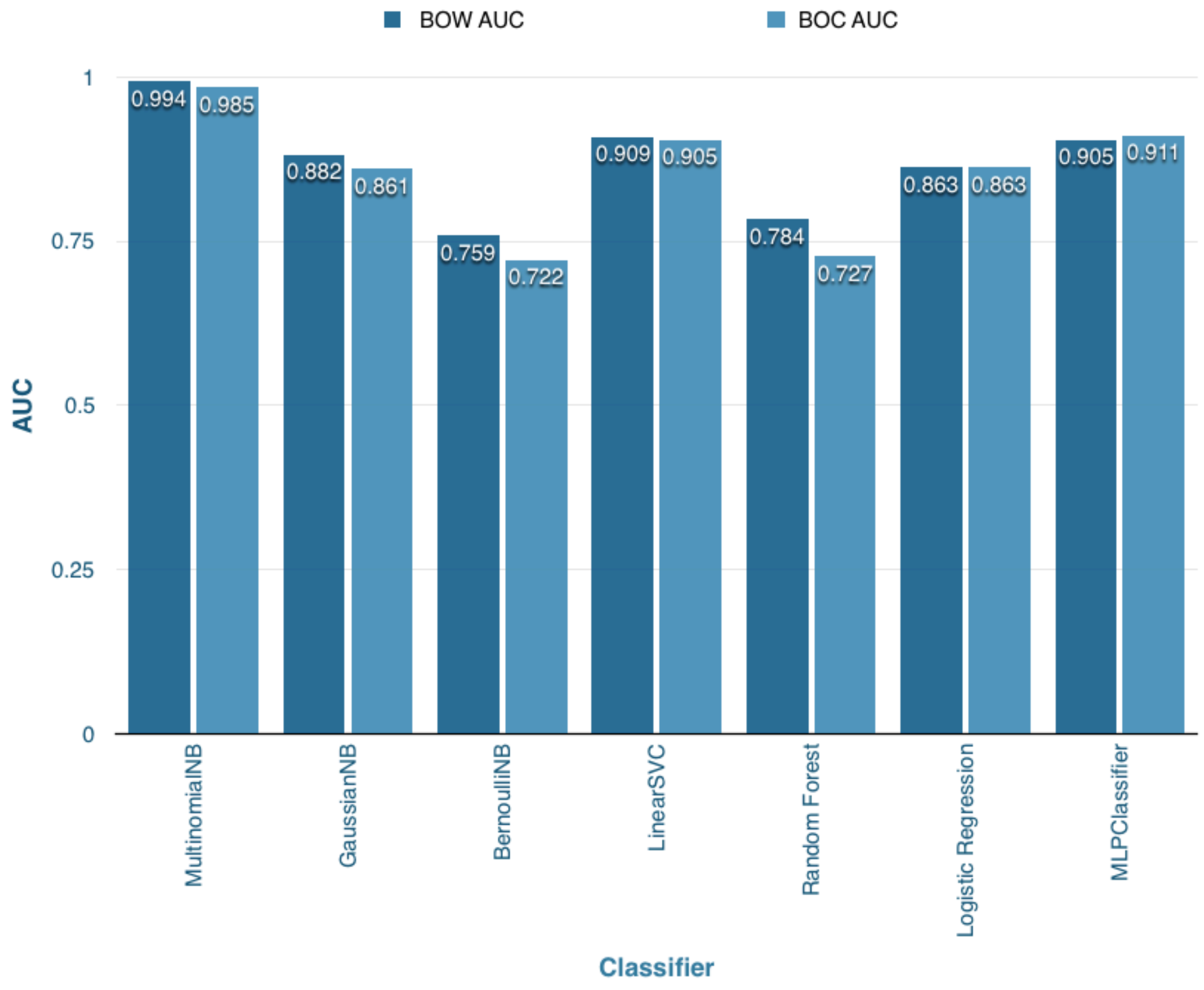


Figure 3.2.10 Comparison between the classifiers using BOW and BOC

3.2.3 Part 3: Receiver Operating Characteristic Curve

3.2.3.1 Bernoulli Naïve Bayes

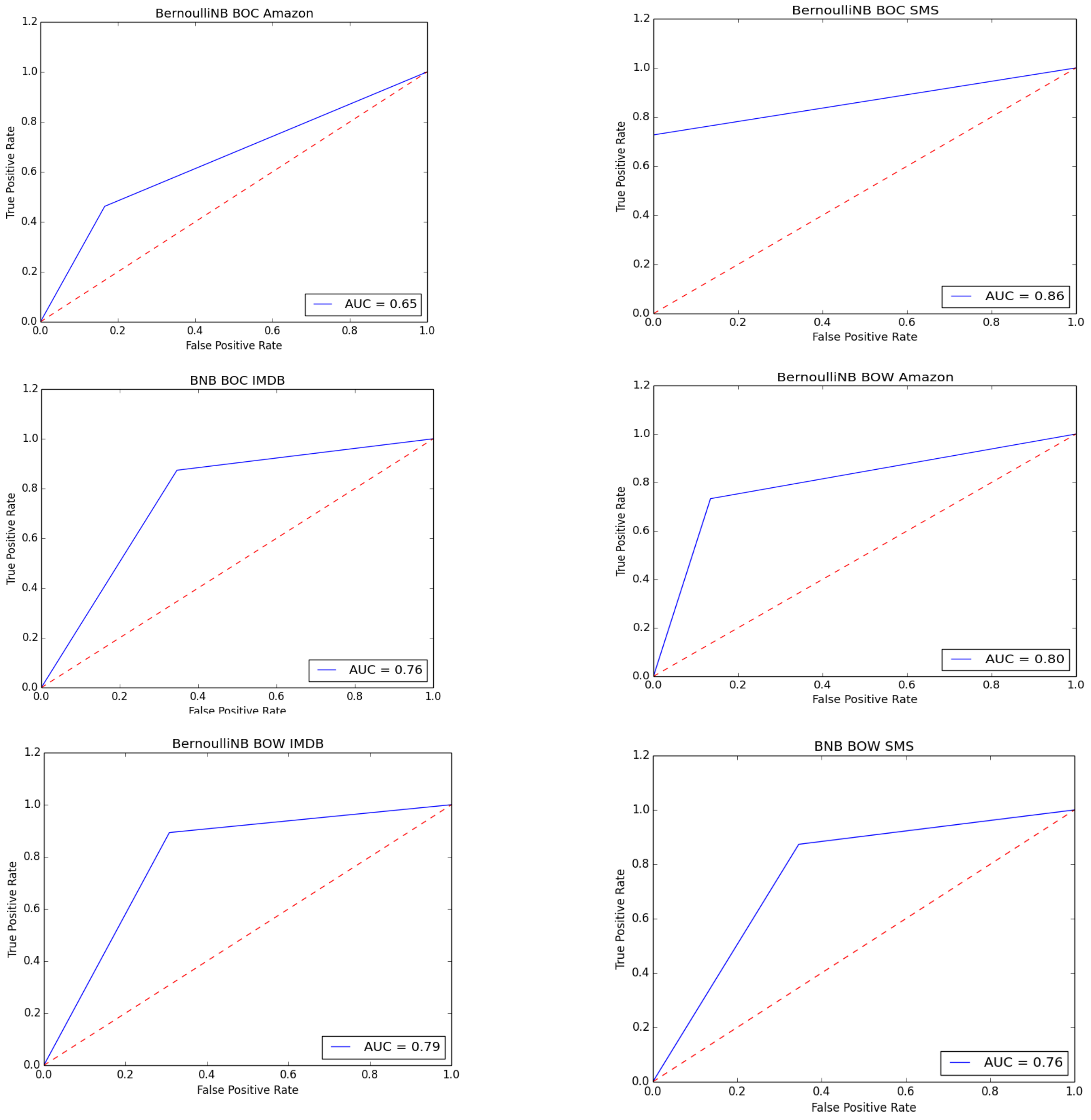


Figure 3.2.11 BernoulliNB ROC Curves

3.2.3.2 Gaussian Naïve Bayes

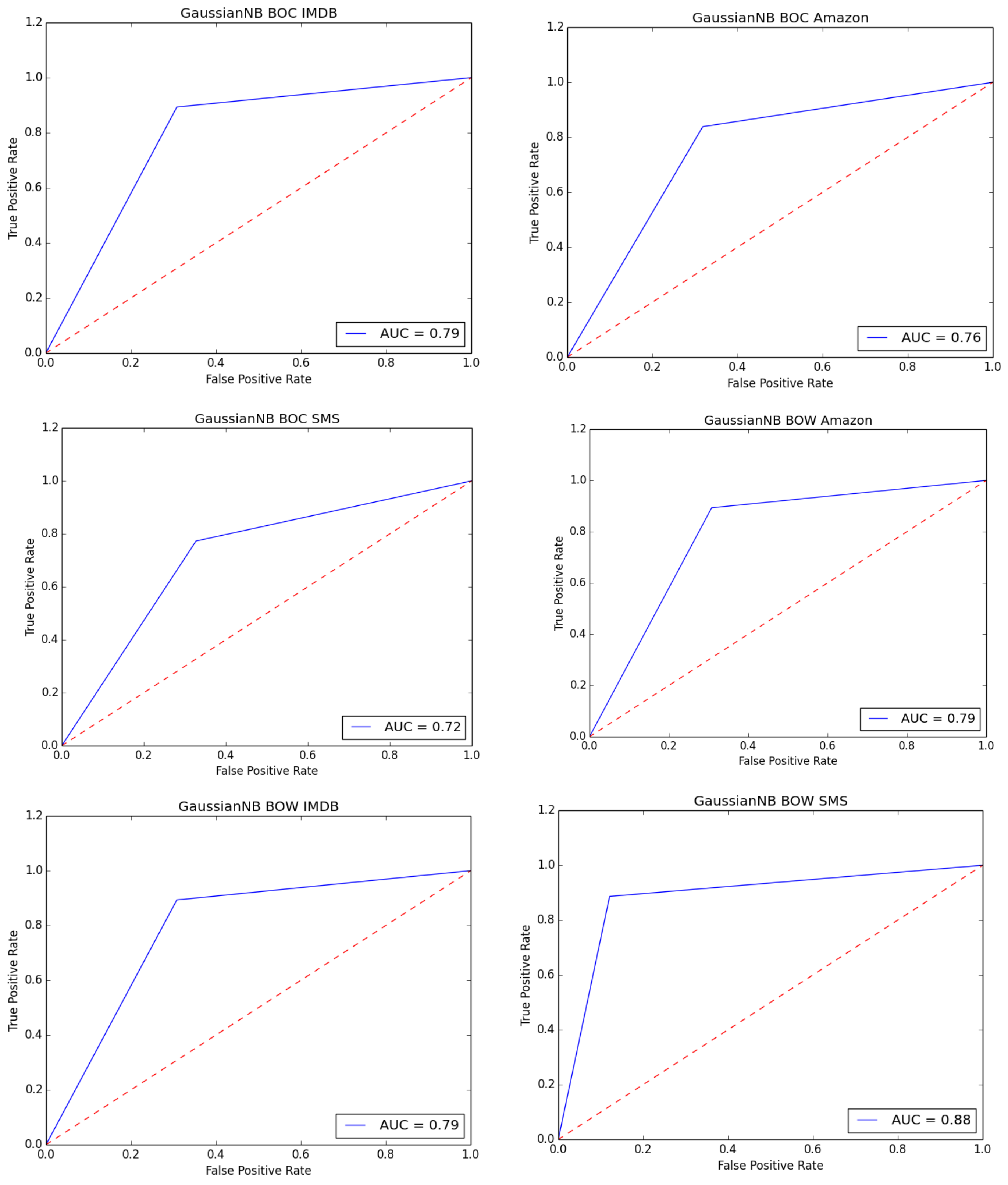


Figure 3.2.12 GaussianNB ROC Curves

3.2.3.3 Logistic Regression

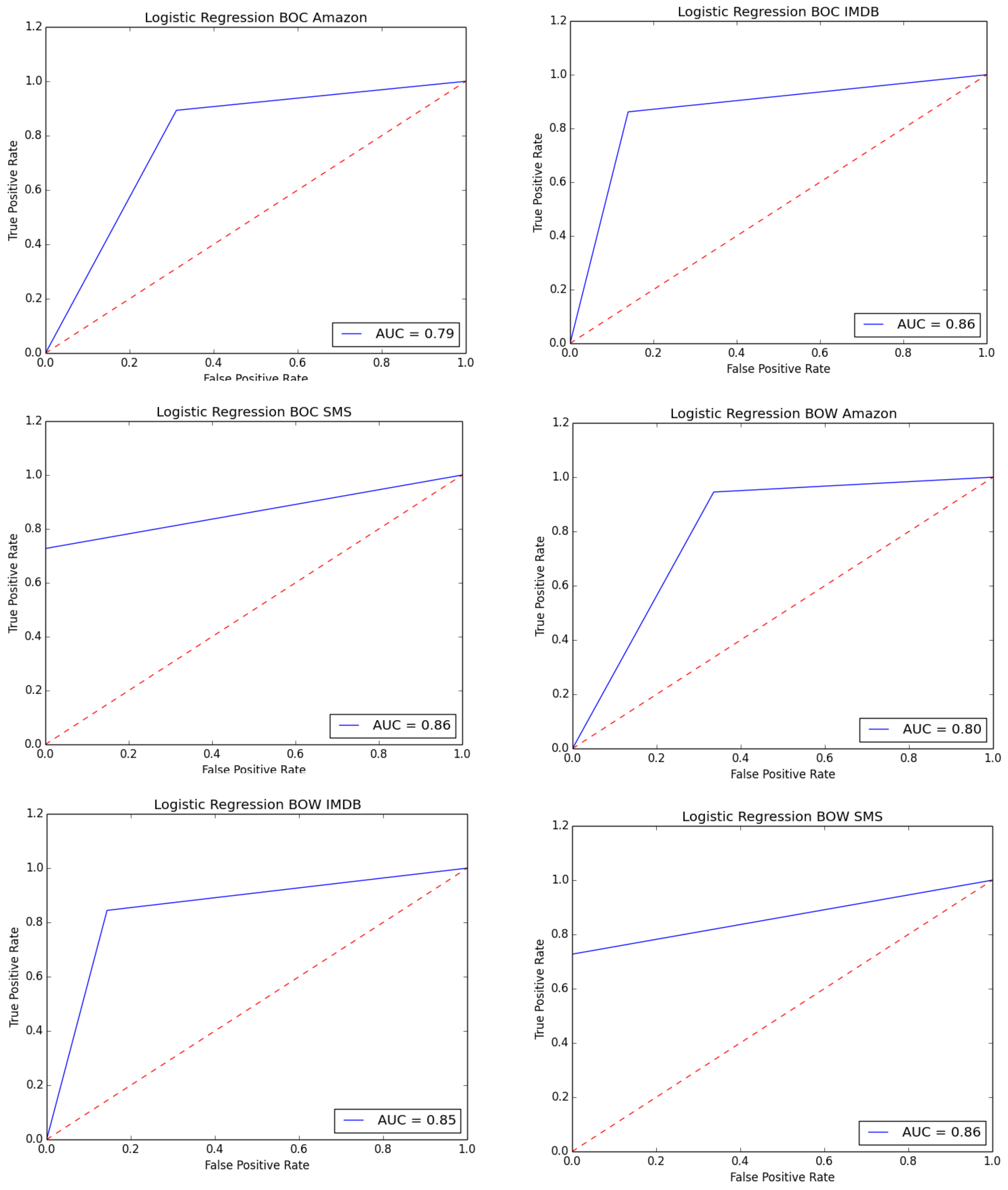


Figure 3.2.13 LogisitcRegression ROC Curves

3.2.3.4 LinearSVC

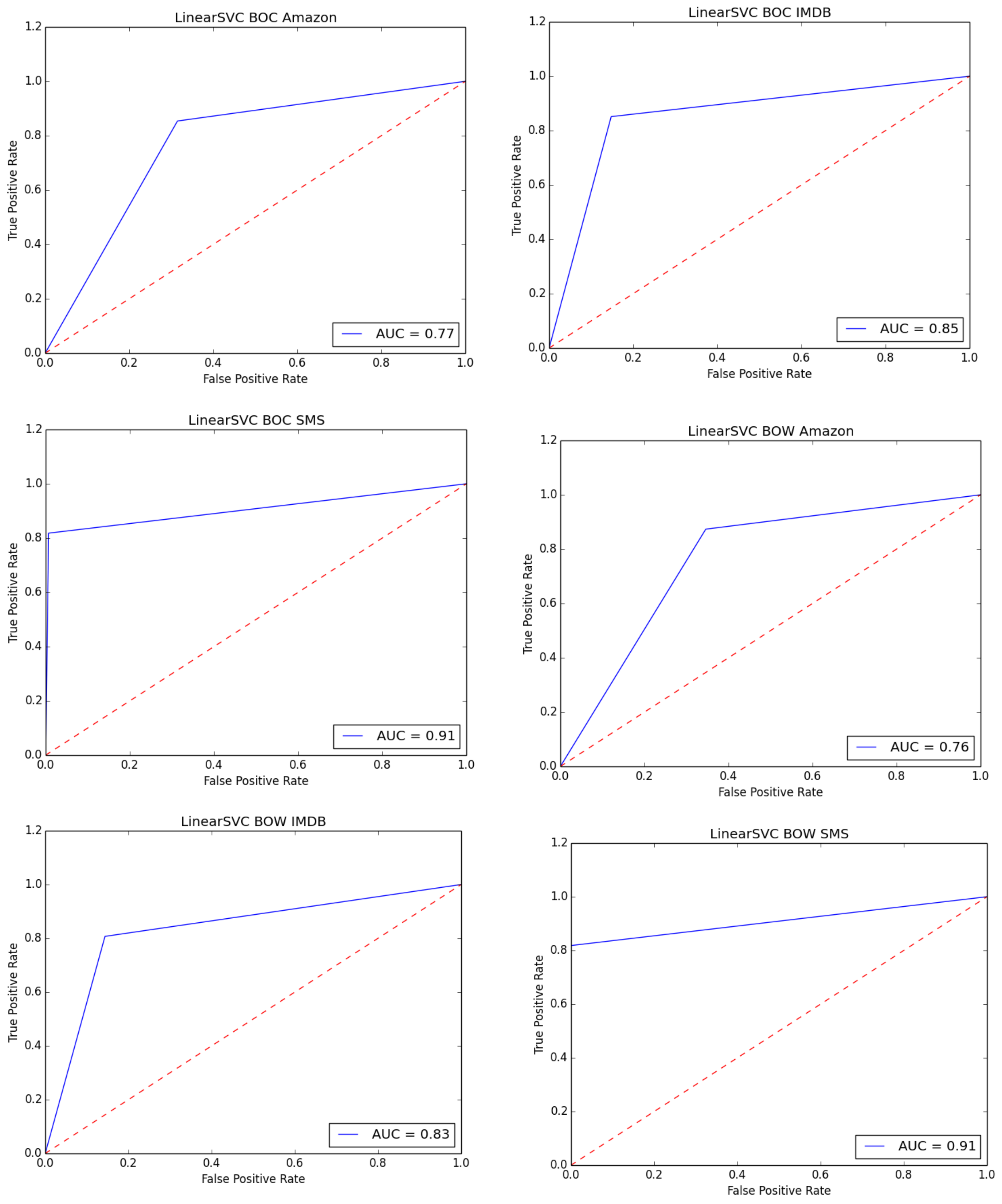


Figure 3.2.14 LinearSVC ROC Curves

3.2.3.5 Multinomial Naïve Bayes

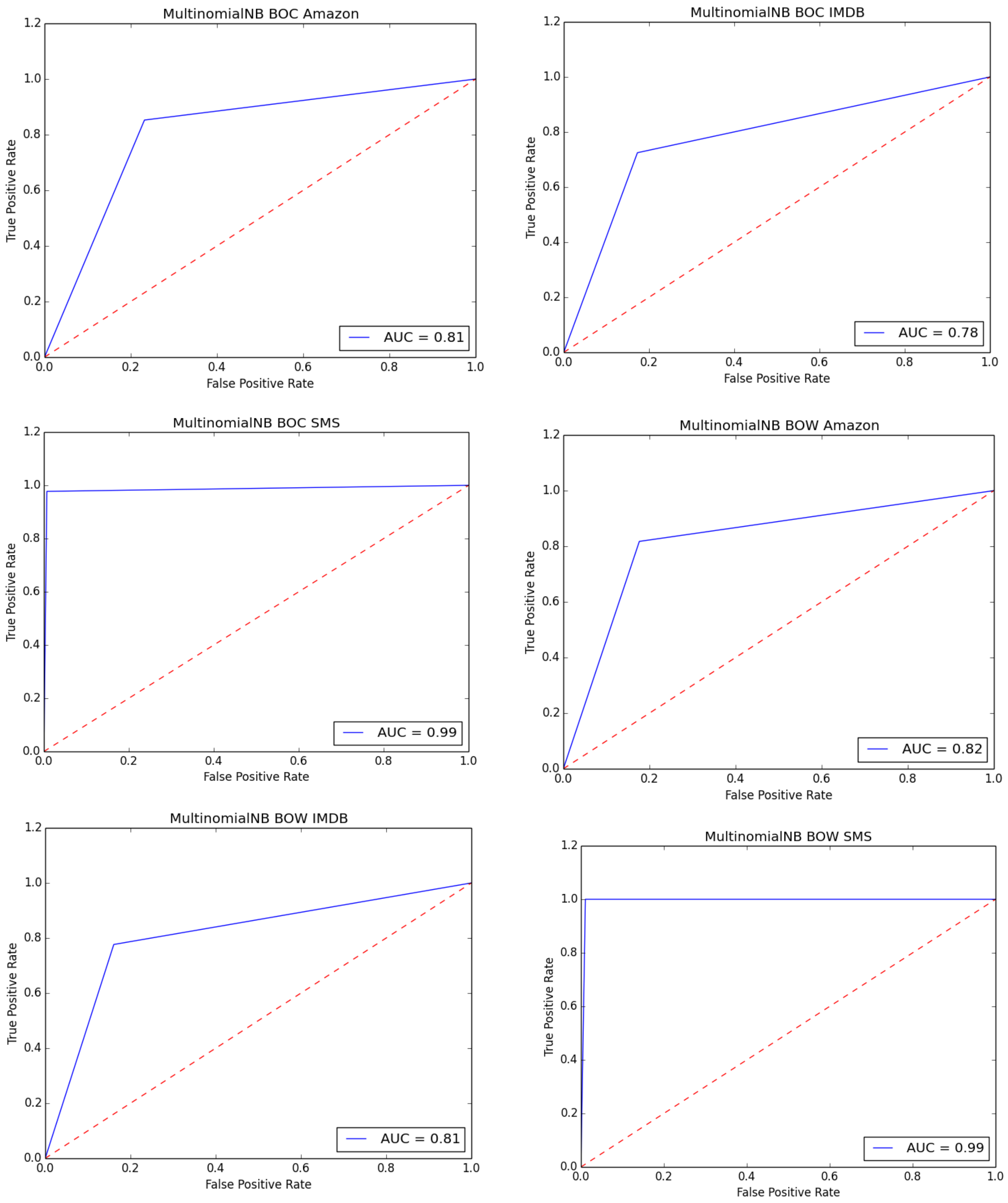


Figure 3.2.15 MultinomialNB ROC Curves

3.2.3.6 MLPClassifier Neural Network

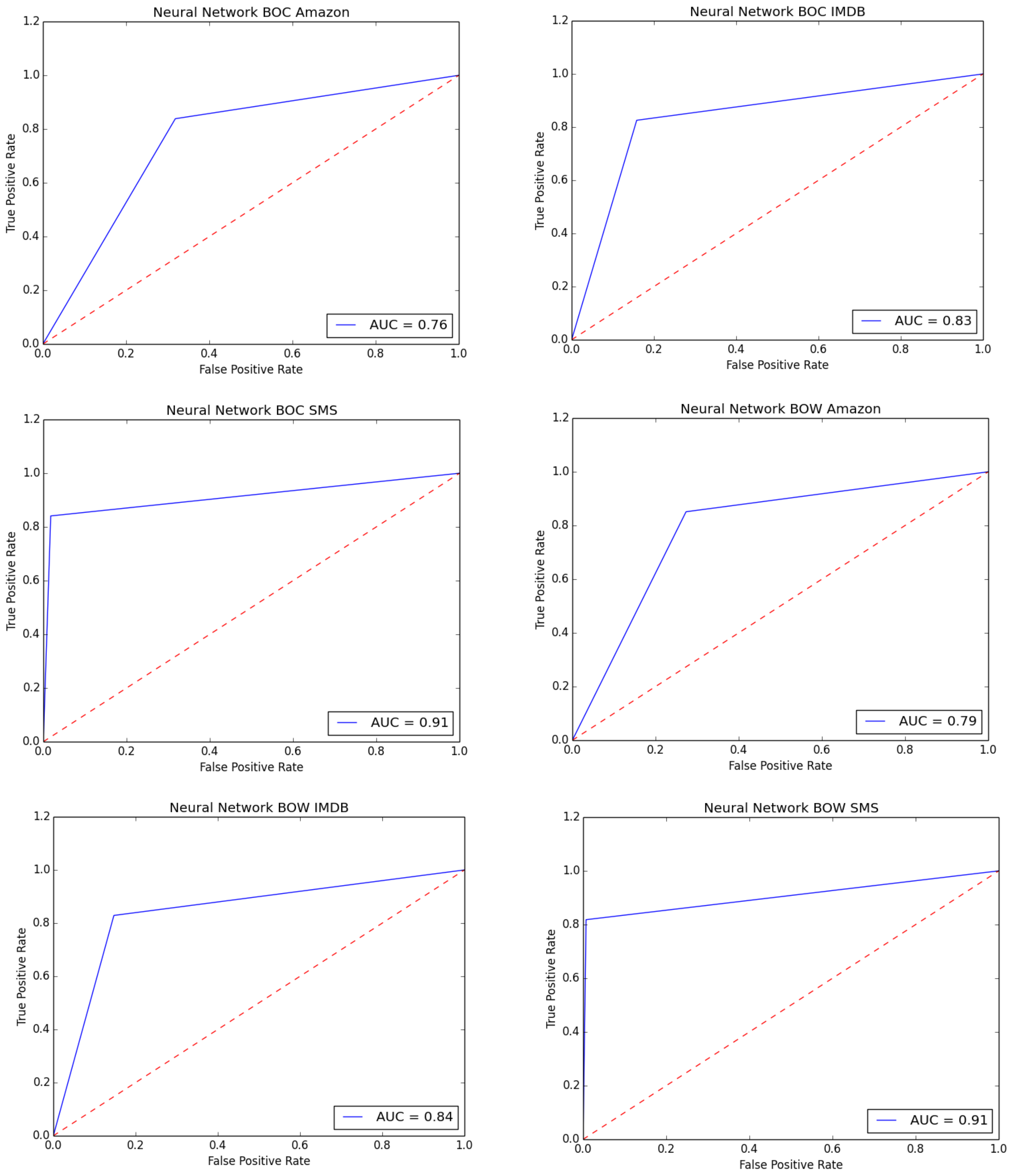


Figure 3.2.16 Neural Network ROC Curves

3.2.3.7 Random Forest

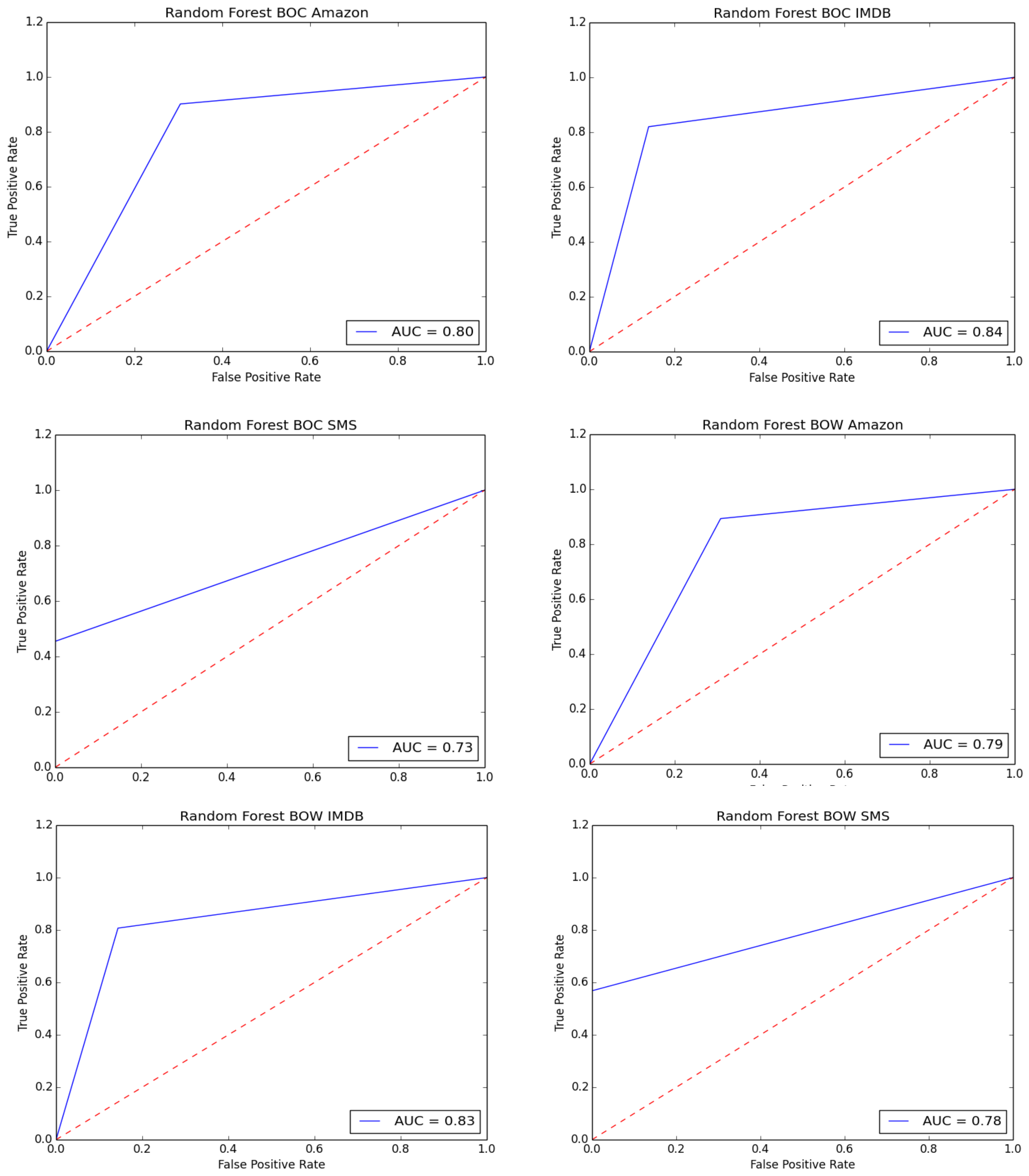


Figure 3.2.17 Random Forest ROC Curves

Chapter 4

Conclusion

In this paper we have studied the use of various classifiers which employ different feature selection techniques such as Bag of Words and Bag of Centroids using K-means. These classifiers are based on the EM algorithm for semi-supervised learning in text classification. All the datasets that we used were binary classification problems and the results obtained are quite promising, considering the high cost of human annotations involved in producing labelled training data. We drew comparisons between the two feature selection techniques, Bag of Words and Bag of Centroids. In our experiments, Bag of centroids showed little or no improvement in the AUC scores of classifiers and in most cases declined the AUC scores when classifying relatively smaller datasets such as the Amazon and SMS dataset. But in the case of IMDB dataset, which was quite large, Bag of Centroids helped improve the scores in better performing classifiers such as Logisitic Regression, LinearSVC and Random Forest. Kaggle hosted a competition called Bag of Words Meets Bags of Popcorn, which required its participants to classify the IMDB Large Movie Review Dataset. The results that our classifiers have produced, using the EM model and the feature selection techniques, are comparable to the leaderboard scores of the competition. LogisticRegression using Bag of Centroids produced an accuracy of 86.1%, which is close to the top 6 submissions made. The scores produced on the synthetic data created out of the UCI SMS Spam Collection, were exceptionally better. MultinomialNB performed consistently and exceptionally well using both BoW and BoC and produing an AUC of about 99%. The precision scores of other classifiers were significantly better too as they ranged from 92%-97% using the BoW model and from 86%-96% using the BoC model. These scores were obtained during the final days of our project and thorough research must be done to understand the reasons for such great scores.

Chapter 5

Future Scope

We built a neural network using the MLPClassifier, which had only a single hidden layer with 15 neurons. Using this, we have only scratched the surface of what promises to show increasingly better results in the domain of semi-supervised learning in text classification. There exists a huge scope of design and complexity improvement in neural networks which may produce marginally better results.

We touched the surface of the question: What happens when the dataset becomes too large or too small? We will have to experiment with more datasets to find the right trend. The choice of classifier still seems to be done quite haphazardly. One would assume that the choice of classifier depends on the feature set and corpus - perhaps SVM is more powerful for smaller feature sets, and Bayes for larger - but there is no real consensus, and other methods are still used. It is worth noting that our classifiers performed worse after adding more information or on bigger datasets. Theoretically, one would imagine a classifier which can tell the features or methods that are dragging down its performance.

We understand the importance of working with multiple languages in text classification to account for the variations in languages used by the reviewers and thus training the classifiers to be able to work with multiple languages is essential.

Chapter 6

Bibliography and References

Books:

[1] Semi-supervised learning by Chapelle, Olivier; Schölkopf, Bernhard; Zien, Alexander (2006) Cambridge, MIT Press

[2] Data mining: concepts and techniques (3rd edition) by Jain Pei, Jiawei Han, Micheline Kamber, publisher: Elsevier

Research papers:

[1] Liu Yang, Rong Jin and Rahul Sukthankar, Semi-supervised Learning with Weakly-Related Unlabeled Data: Towards Better Text Categorization,2009.

[2] Jiang Su, Jelber Sayyad-Shirabad, Stan Matwin,Large Scale Text Classification using Semisupervised Multinomial Naive Bayes, 2011.

[3] Gargi Joshi, Semi Supervised Text Classification with Universum**,** 2015.

[4] Ishtiaq Ahmed, Rahman Ali, Donghai Guan, Young-Koo Lee, Sungyoung Lee and TaeChoong Chung, Semi-supervised learning using frequent itemset and ensemble learning for text classification, 2014.

[5] Ion Muslea, Steven Minton, Craig A. Knoblock, Active + Semi-Supervised Learning = Robust Multi-View Learning, 2009.

[6] Ronei Marcos de Moraes and Liliane dos Santos Machado,Gaussian Naive Bayes for Online Training Assessment in Virtual Reality-Based Simulators,2009.

[7] Andrew McCallum and Kamal Nigam,A Comparison of Event Models for Naive Bayes Text Classication,2000.

[8] Rie Kubota Ando, Tong Zhang A High-Performance Semi-Supervised Learning Method for Text Chunking,2005.

[9] Keziah Plattner ,Laura Hunter and Lilith Wu, Predicting Author Gender for Yelp Reviews,2016.

[10] Olivier Chapelle, Jason Weston, Bernhard Scholkopf, Cluster Kernels for Semi-Supervised Learning,2002.

[11] Mikhail Bilenko, Sugato Basu and Raymond J. Mooney, Integrating Constraints and Metric Learning in Semi-Supervised Clustering,2004.

[12] Mita K. Dalal and Mukesh A. Zaveri,Automatic Text Classification: A Technical Review,2011

[13] Xiaojin Zhu,Semi-Supervised Learning Literature Survey,2008.

[14] K.Aas and A.Eikvil, Text Categorization: A survey. Technical report, Norwegian Computing Center, June, 1999.

[15] Tyler (Tian) Lu, Fundamental Limitations of Semi-Supervised Learning,2009.

[16] Xiaojin Zhu, Zoubin Ghahramani ,John Lafferty , Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions,2006.

[17] Andrew Carlson ,Justin Betteridge ,Richard C. Wang, Coupled Semi-Supervised Learning for Information Extraction,2010.

[18] Qiuhua Liu, Xuejun Liao, Hui Li, Jason Stack, and Lawrence Carin. Semi-supervised multitask learning. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2009.

[19] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In Proc. of NAACL, 2006.

[20] Marius Pa¸sca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. Names and similarities on the web: fact extraction in the fast lane. In Proc. of ACL, 2006.

[21] Marco Pennacchiotti and Patrick Pantel. Entity extraction via ensemble semantics. In Proc. of EMNLP, 2009.

[22] Olivier Chapelle, Bernhard Sch□olkopf, and Alexander Zien, editors. Semi-Supervised Learning. MIT Press,2006.

[23] Fabio Cozman and Ira Cohen. Risks of semi-supervised learning: How unlabeled data can degrade performance of generative classiers,2006.

[24] Sanjoy Dasgupta, Michael L. Littman, and David A. McAllester. Pac generalization bounds for co-training,2001.

[25] Luc Devroye, Laszlo Gyorfi and Gabor Lugosi. A Probabilistic Theory of Pattern Recognition (Stochastic Modelsling and Applied Probability). 1997.

[26] Andrzej Ehrenfeucht, David Haussler, Michael J. Kearns, and Leslie G. Valiant. A general lower bound on the number of examples needed for learning,1989.

[27] Ran El-Yaniv and Dmitry Pechyony. Stable transductive learning, 2006.

[28] Ran El-Yaniv and Dmitry Pechyony. Transductive rademacher complexity and its Applications,2007.

[29] Claudio Gentile and David P. Helmbold. Improved lower bounds for learning from noisy examples: and information-theoretic approach,1998.

[30] David Haussler. Decision theoretic generalizations of the pac model for neural net and other learning,2002.

[31] Shuicheng Yan, Huan Wang, Semi-supervised Learning by Sparse Representation,2009.

[32] [14] M. Seeger. Learning with labeled and unlabeled data. Technical report, Univ. of Edinburgh, 2001.

[33] V. Sindhwani and S. S. Keerthi. Large scale semi-supervised linear support vector machines. In Proc.ACM SIGIR, 2006.

[34] Kamal Nigam,Andrew McCallum,Tom Mitchell, Semi-Supervised Text Classification Using EM, 2005

[35] M. Szummer and T. Jaakkola. Information regularization with partially labeled data. In Proc. NIPS, 2002.

[36] L. Yang, R. Jin, C. Pantofaru, and R. Sukthankar. Discriminative cluster refinement: Improving object category recognition given limited training data. In Proc. CVPR, 2007.

[37] Y. Yang. An evaluation of statistical approaches to text categorization. Journal of Info. Retrieval, 1999.

[38] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In Proc.ICML, 1997.

[39] X. Zhu. Semi-supervised learning literature survey. Technical report, UW-Madison, Comp. Sci., 2006.

[40] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In Proc. ICML, 2003.

[41] K. Bennett and A. Demiriz. Semi-supervised support vector machines. In Proc. NIPS, 1998.

[42] S. Boyd and L. Vandenberghe. Convex optimization. Cambridge University Press, 2004.

[43] C. Burges. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2), 1998.

[44] M. Fazel, H. Hindi, and S. Boyd. A rank minimization heuristic with application to minimum order system approximation. In Proc. American Control Conf., 2001.

[45] Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm,1999.

[46] Forman, G. and Cohen, I. Learning from little: Comparison of classi¯ers given little training, 2004.

[47] Hall, Mark, Frank, Eibe, Holmes, Geo®rey, Pfahringer,Bernhard, Reutemann, Peter, and Witten, Ian H.The weka data mining software: an update,2009.

[48] Lewis, David D., Yang, Yiming, Rose, Tony G., and Li, Fan. Rcv1: A new benchmark collection for text categorization research. Journal of Machine Learning Research, 2004.

[49] Mann, Gideon S. and McCallum, Andrew. Generalized expectation criteria for semi-supervised learning with weakly labeled data. Journal of Machine Learning Research, 2010.

[50] Pak, Alexander, and Patrick Paroubek. "Twitter as a Corpus for Sentiment Analysis and Opinion Mining, 2010.

[51] Miller, D., & Uyar, H, A mixture of experts classifier with learning based on both labelled and unlabelled data,1997.

[52] Mitchell, T,The role of unlabeled data in supervised learning. Proceedings of the Sixth International Colloquium on Cognitive Science,1999.

[53] Bennett, K., & Demiriz, Semi-supervised support vector machines.Advances in Neural Information Processing Systems,2000.

[54] Bansal, N., Blum, A., & Chawla, S. (2002). Correlation clustering,2002.

[55] Bar-Hillel, A., Hertz, T., Shental, N., & Weinshall,Learning distance functions using equivalence relations,2003.

[56] Basu, S., Banerjee, A., & Mooney, R. J.,Semi-supervised clustering by seeding,2002.

[57] Basu, S., Bilenko, M., & Mooney, R. J.,A probabilistic framework for semi-supervised clustering,2004.

[58] Bilenko, M., & Mooney, R. J.,Adaptive duplicate detection using learnable string similarity measures,2003.

[59] Miao, Yingbo Wei, Gang Yu, Zheyuan Sheng, XinThe implementation of text categorization with term association, 2003