University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

Sentiment Analysis Using Customizable Naive Bayes Classifier

A special problem in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Computer Science

Submitted by:

Alyssa Jayne I. Pasia

May 2018

Permission is given for the following people to have access to this SP:

Available to the general public	Yes
Available only after consultation with author/SP adviser	No
Available only to those bound by confidentiality agreement	No

ACCEPTANCE SHEET

The Special Problem entitled "Sentiment Analysis Using Customizable Naive Bayes Classifier" prepared and submitted by Alyssa Jayne I. Pasia in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

Perlita E. Gasmen, M.S. (*cand.*)
Adviser

EXAMINERS:

	Approved	**Disapproved**
1. Gregorio B. Baes, Ph.D. (*cand.*)	_____	_____
2. Avegail D. Carpio, M.S.	_____	_____
3. Richard Bryann L. Chua, Ph.D. (*cand.*)	_____	_____
4. Marvin John C. Ignacio, M.S. (*cand.*)	_____	_____
5. Ma. Sheila A. Magboo, M.S.	_____	_____
6. Vincent Peter C. Magboo, M.D., M.S.	_____	_____
7. Geoffrey A. Solano, Ph.D. (*cand.*)	_____	_____

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

Ma. Sheila A. Magboo, M.S.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences
and Mathematics

Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Sciences
and Mathematics

Leonardo R. Estacio Jr., Ph.D.
Dean
College of Arts and Sciences

Abstract

Huge amount of data are available on the internet. Due to this, there is an increasing interest in automatically obtaining valuable information from these data using Sentiment Analysis. A lot of methods and datasets are used to build models for classifying. This project aims to develop a system that lets the user decide on what datasets and preprocessing techniques are to be used on a Naive Bayes Classifier model.

Keywords: Sentiment analysis, Opinion mining, Text classification, Naive Bayes classification, Bayes rule

Contents

List of Figures

I. Introduction

A. Background of the Study

Human life consists of emotions and opinions that influence the way humans think, move, and act [2]. In the past few years, data on the web has been increasing exponentially [3] and has been receiving great attention as a new source of individual experiences and opinions [2]. Opinions about movies, products, etc. can be found in discussion forums, social networks, web blogs, and so on [4]. A large amount of user generated content comes from the growth of social web [5]. The online users are free to express their feelings, impressions, and thoughts concerning certain topics in social networking sites [1]. Huge amount of comments, feedbacks, articles, suggestions, etc. are available on the internet [6]. New opportunities and challenges arise with the growing availability and popularity of opinion-rich resources such as personal blogs and online review websites [7].

Due to the easy accessibility of machine-readable documents on the web, there is an increase in interest in methods for automatically extracting and analyzing web documents [2]. A common concern for organizations is to be able to automatically obtain valuable information from these data by extracting the opinion or sentiment from a message [3].

Sentiment analysis is among the most researched topics in Natural Language Processing which involves the extraction of subjective information from documents to determine its contextual polarity (positive, negative, or neutral) [8].

Our decisions are sometimes influenced by the opinions of others. Before the widespread use of the internet, we used to ask our friends about their opinion on gadgets, appliances, books, or movies before actually making any purchase but now, people can use information technologies in understanding and seeking out the opinions of others [7]. Now we can use the internet to take comments, feedbacks, and

suggestions of many people [6]. Consumers usually check opinions of others about a product online when making any purchase [5].

Another way of extending companies' customer satisfaction analysis apart from written surveys is through gathering a large amount of data from the web. Even though the simplest way to collect opinions is through surveys, problems on this approach emerges such as the conduct of a survey for each product, feature, or services, the distribution and timing of the survey, and the reliance on the good will of people that takes the survey [1].

Companies take customer feedback through their websites about their products, its features, or services [6]. Identifying the trends of public opinion in the social media is useful for the purpose of marketing and consumer research [8]. Though these are helpful for both business organizations and individuals, huge amount of data are overwhelming to users [5].

In business consumer industry and online recommendation systems, sentiment classification is very useful [3]. It can be used to get customer feedback about new product launches, ad campaigns, and even in financial markets [8]. From these contents, manufacturers can gather insights from the customers' sentiments about the strengths and weaknesses of their products [5]. Examples are when reviews are analyzed to decide what products they will produce to reduce risk [3], which products or services are popular [9], and when they solve reported problems to enhance their product qualities and discover their competitors' product feedbacks [6]. Based on the reviews and comments of their customers, companies can improve their products and services [4].

B. Statement of the Problem

A lot of classifiers made are trained using different datasets that may not be of the same nature of what the users want to classify. There are classifiers for Facebook

statuses [1], for Tweets in Twitter [4], for movie reviews [7], and many other genres such as restaurant reviews and product reviews. Using a classifier trained specifically for one category may not be ideal to be used for other categories because a word may have totally different contexts in those categories.

Preprocessing techniques to be used are also a topic of interest since not all preprocessing techniques are as effective as they are intended to be when applied from one dataset to another.

C. Objectives of the Study

The aim of this project is to create a tool that lets you choose the dataset and the preprocessing methods to be used for training the model and display the model's performance. The tool accepts a text file input then classify its contents into positive or negative and produce text files containing the statements classified as positive and the statements classified as negative.

1. For training, the system shall be able to:
    (a) Allow the user to browse and open a text file from the computer containing the labeled statements to be used for training
    (b) Select preprocessing methods for the model from the following:
        i. Text normalization
        ii. Negation handling
        iii. Noise words elimination
        iv. Emoticon conversion
        v. Inflated or derived words reduction
    (c) Train the model using the dataset and the preprocessing methods selected
    (d) Compute and display the performance of the resulting model using 10-fold validation

2. For classifying, the system shall be able to:

   (a) Allow the user to browse and open a text file from the computer containing the unlabeled statements to be classified

   (b) Allow the user to enter a filename where the classified outputs will be saved

   (c) Allow the user to choose output file format

   (d) Classify each statement into positive or negative

   (e) Count and display the number of positive and the number of negative statements contained in the output files

   (f) Produce either of the following output types:

      i. Two text files containing all the positive statements and all the negative statements from the input file with the filenames as entered by the user concatenated with "(Positive)" and "(Negative)"

      ii. One text file containing labeled statements in format "*statement*\t*polarity*\n" with polarity equal to 1 if positive and 0 if negative

D. Significance of the Project

This tool is a flexible tool made to cater most of the categories for sentiment classification. There is no need to find a classifying model that suits the user's data and that has the preprocessing methods the data need. The user just have to provide an existing dataset where the mode would learn from and select the preprocessing methods he finds applicable to use or just try all the combinations of preprocessing methods available until he is satisfied with the model's performance.

E. Scope and Limitations

1. The classifier classifies the statements as positive or negative only. There is no statement classified as neutral.

2. The system can only read text inputs.

3. The system only accepts text files as inputs.

4. The classifier can be trained with the English language only.

5. Sarcasm in statements is not catered.

F. Assumptions

1. The contents of the training set file is in the format:

   "*statement*\t*polarity*\n"

   with polarity equal to 1 if positive and 0 if negative.

2. The contents of the input test file is in the format:

   "*statement*\n"

II. Review of Related Literature

In [9] some data sources were listed namely:

1. Blogs: Blogging and blog pages are growing rapidly with an increasing usage of the Internet with blog pages becoming the most popular means of expressing because bloggers record daily events in their lives to express their opinions, feelings, and emotions in a blog. Some of these blogs contain reviews and are used in many studies related to sentiment analysis.

2. Review sites: Available on the Internet are a large growing number of user-generated reviews for products or services which are usually based on the opinions expressed in unstructured format.

3. Dataset: Most of the work in sentiment analysis uses movie reviews which are available as a dataset. Other dataset available is multi-domain sentiment.

4. Micro-blogging: One popular microblogging site is Twitter where users create status messages that sometimes express opinions about different topics. These tweets are also used as data source for sentiment classification.

[9] also stated that a very common linguistic construction that needs to be taken into consideration in sentiment analysis is negation for it affects polarity. It is a difficult yet important aspect of sentiment analysis. They are not only conveyed by common negation words like not, neither, nor but also by many other words that invert the polarity of an opinion expressed such as valence shifters, connectives, and modals.

In [8], they used a dataset of movie reviews publicly available from the Internet Movie Database (IMDb). It is a set of 50, 000 highly polar movie reviews divided equally for training and for testing. Movie reviews were chosen as their dataset

because it covers a wide range of human emotions and most of the adjectives relevant to sentiment classification are captured by it.

A basic filtering step is performed. They removed features/terms that occurs only once. On the basis of mutual information, these features are then further filtered. Mutual information is a quantity that measures the mutual dependence of the two random variables.

They removed duplicate words from the document because they don't add any additional information. The type of Naive Bayes algorithm that they used was called Bernoulli Naive Bayes. It has been found that including just the presence of the word instead of its count improves performance marginally when the training examples are large in number.

Since they used each word as feature, the presence of the word "not" before the word "good" in the phrase "not good" will not be taken into account and the word "good" will be contributing to the positive sentiment rather than to the negative sentiment. They devised a simple algorithm to solve this problem of handling negations using state variables and bootstrapping. They built on the idea of using an alternate representation of the negated forms. A state variable to store the negation state was used in their algorithm. The algorithm transforms a word followed by a "not" or "n't" into "not_" + word. The words read are treated as "not_" + word whenever the negation state variable is set. When a punctuation mark is encountered or when there is double negation, the state variable is reset.

It is possible that many words with strong sentiment occur only in their normal forms in the training set. Negated forms would be of strong polarity but its number might not be adequate for correct classifications. They addressed this problem by adding negated forms to the opposite class along with normal forms of all the features during the training phase. For example, if they find the word "good" in a positive document during the training phase, they increment the count of "good" in

the positive class and also the count of "not_good" in the negative class. This ensures that the "not_" forms are sufficient in number for classification.

Here, the classifier was implemented in Python using hash tables to store the word counts in their respective classes. Before counting the words, they preprocessed the data and applied negation handling during training. Each word is counted only once per document since they were using Bernoulli Naive Bayes.

The results in [8] show that by choosing the right type of features and removing noise by appropriate feature selection, a simple Naive Bayes classifier can be enhanced to match the classification accuracy of more complicated models for sentiment analysis. Due to assumptions on their conditional independence, Naive Bayes classifiers are extremely fast to train and can scale over large datasets. They are also less prone to overfitting and robust to noise.

[7] used and standardized the movie review dataset from IMDb which consists of 1000 documents for each class totaling of 2, 000 documents.

Having more training examples for one class than another (skewed data) causes biased decision boundary weights inducing the classifier to unwittingly prefer one class over the other. Because of this, they used a variant of Naive Bayes classifier called Complemented Naive Bayes classifier that tackles the poor assumptions made by its parent classifier such as uneven training size (skewed data) and the independence assumption in which all features and attributes are treated individually.

The preprocessing and feature selection here were carried out as a sequence of steps. First, string tokenization and punctuation removal wherein all the words present in the text are extracted using the bag-of-words approach while removing the punctuations and symbols except for '?' and '!' since these characters express negative feelings in evaluative texts. Second, they eliminated some of the noise words such as conjunctions, prepositions, etc. and reduced inflated or derived words into their respective stems or root words. For example, "connected", "connecting", "connection"

are reduced to "connect". Another step is the construction of Term-by-Document matrix with both the binary and frequency values separated which is dimensionally reduced by the feature selection process. They first eliminated words by document frequency, that is, they calculated the frequency of each word in all the documents and retained only those words satisfying a certain threshold, and applied attribute selection measure like Information Gain which finds the best subset that maximizes the classification efficiency. Information of any feature or attribute is its measure of purity which helps in classifying a new instance based on this word alone and represents the amount of information a feature carries.

[7] followed two different kinds of evaluation schemes: the K-fold cross validation and the X:Y validation. In K-fold cross validation, the entire data is divided into K equal sets. One set is alternatively selected as the testing set and the remaining K-1 sets are combined to be the training set. The final efficiency of the model is the sum of K efficiencies divided by K. On the other hand, the X:Y validation splits the data into two not necessarily equal size wherein X% is used for training and Y% is used for testing such that X + Y is 100.

The raw data that was used in [3] came from large sets of movie reviews collected by research communities. They used the Cornell University movie review dataset which has 1000 positive and 1000 negative reviews on which's class is then decided by the frequency of each words appearing in the model obtained from the training dataset and the Stanford SNAP Amazon movie review dataset which is organized into eight lines for each review in their experiments. The reviews were simply divided into positive and negative by setting 3.0 as a threshold since it has a 5-point rating system.

They considered only unigrams for the model. They simply deleted unwanted context such as punctuations, special symbols, and numbers to preprocess the raw reviews from the dataset. They did not use a lexicon or vocabulary to filter out words

without meaning.

A simple and complete system for sentiment mining on large datasets was presented by [3] using a Naive Bayes classifier with the Hadoop framework to evaluate the scalability of Naive Bayes classifier in large-scale datasets. 80.85% average accuracy of ten trials was attained when they tested their code on Cornell dataset. It was also able to classify with comparable accuracy different subsets of Amazon movie review dataset. The size of the dataset in their experiment varies from one thousand to one million reviews in each class to test the scalability of the Naive Bayes classifier. The study shows that the accuracy is unstable when the dataset is relatively small because the training data are not big enough for the model to learn enough knowledge about the class but the accuracy gradually increase above 80% and approaching 82% as the dataset increase above 400K demonstrating that the accuracy of Naive Bayes classifier is stable when the dataset increases. True positive and negative increase as the size of the dataset increase while the false positive and negative decrease as expected.

In [4]'s preliminary experiments, they used the training dataset of tweets provided by SemEval2014 organization containing 6, 408 tweets. They also used a selection of 5, 050 positive and negative labeled tweets compiled from an external source.

According to two different strategies, they built two different Naive Bayes classifiers: baseline and binary. A baseline Naive Bayes classifier learns from the original training corpus wherein no modification has been introduced. It classifies three categories found in the corpus which are positive, negative, and neutral. On the other hand, a binary Naive Bayes classifier doesn't take neutral tweets into account. This basic binary or Boolean classifier was trained on a simplified training corpus considering only positive and negative tweets and identifies only both positive and negative tweets. It uses a polarity lexicon in order to detect neutral tweets or tweets without polarity. Basically, if the tweet contains at least one word that is found in the polarity

lexicon, then the tweet has some degree of polarity. Otherwise, if the tweet does not contain at least one lemma found in an external polarity lexicon, then it is classified as neutral and is considered having no polarity.

They searched for a polarity word (adjective, noun, or verb) within a window of 2 words after the negation whenever a negation word is found considering its parts-of-speech tag and its syntactic properties. If a syntactically linked polarity word and negative word is found, its polarity is reversed. For instance, the system only reverses the polarity of adjectives or verbs appearing to the right of an adverb "not" since nouns are not syntactically linked to this adverb. By contrast, only the polarity of nouns can be reversed by the negation determiner "no" or "none".

Given that the language of microblogging like tweets must be corrected and normalized before lemmatizing them, they proposed preprocessing tasks. They removed URLs, references to usernames, and hashtags. They also reduced replicated characters like "looooove" to "love" and replaced emoticons and interjections with polarity or sentiment expressions like ":-)" to "good". The features that the classifier considered are lemmas, multiwords, polarity lexicons, and valence shifters. Instead of tokens, they used unigrams of lemmas to minimize the problems due to sparse distribution of words. Only lemmas belonging to lexical categories such as nouns, verbs, adjectives, and adverbs are selected as features while grammatical words such as conjunctions, determiners, and prepositions were removed. They also used multiword expression identified by parts-of-speech tags patterns such as noun-adj, noun-noun, adj-noun, noun-prp-noun, verb-noun, verb-prp-noun. The bigrams and trigrams produced with these patterns are added to the unigrams in the model. They also built a polarity lexicon with 10, 850 positive and negative entries from different sources and used it to identify neutral tweets and to build artificial tweets wherein each entry of the lexicon is converted into artificial tweets, which will be taken into account for training classifiers, with just one lemma inheriting the polarity from the lexicon and the frequency

of the word in each new tweet is the average frequency of lemmas in the training corpus.

All the classifiers here have been implemented using Perl language. [4] showed that the performance of the classifier improved when it was implemented with the binary strategy using a polarity lexicon and when multiwords are selected as features.

Twitter users tend to use heavy abbreviations and fragmented expressions because every twitter update is restricted to 140 characters in length unlike the social networking site Facebook which has 5, 000 characters for every status update making clearer sentences construction more expectable. Because of this, [1] focused on users' Facebook status updates but did not include other Facebook posts like application, photo, and other similar stories. From 90 users, they collected around 7000 status updates. These statuses were not collected by using specific queries but by random sampling of streaming Facebook statuses. Figure 1 shows the sample of status updates in [1].

Sample of Negative Status Updates:	Sample of Positive Status Updates:
Freaking full of doubt.	Just finished making pancakes for breakfast.. oh and the yummiest part, it comes with a free strawberry syrup!
i really don't like to shave my hair but i have to. :/ frustrated :(	inspired by you! <3
can't sleep...	11 days to go before Christmas :)

Figure 1: Sample of Status Updates in [1]

They based the final distribution of the final dataset from the negative samples because they are fewer. The dataset was randomly selected for each partition. The following data distribution was used for training and testing set. Figure 2 shows data

distribution of status updates in [1].

	Training	Testing
Positive	1131	1131
Negative	1131	1131

Figure 2: Data Distribution in [1]

The main methodology for sentiment analysis in [1] is the Naive Bayes classifier method wherein a status update is classified whether positive or negative. Figure 3 shows the main methodology for sentiment analysis in [1].

The classifier was evaluated using the precision, recall, and F-score performance and showed the following results in Figure 4 and Figure 5.

Negation words may reverse the sentiment of opinion word sentence so [6] considered this feature for sentiment extraction from negative subjective sentences. They listed some negative prefixes which reverses the meaning of words when used. They are:

1. Dis- can be used with adjectives, adverbs, nouns, or verbs. Examples are disaffected, disbelief, disinfect.

2. In-, il-, im-, and ir- are usually used for adjectives, nouns, or the adverbs formed from them. Examples are imbalance, illegal, inaccurate, irregular.

3. De- is almost always used before a verb or a word from that verb which reverses the verb's action. Examples are deactivate, decrease, dehydrate.

4. Non- means not or lack of something. Examples are nonmetallic, nonstop, nonrestrictive.

5. Un- is the most common negative English prefix. Examples are unafraid, unable, uncertain.

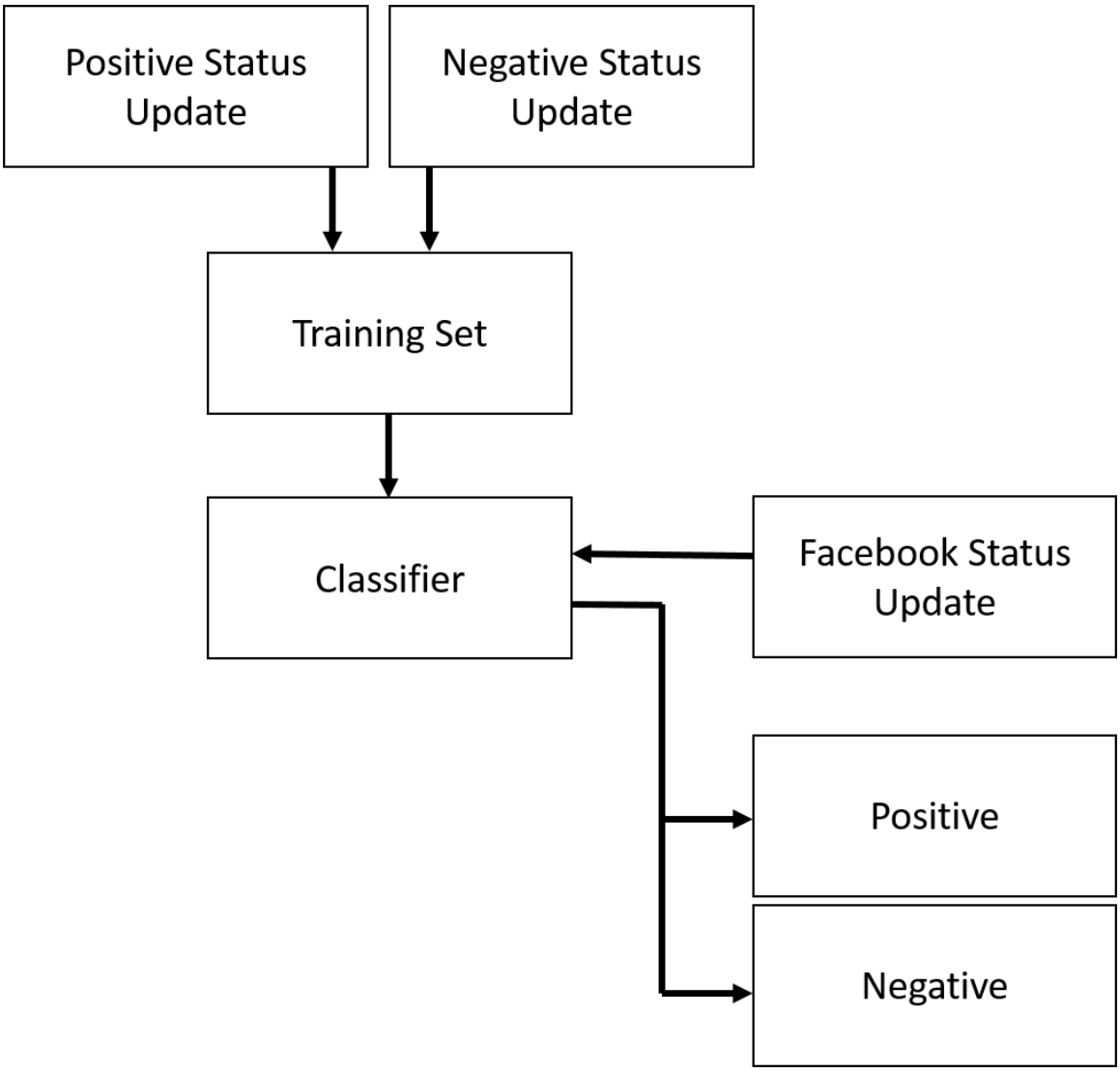

Figure 3: Main Methodology for Sentiment Analysis in [1]

Naïve Bayes Classifier	Actual Positive	Actual Negative
Predicted Positive	0.76	0.23
Predicted Negative	0.35	0.65

Figure 4: Naive Bayes Precision and Recall Performance in [1]

	Naïve Bayes Classifier
Precision	0.77
Recall	0.68
F-score	0.72

Figure 5: Precision, Recall, and F-score of the Classifier in [1]

6. Mis- is used with verbs as well as nouns and adverbs and adjectives made from verbs. Examples are misspell, misinterpret, mistake.

According to [5] supervised machine learning techniques performs relatively better than the unsupervised lexicon-based methods. An important decision to make in a supervised classification technique is feature selection which tells us how documents are represented. Listed below are the most commonly used features in sentiment analysis.

1. Term presence and frequency: In sentiment classification, these features have been widely and successfully used. These includes unigrams, bigrams, trigrams, or n-grams, their presence or frequency.

2. Part of speech information: Here, each term in sentences will be assigned a label or POS tag which represents its role or position in the grammatical context. In turn, it is used to guide feature selection because it can identify adjectives and adverbs which are used as sentiment indicators.

3. Negations: It is an important feature for it can reverse the sentiment of a sentence.

4. Opinion words and phrases: They express positive or negative sentiments using mainly statistical-based or lexicon-based approaches to identify its semantic orientation.

5. Syntactic dependency: Word dependency based features generated from dependency tree or parsing are used by several research works in this area.

The biggest limitation of supervised learning is that it is affected by the quantity and quality of the training data and may fail when it is biased or insufficient [2].

In [10], examples of feature vectors are shown. They looked at two probabilistic models of documents, both representing documents as a bag of words whose components correspond to word types using the Naive Bayes assumption. If given a vocabulary V, containing $|V|$ word types, then the feature vector dimension $d = |V|$.

Example: Consider the vocabulary

$$V = blue, red, dog, cat, biscuit, apple$$

In this example, $|V| = d = 6$. Consider the document the blue dog ate a blue biscuit. If d^B is the Bernoulli feature vector for this document, and d^M is the multinomial feature vector, then:

$$d^B = (1, 0, 1, 0, 1, 0)^T$$

$$d^M = (2, 0, 1, 0, 1, 0)^T$$

III. Theoretical Framework

A. Sentiment Analysis

Sentiment analysis or opinion mining is an extended field of information retrieval or data mining [6]. It is the automated mining of attitudes, emotions, and opinions from database resources, speech, and text through Natural Language Processing [5]. It is a technique in extracting and detecting subjective information and finding the opinion in text documents to determine the contextual polarity (e.g. positive, negative, or neutral) of a document with respect to certain objects [2] [8] [4]. Figure 6 shows the techniques in sentiment classification according to [6].

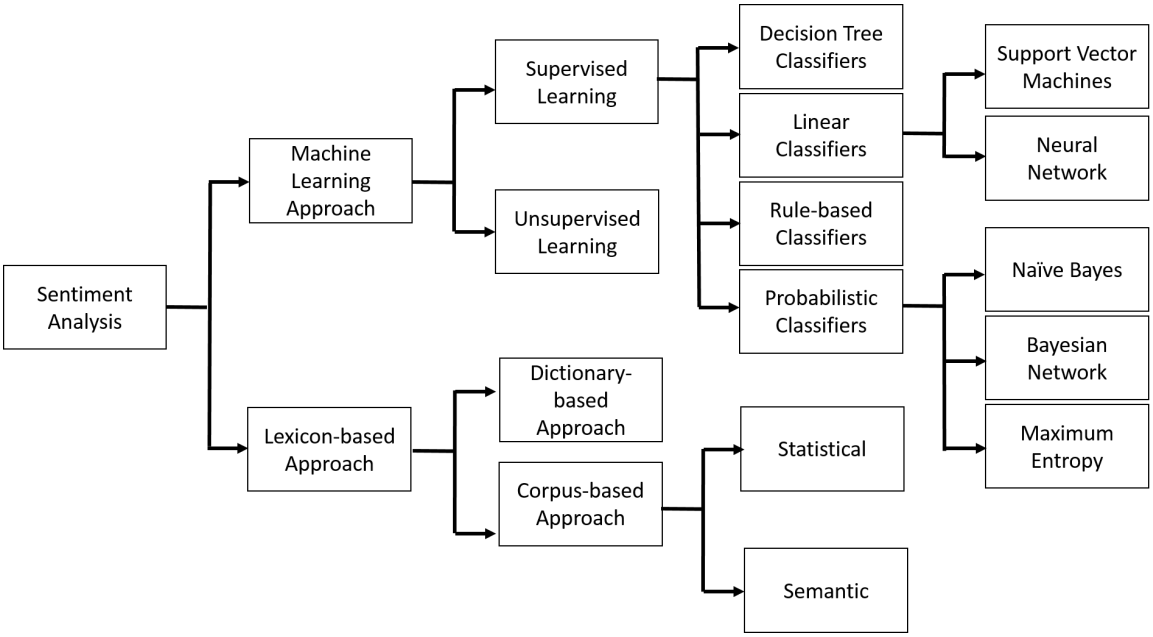

Figure 6: Techniques in Sentiment Classification

B. Supervised Learning/Approach

In a supervised approach, a classifier or a model is trained on a series of text documents or reviews that are categorized manually beforehand [7] requiring pre-classified examples to train on [1]. Two sets of data are required in a machine learning based

approach: a training and a test set. To learn the differentiating characteristics of documents, the training set is used by an automatic classifier while the test set is used to validate and check the how well the performance of the classifier is [6] [9].

C. Text Classification

It is the task of classifying documents by the words which they are comprised or by their content [10].

D. Naive Bayes Classifier

Naive Bayes is an intuitive method in classifying that combines efficiency with reasonable accuracy [4]. This classification algorithm is a very simple probabilistic model yet effective and works well on text classifications [9] [8].

The basic idea of this method is to use the joint probabilities of words and categories in estimating the probabilities of categories given a document [7]. The Bayes rule is the basis of Bayesian classifiers. It is a way of looking at conditional probabilities allowing us to flip the condition around which can be very helpful when we are estimating the probability of something based on examples of it occurring [1].

The large assumption that we make rendering this process as naive is calculating the probability of the document occurring by getting the product of the probabilities of each word within its occurrence [1]. The conditional probability of a word given a category is assumed to be independent from the conditional probabilities of other words given that category called word independence assumption [7].

The maximum likelihood probability of a word belonging to a particular class is given by the equation:

$$P(x_i|c) = \frac{\text{Count of } x_i \text{ in documents of class } c}{\text{Total number of words in documents of class } c}$$

The probability of a particular document belonging to a class c_i according to the Bayes Rule is given by:

$$P(c_i|d) = \frac{P(d|c_i) * P(c_i)}{P(d)}$$

Using the conditional independence assumption, the model is termed as naive [8].

$$P(c_i|d) = \frac{(\Pi P(x_i|c_j)) * P(c_j)}{P(d)}$$

E. Independence Assumption

Among the linguistic features, independence assumption assumes conditional independence. That is, given a positive or negative class, the words are conditionally independent of each other implying that there is no link between one word or another word [4] [8] [1].

F. Bernoulli Naive Bayes

A Bernoulli Naive Bayes is a type of a Naive Bayes algorithm that includes just the presence of a word instead of its number of occurrence removing duplicate words from the document which don't add any additional information [8].

G. Bernoulli Document Model

In a Bernoulli document model, a document is represented by a feature vector with binary elements 1 or 0. If the corresponding word is present in the document, it takes the value 1 and 0 if the word is not present [10].

H. Feature Extraction/Selection

An important step in a supervised classification technique [6] that removes redundant features and includes only those features that have high disambiguation capabilities [8] which's goal is to improve computational efficiency, classification effectiveness, or both [7].

I. Preprocessing Techniques

The following preprocessing techniques are done in each method:

1. Text normalization
    (a) Removal of hashtags, usernames, and URLs from [11]
    (b) Conversion of all letters to lowercase
    (c) Removal of special characters
    (d) Removal of repeating characters
2. Negation handling
    (a) Conversion of "not" + word to "not_word"
    (b) Conversion of words with negative prefix from [12] to "not_word"
3. Noise words elimination
    (a) Removal of prepositions, conjunctions, and determiners
4. Emoticon conversion
    (a) Conversion of positive emoticons from [13] to "positive_emoticon"
    (b) Conversion of negative emoticons from [13] to "negative_emoticon"
5. Inflated or derived words reduction
    (a) Reduction of inflated or derived words into its base form

J. Laplace Smoothing

In [8], the Laplacian smoothing was used to solve the problem of a classifier encountering a word that has not been seen in the training set that would cause the probability of both classes to become zero and there wouldn't be anything to compare. The Laplacian smoothing that was used was

$$P(x_i|c_j) = \frac{Count(x_i) + k}{(k+1) * (\text{Number of words in class } c_j\ )}$$

where k is usually chosen as 1. This way, the probability of the new word being in either class is equal.

K. Evaluation Measures

Generally, the performance of sentiment classification is evaluated by accuracy, precision, recall, and F1-score. The confusion matrix is a common way of computing these four indexes [2]. Figure 6 shows the confusion matrix.

	Predicted positives	**Predicted negatives**
Actual positive instances	# of True Positive instances (*TP*)	# of False Negative instances (*FN*)
Actual negative instances	# of False Positive Instances (*FP*)	# of True Negative instances (*TN*)

Figure 7: Confusion Matrix

These indexes can be defined by the following equations:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

The portion of all true predicted instances against all predicted instances is called the accuracy. When the predicted instances are exactly the same as the actual instances, an accuracy of 100% is achieved. The portion of true positive predicted instances against all positive predicted instances is the precision and the portion of true positive predicted instances against all actual positive instances is the recall. The harmonic average of precision and recall is the F1.

IV. Design and Implementation

Figure 8 shows the flow chart for training the model.

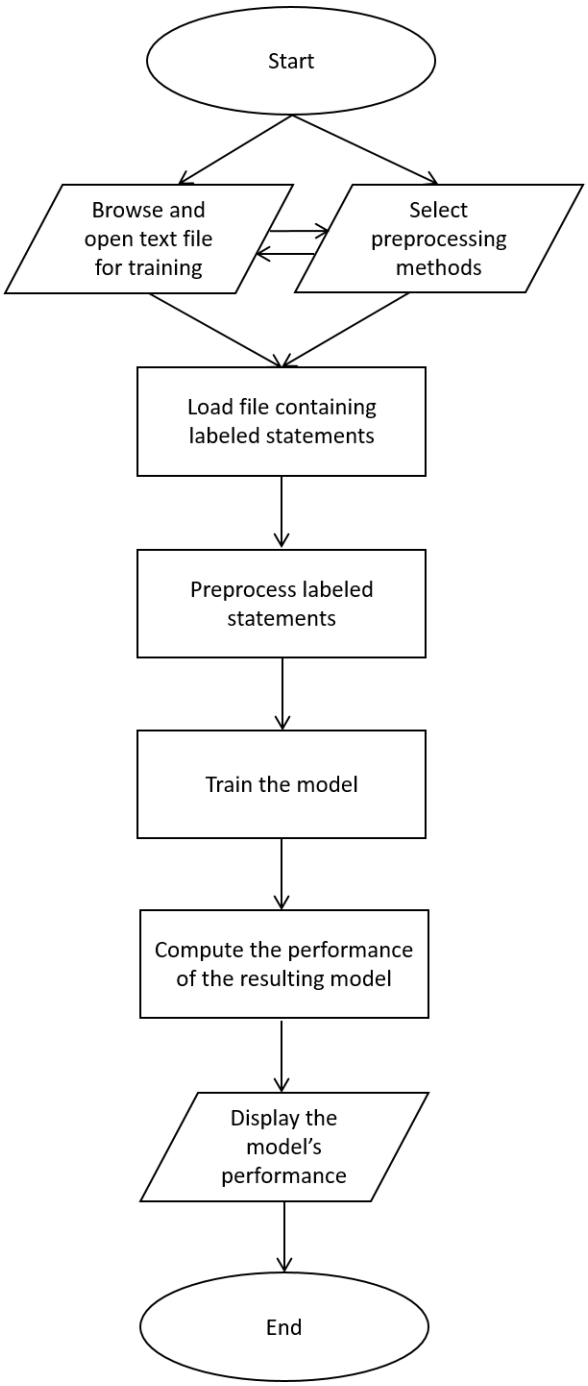

Figure 8: Flow Chart for Training the Model

Figure 9 shows the flow chart for classifying the input using the trained model.

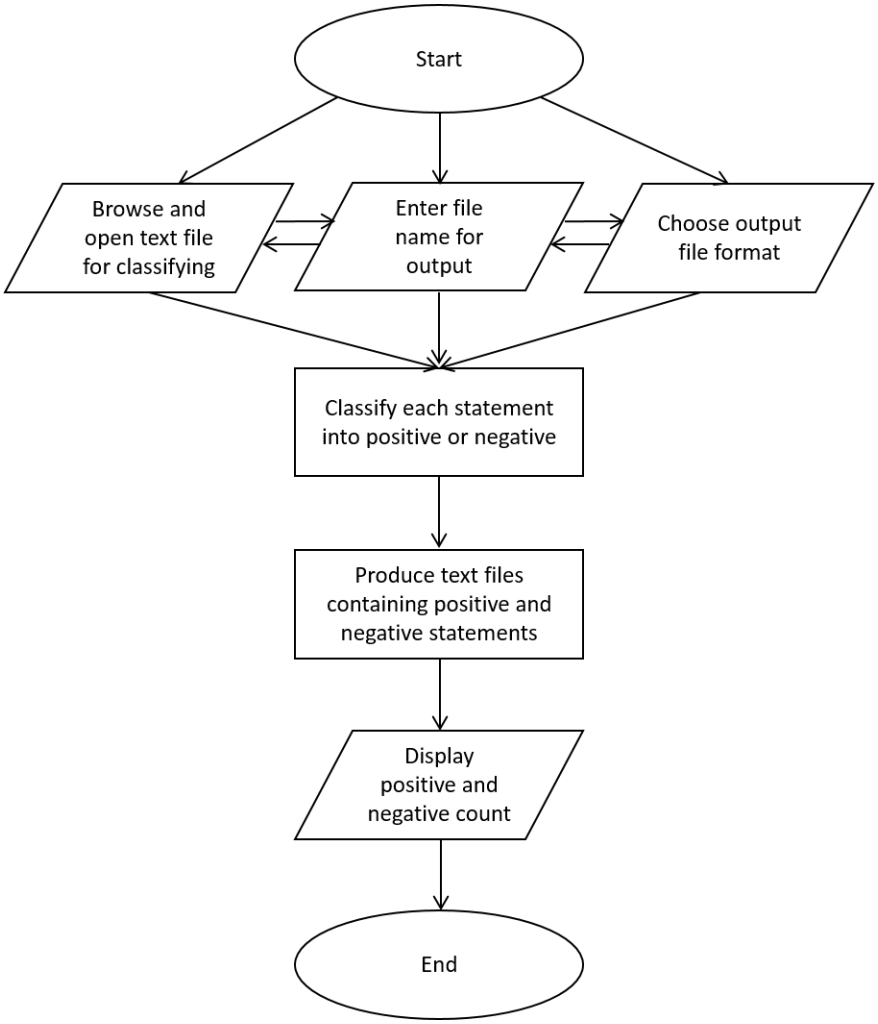

Figure 9: Flow Chart for Classifying the Input Using the Trained Model

A. Use Cases

Figure 10 shows the use case diagram for the system.

B. System Architecture

The tool was created using Java. No library was used for the implementation of the classifier while the interface was created using javax.swing which provides components in creating a GUI.

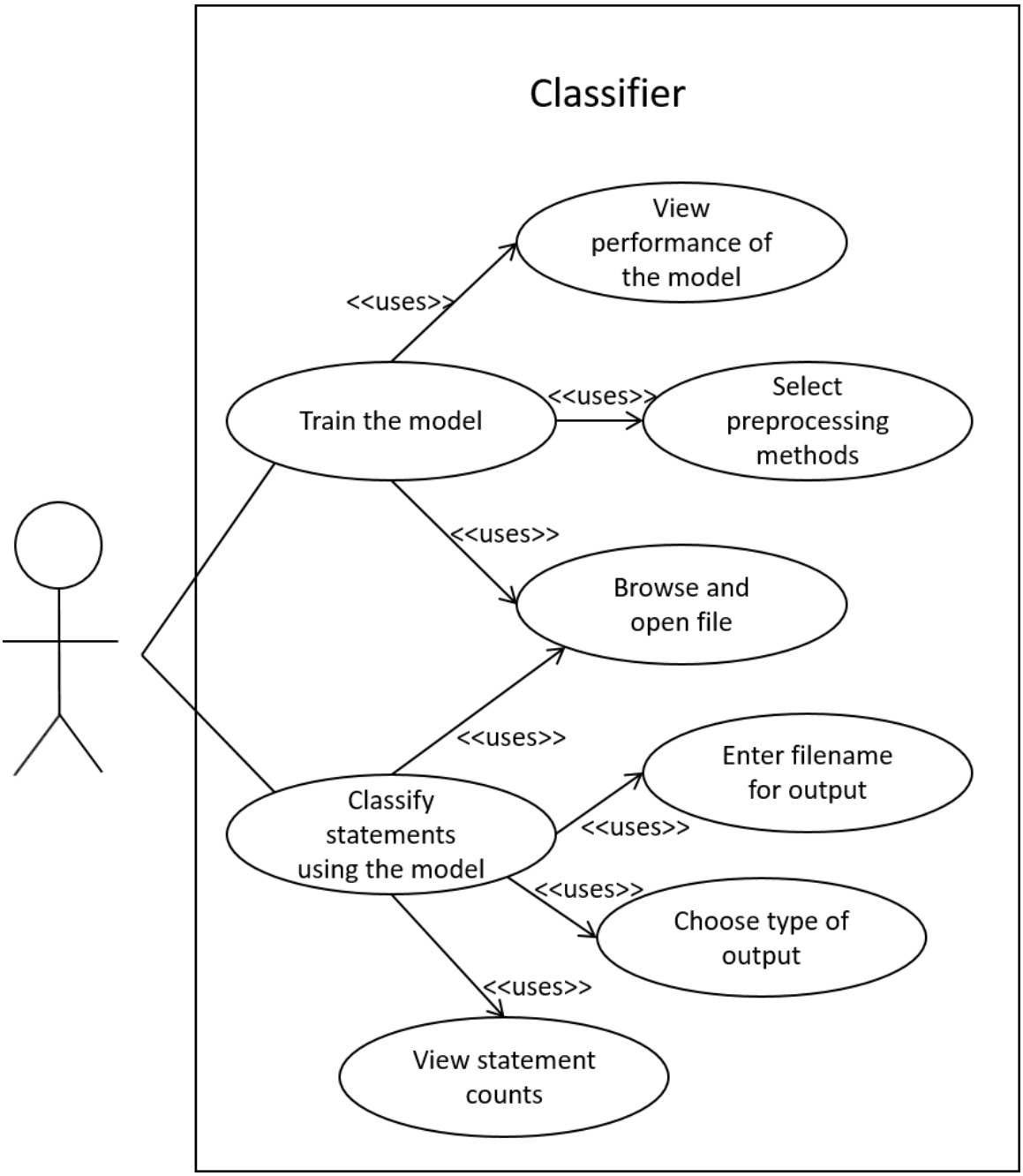

Figure 10: Use Case Diagram for the System

C. Technical Architecture

The tool is a stand-alone program run in Java and only requires local memory.

Minimum System Requirements:

1. A mid-range CPU with with at least 1.70 GHz

2. At least 2.00GB RAM

3. A 64-bit operating system

4. Windows 7/8/10

5. Java runtime environment

V. Results

The Customizable Naive Bayes Classifier is a classifier used for Sentiment Analysis. The user will be the one to train the model by loading data and selecting preprocessing methods.

When opening the tool, a splash screen shown in figure 11 will appear.

When using the tool, figure 12 will be your main working environment.

A. Training the Model

Before the user can use the tool to classify inputs as positive or negative, he must train it first by choosing an input file contaning already classified statements in format "*statement*\t*polarity*\n" with polarity equal to 1 if positive and 0 if negative. The tool only accepts text files in this format. They can be easily found online or the user may provide his own dataset as he wishes.

The user can easily choose a file by clicking the "browse" button. The tool will show a file explorer window as shown in figure 13. Once the file is selected, the filepath will be displayed in figure 14. Not choosing an input file for training will pop up an error message (see figure 15).

In figure 16, the user selects the preprocessing methods he wishes to perform to the statements before the model uses them for learning.

By clicking the "Train the Model" button, the tool will show an indicator in 17 and display the performance of the model built from the input file and the preprocessing methods selected. See figure 18.

The user can train the model using different input files and different combinations of preprocessing methods until he is satisfied with its performance.

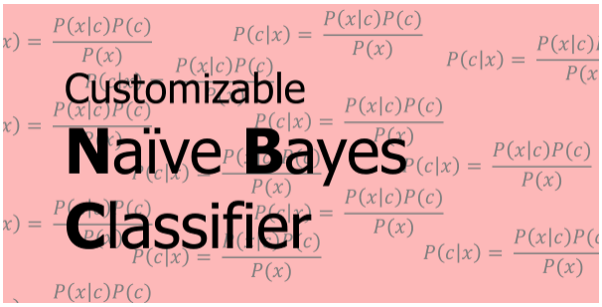

Figure 11: Splash Screen When Opening the Tool

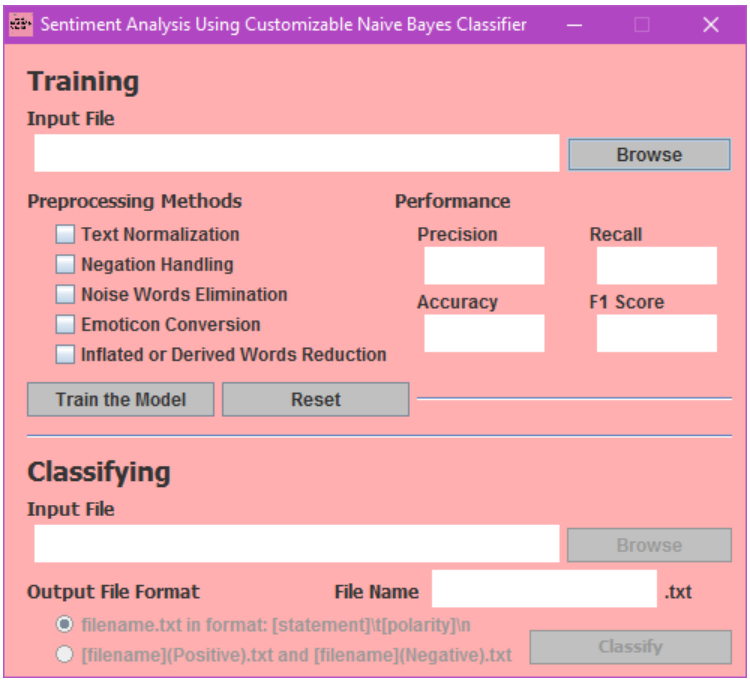

Figure 12: Main Window of the System

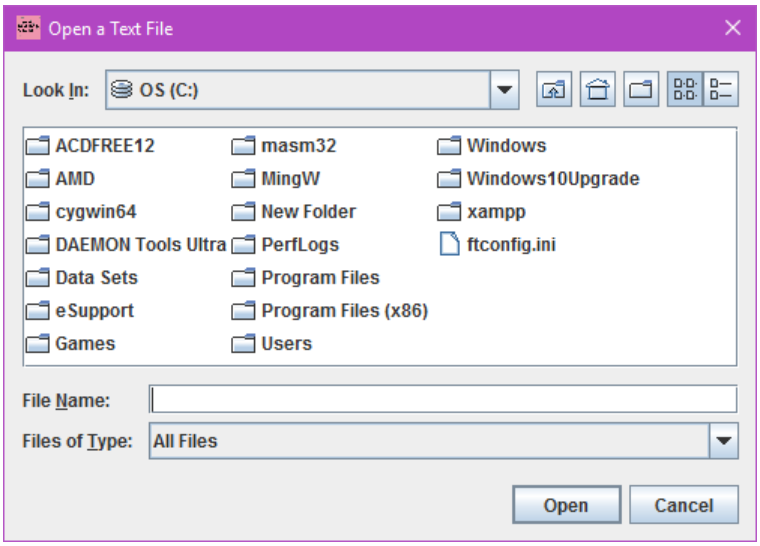

Figure 13: Browsing for a file

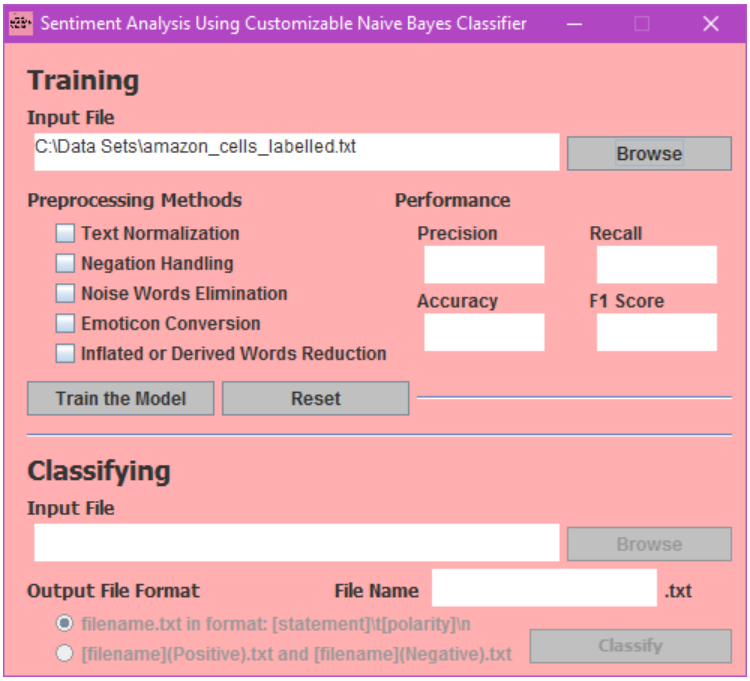

Figure 14: Choosing an input file for training

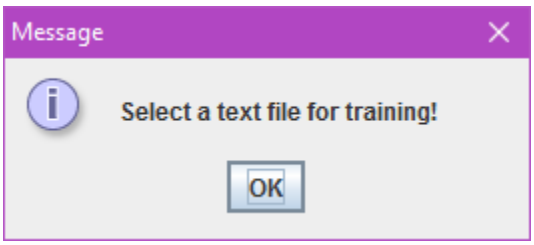

Figure 15: Error message when you didn't choose an input file for training

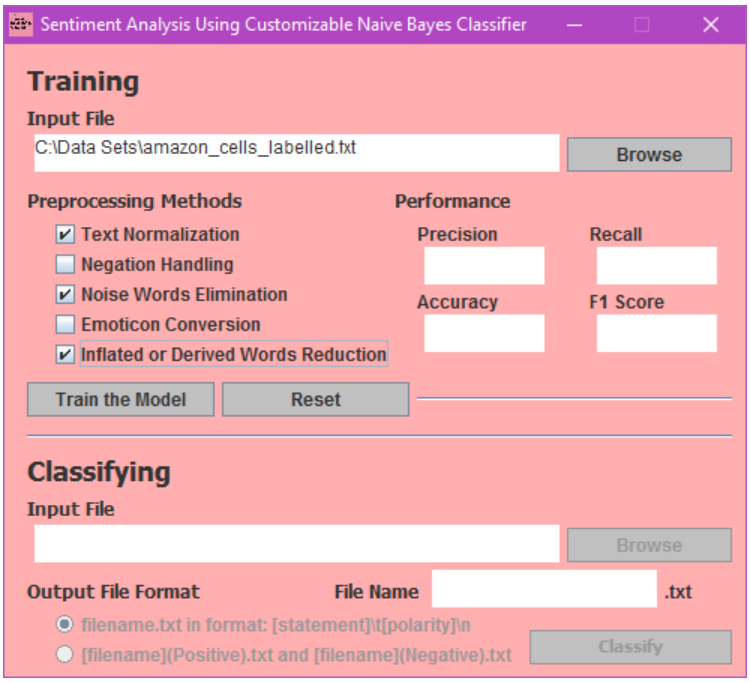

Figure 16: Selecting of preprocessing methods

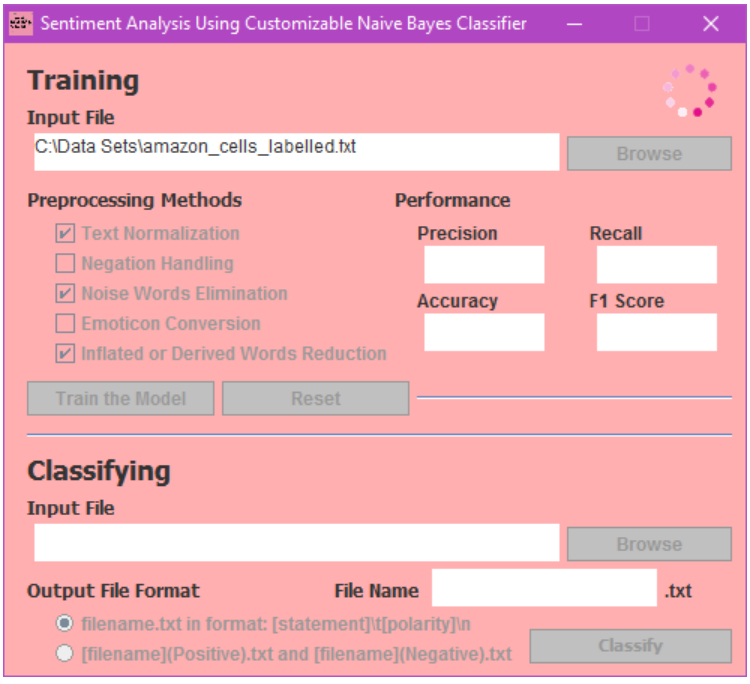

Figure 17: Training the model

B. Classifying Using the Model

Once the training phase is done, the tool can now be used to classify statements into positive or negative.

The user starts by clicking the "browse" button beside the "Input File" text area in figure 19 and opening the input file containing the statements to be classified. The user must also enter the file name for the output file/s shown in 20 that will be produced with the format selected by the user in figure 21 or 22. Failing to choose an input file or enter a file name for the output file will bring you errors shown in figure 23 and 24, respectively.

When the "Classify" button is clicked, the model will start classifying and an indicator will be shown (see 25)the input and will write output text file/s in the same directory where the input file for classification came from as shown in 26. A pop-up message in figure 27 will show the number of positive and negative statements classified.

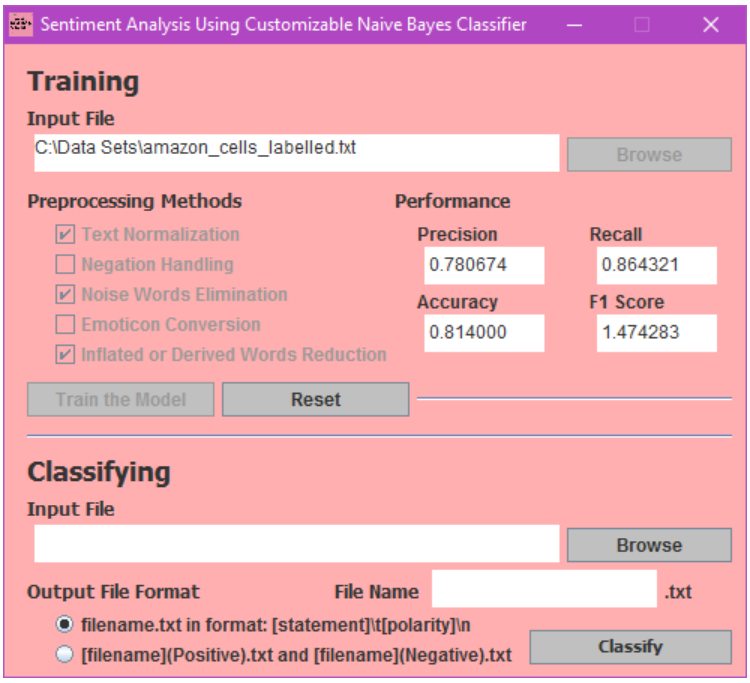

Figure 18: Displaying the model's performance

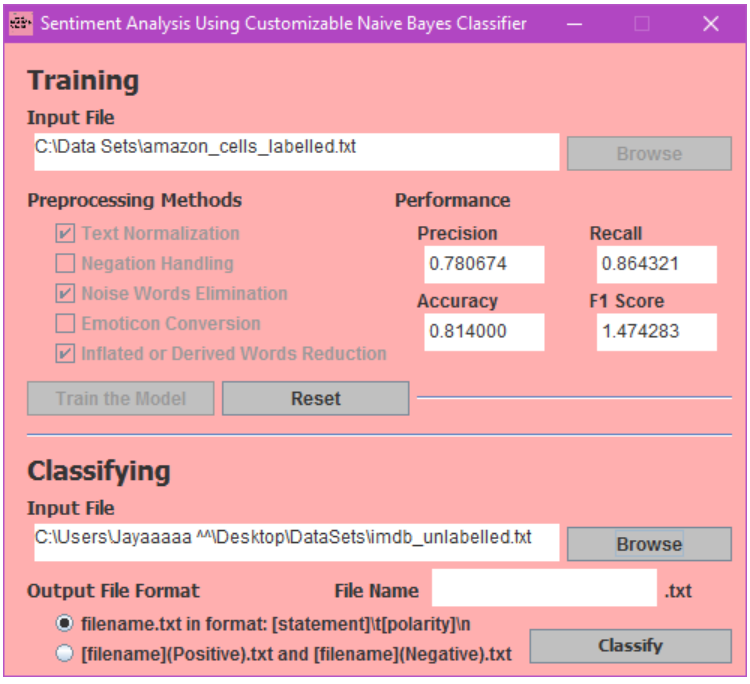

Figure 19: Choosing an input file to be classified

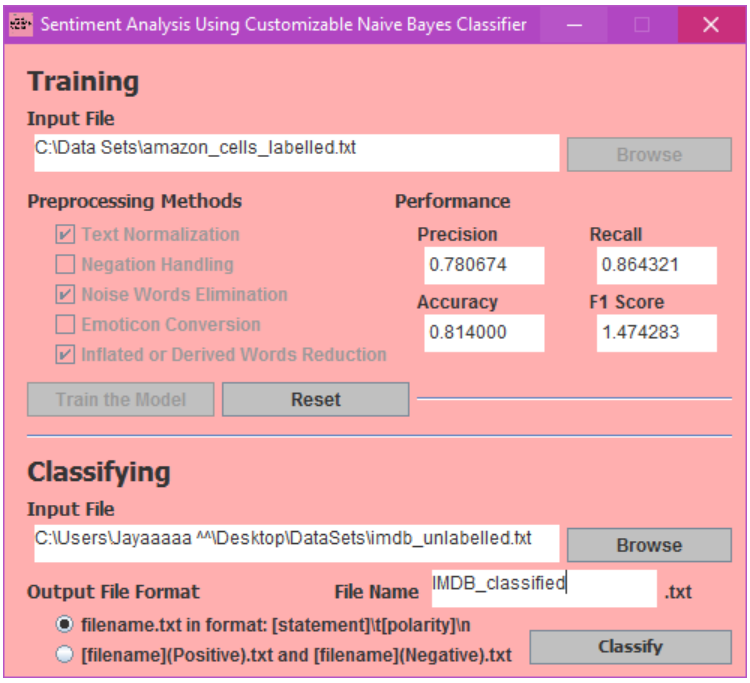

Figure 20: Entering a file name for the output file/s

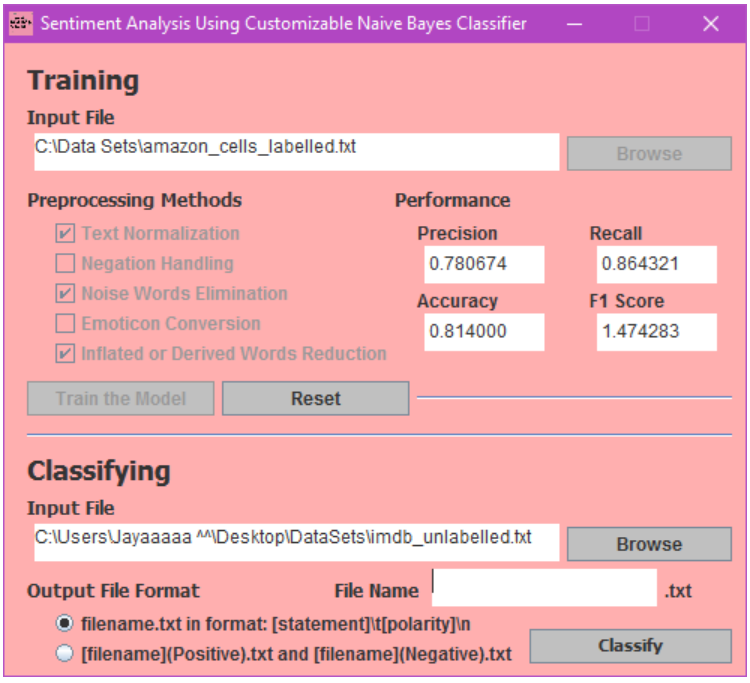

Figure 21: Choosing one-output-file format

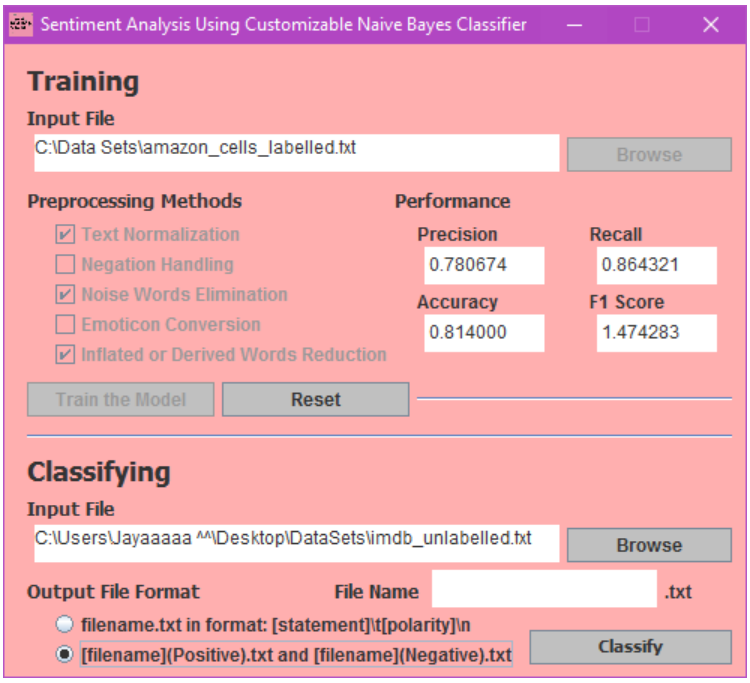

Figure 22: Choosing two-output-file format

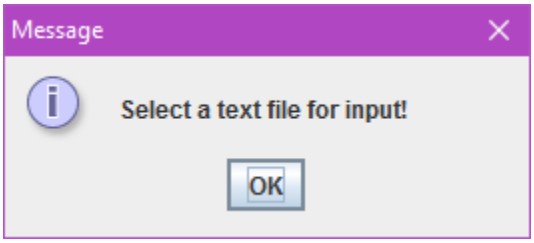

Figure 23: Error message when you didn't choose an input file to be classified

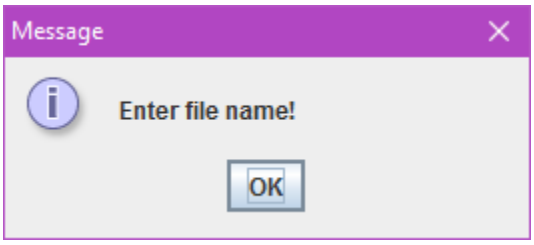

Figure 24: Error message when you didn't enter a file name for output file/s

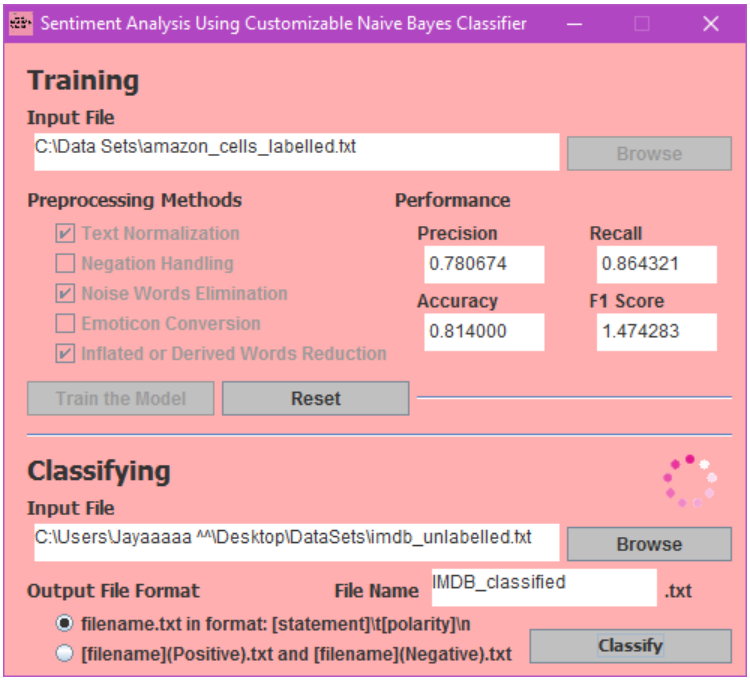

Figure 25: Classifying the Model

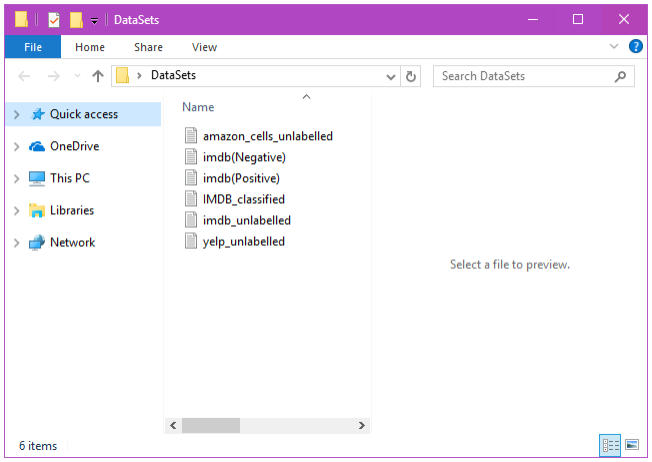

Figure 26: The location of the produced outputs

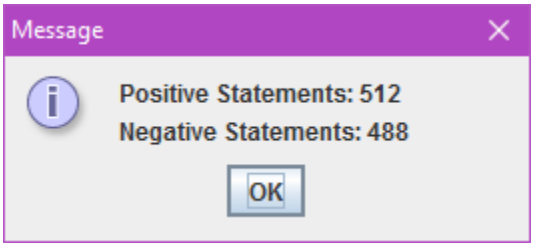

Figure 27: Dialog box showing the number of positive and negative statements classified

VI. Discussions

In figure 28, the resulting evaluation measures when different combinations of preprocessing methods are used to labeled statements from amazon_cells_labelled.txt in [14] are presented.

The highest accuracy with a score of 0.824 was achieved twice. First when Text Normalization, Emoticon Conversion, and Inflated or Derived Words Reduction preprocessing methods were used. Second was when Text Normalization was used along with Inflated or Derived Words Reduction. This means that among all statements classified, 82.4% were classified correctly while the remaining 17.6% was wrongly classified.

The earlier mentioned preprocessing methods also produced the highest precision score, 0.794708. This implies that 79.47% of all statements classified as positive are true positive while the others are false positive.

Same preprocessing methods are used when the highest F1-score resulted. Meaning that the highest harmonic average between recall and precision achieved is 1.47791.

Three of the highest evaluation measures were achieved using the same preprocessing methods except for recall. 0.87665 was the highest recall score which resulted when combination of Text Normalization, Negation Handling, Emoticon Conversion, and Inflated or Derived Words Reduction were applied and another when Text Normalization, Negation Handling, and Inflated or Derived Words Reduction were used.

As observed on figure 28, when the only difference from the set of preprocessing methods used is Emoticon Conversion, the same performance is achieved. This implies that Emoticon Conversion isn't useful with this dataset.

In figure 29, another dataset used for training is from imdb_labelled.txt in [14]. The set of Text Normalization, Negation Handling, Emoticon Conversion, and Inflated or Derived Words Reduction and set of Text Normalization, Negation Handling, and Inflated or Derived Words Reduction produced the highest accuracy and

recall scores obtaining 0.82 and 0.751364, respectively.

The highest precision was achieved when Noise Words Elimination was used instead of Inflated or Derived Words Reduction. 87.28% of all classified positive statements are true positive.

When Negation Handling and Emoticon Conversion is applied, the same highest F1-score, 1.555432, is computed compared with using Negation Handling alone.

As shown in the figure, just like with the previous dataset, it seems that Emoticon Conversion doesn't cause a difference to the outcomes.

The results gathered using labeled statements from yelp_labelled.txt in [14] are presented in figure 30. Here, four combinations of preprocessing techniques achieved the highest accuracy score of 0.804. These combinations are (i) Text Normalization, Negation Handling, Noise Words Elimination, Emoticon Conversion, and Inflated or Derived Words Reduction, (ii) Text Normalization, Negation Handling, Noise Words Elimination, and Emoticon Conversion, (iii) Text Normalization, Negation Handling, Noise Words Elimination, and Inflated or Derived Words Reduction, (iv) Text Normalization, Negation Handling, and Noise Words Elimination.

The highest precision of 0.785556 is obtained when Text Normalization and Noise Words Elimination. Adding Emoticon Conversion to the set gives you the same precision score.

An F1-score of 1.492039 is computed when using Text Normalization and Emoticon Conversion. Same F1-score is computed when only Text Normalization is applied.

Based from the results, it seems that Emoticon Conversion is the least effective preprocessing method. Selecting it or not doesn't affect the outcome in these particular datasets. On the otherhand, Text Normalization is present in almost all sets of preprocessing methods that produced the highest scores implying that Text Normalization is a good preprocessing method.

TN	NH	NWE	EC	IDWR	Precision	Accuracy	Recall	F1-Score
✓	✓	✓	✓	✓	0.782640	0.819000	0.875861	1.471489
✓	✓	✓	✓		0.785111	0.818000	0.867614	1.474905
✓	✓	✓		✓	0.782640	0.819000	0.875861	1.471489
✓	✓	✓			0.785111	0.818000	0.867614	1.474905
✓	✓		✓	✓	0.773460	0.813000	0.876650	1.468388
✓	✓		✓		0.780177	0.818000	0.876482	1.470692
✓	✓			✓	0.773460	0.813000	0.876650	1.468388
✓	✓				0.780177	0.818000	0.876482	1.470692
✓		✓	✓	✓	0.780674	0.814000	0.864321	1.474283
✓		✓	✓		0.773831	0.808000	0.861101	1.472991
✓		✓		✓	0.780674	0.814000	0.864321	1.474283
✓		✓			0.773831	0.808000	0.861101	1.472991
✓			✓	✓	0.794708	0.824000	0.867517	1.477910
✓			✓		0.787224	0.818000	0.863351	1.476677
✓				✓	0.794708	0.824000	0.867517	1.477910
✓					0.787224	0.818000	0.863351	1.476677
	✓	✓	✓	✓	0.714249	0.752000	0.836822	1.460644
	✓	✓	✓		0.709097	0.744000	0.825203	1.462054
	✓	✓		✓	0.714249	0.752000	0.836822	1.460644
	✓	✓			0.709097	0.744000	0.825203	1.462054
	✓		✓	✓	0.715126	0.756000	0.852967	1.455794
	✓		✓		0.712878	0.753000	0.848443	1.456334
	✓			✓	0.715126	0.756000	0.852967	1.455794
	✓				0.712878	0.753000	0.848443	1.456334
		✓	✓	✓	0.719290	0.759000	0.845709	1.459528
		✓	✓		0.721179	0.759000	0.841813	1.461187
		✓		✓	0.719290	0.759000	0.845709	1.459528
		✓			0.721179	0.759000	0.841813	1.461187
			✓	✓	0.725808	0.763000	0.845011	1.461850
			✓		0.725766	0.764000	0.846220	1.461313
				✓	0.725808	0.763000	0.845011	1.461850
					0.725766	0.764000	0.846220	1.461313

Figure 28: Performance values of the model using different sets of preprocessing methods with amazon_cells_labelled.txt as input

TN	NH	NWE	EC	IDWR	Precision	Accuracy	Recall	F1-Score
✓	✓	✓	✓	✓	0.866203	0.817000	0.746344	1.537372
✓	✓	✓	✓		0.872755	0.819000	0.742392	1.540766
✓	✓	✓		✓	0.866203	0.817000	0.746344	1.537372
✓	✓	✓			0.872755	0.819000	0.742392	1.540766
✓	✓		✓	✓	0.869644	0.820000	0.751364	1.536614
✓	✓		✓		0.871325	0.816000	0.739934	1.540922
✓	✓			✓	0.869644	0.820000	0.751364	1.536614
✓	✓				0.871325	0.816000	0.739934	1.540922
✓		✓	✓	✓	0.865384	0.813000	0.741742	1.538718
✓		✓	✓		0.859056	0.807000	0.733338	1.539999
✓		✓		✓	0.865384	0.813000	0.741742	1.538718
✓		✓			0.859056	0.807000	0.733338	1.539999
✓			✓	✓	0.858594	0.807000	0.736271	1.538180
✓			✓		0.869203	0.805000	0.718199	1.547603
✓				✓	0.858594	0.807000	0.736271	1.538180
✓					0.869203	0.805000	0.718199	1.547603
	✓	✓	✓	✓	0.823882	0.770000	0.681360	1.547824
	✓	✓	✓		0.811518	0.760000	0.670614	1.547905
	✓	✓		✓	0.823882	0.770000	0.681360	1.547824
	✓	✓			0.811518	0.760000	0.670614	1.547905
	✓		✓	✓	0.816006	0.763000	0.676119	1.547097
	✓		✓		0.822167	0.759000	0.657878	1.555432
	✓			✓	0.816006	0.763000	0.676119	1.547097
	✓				0.822167	0.759000	0.657878	1.555432
		✓	✓	✓	0.825950	0.769000	0.678599	1.548799
		✓	✓		0.820227	0.761000	0.664495	1.552615
		✓		✓	0.825950	0.769000	0.678599	1.548799
		✓			0.820227	0.761000	0.664495	1.552615
			✓	✓	0.828285	0.766000	0.671125	1.552462
			✓		0.818918	0.758000	0.662970	1.552387
				✓	0.828285	0.766000	0.671125	1.552462
					0.818918	0.758000	0.662970	1.552387

Figure 29: Performance values of the model using different sets of preprocessing methods with imdb_labelled.txt as input

TN	NH	NWE	EC	IDWR	Precision	Accuracy	Recall	F1-Score
✓	✓	✓	✓	✓	0.782944	0.804000	0.835153	1.483408
✓	✓	✓	✓		0.785161	0.804000	0.833265	1.484654
✓	✓	✓		✓	0.782944	0.804000	0.835153	1.483408
✓	✓	✓			0.785161	0.804000	0.833265	1.484654
✓	✓		✓	✓	0.770857	0.790000	0.822000	1.483577
✓	✓		✓		0.773915	0.794000	0.829442	1.482278
✓	✓			✓	0.770857	0.790000	0.822000	1.483577
✓	✓				0.773915	0.794000	0.829442	1.482278
✓		✓	✓	✓	0.780643	0.799000	0.826598	1.485323
✓		✓	✓		0.785556	0.797000	0.813249	1.490833
✓		✓		✓	0.779248	0.798000	0.826598	1.484840
✓		✓			0.785556	0.797000	0.813249	1.490833
✓			✓	✓	0.776780	0.789000	0.806995	1.490296
✓			✓		0.781472	0.792000	0.805740	1.492039
✓				✓	0.776780	0.789000	0.806995	1.490296
✓					0.781472	0.792000	0.805740	1.492039
	✓	✓	✓	✓	0.718105	0.750000	0.813476	1.468900
	✓	✓	✓		0.718775	0.751000	0.814215	1.469038
	✓	✓		✓	0.718105	0.750000	0.813476	1.468900
	✓	✓			0.718775	0.751000	0.814215	1.469038
	✓		✓	✓	0.716920	0.751000	0.816608	1.467597
	✓		✓		0.710292	0.746000	0.820626	1.464209
	✓			✓	0.716920	0.751000	0.816608	1.467597
	✓				0.710292	0.746000	0.820626	1.464209
		✓	✓	✓	0.723837	0.748000	0.795622	1.475999
		✓	✓		0.721042	0.748000	0.798570	1.474411
		✓		✓	0.723837	0.748000	0.795622	1.475999
		✓			0.721042	0.748000	0.798570	1.474411
			✓	✓	0.726737	0.753000	0.799178	1.476341
			✓		0.712727	0.741000	0.796753	1.472444
				✓	0.726737	0.753000	0.799178	1.476341
					0.712727	0.741000	0.796753	1.472444

Figure 30: Performance values of the model using different sets of preprocessing methods with yelp_labelled.txt as input

VII. Conclusions

The tool lets the user create his own Naive Bayes Classifier by feeding learning inputs and choosing preprocessing methods from

1. Text Normalization
2. Negation Handling
3. Noise Words Elimination
4. Emoticon Conversion
5. Inflated or Derived Words Reduction.

The tool displays the generated model's performance through 10-fold validation by showing computed

1. Precision
2. Accuracy
3. Recall
4. F1-score

to let the user decide whether the classifier is good enough or not effective to use. The user can improve the model's performance by changing learning inputs and choosing different combinations of preprocessing methods available.

It is a tool useful in classifying automatically the sentiment of a statement much faster than manually analyzing them with a very user-friendly user interface.

VIII. Recommendations

This tool can be improved by adding more preprocessing methods for feature extraction. A tool that is less customizable than this can also be made by just producing the model that has the best performance among all others when using different combinations of preprocessing methods.

Another way to enhance this tool is by using a pre-trained or default model so that users who don't have a dataset for training can still use this tool. Adding a "save" function will also be useful for the user to reuse previously generated models.

Since this is a Bernoulli Naive Bayes classifier, future works can be made using Multinomial Naive Bayes classifier or any other classifiers.

A tool with a different validation method is also recommended.

This tool only accepts text files as input and output. It is also good to develop a tool that is able to read and write a variety of file types.

IX. Bibliography

[1] C. Troussas, M. Virvou, K. J. Espinosa, K. Llaguno, and J. Caro, "Sentiment analysis of facebook statuses using naive bayes classifier for language learning," in *Information, Intelligence, Systems and Applications (IISA), 2013 Fourth International Conference on*, pp. 1–6, IEEE, 2013.

[2] M. Sadegh, R. Ibrahim, and Z. A. Othman, "Opinion mining and sentiment analysis: A survey," *International Journal of Computers & Technology*, vol. 2, no. 3, pp. 171–178, 2012.

[3] B. Liu, E. Blasch, Y. Chen, D. Shen, and G. Chen, "Scalable sentiment classification for big data analysis using naive bayes classifier," in *Big Data, 2013 IEEE International Conference on*, pp. 99–104, IEEE, 2013.

[4] P. Gamallo and M. Garcia, "Citius: A naive-bayes strategy for sentiment analysis on english tweets," *Proceedings of SemEval*, pp. 171–175, 2014.

[5] S. Vohra and J. Teraiya, "A comparative study of sentiment analysis techniques," *Journal JIKRCE*, vol. 2, no. 2, pp. 313–317, 2013.

[6] S. K. Singh and S. Paul, "Sentiment analysis of social issues and sentiment score calculation of negative prefixes," *International Journal of Applied Engineering Research*, vol. 10, no. 55, p. 2015.

[7] S. R. V. Reddy, D. V. L. N. Somayjulu, and A. R. Dani, "Classification of movie reviews using complemented naive bayesian classifier," *International Journal of Intelligent Computing Research*, vol. 1, p. 162167, Dec 2010.

[8] V. Narayanan, I. Arora, and A. Bhatia, "Fast and accurate sentiment classification using an enhanced naive bayes model," in *International Conference on*

Intelligent Data Engineering and Automated Learning, pp. 194–201, Springer, 2013.

[9] G. Vinodhini and R. Chandrasekaran, "Sentiment analysis and opinion mining: a survey," *International Journal*, vol. 2, no. 6, 2012.

[10] H. Shimodaira, *Text Classification using Naive Bayes*, p. 19.

[11] "List of internet top-level domains," May 2018.

[12] "Negative prefix list: Examples from de- and dis- to un-."

[13] "List of emoticons," Apr 2018.

[14] D. Kotzias, M. Denil, N. de Freitas, and P. Smyth, "From group to individual labels using deep features," 08 2015.

X. Appendix

A. Source Code

```
#include <iostream>

using namespace std;

int main{
        cout << "Hello world!" << endl;
        return 0;
}
```

```
import java.util.ArrayList;
import java.util.StringTokenizer;

public class BagOfWords {

        private ArrayList<String> words;

        public BagOfWords(Documents documents) {
                words = new ArrayList<String>();

                for (int i = 0; i < documents.size(); i++) {
                        StringTokenizer tokens = new StringTokenizer(documents.get(i), " ");

                        while (tokens.hasMoreTokens()){
                                words.add(tokens.nextToken());
                        }
                }
        }

        public BagOfWords(String document) {
                words = new ArrayList<String>();
                StringTokenizer tokens = new StringTokenizer(document, " ");

                while (tokens.hasMoreTokens()){
                        words.add(tokens.nextToken());
                }
        }

        public int getWordCount(String word) {
                int count = 0;

                for (int i = 0; i < words.size(); i++) {
                        if (word.equals(words.get(i))) {
                                count++;
                        }
                }

                return count;
        }

        public void removeRepeatingWords() {
                for (int i = 0; i < words.size(); i++) {
                        for (int h = 0; h < words.size(); h++) {
                                if (i != h && words.get(i).equals(words.get(h))) {
                                        words.remove(h);
                                        h--;
                                }
                        }
                }
        }

        public void deleteBlankWords() {
                for (int i = 0; i < words.size(); i++) {
                        if (words.get(i).length() == 0) {
                                words.remove(i);
                                i--;
                        }
                }
        }

        public void add(String word) {
                words.add(word);
        }

        public void add(int index, String word) {
                words.add(index, word);
        }

        public int size() {
                return words.size();
```

```
        }

        public String get(int index) {
                return words.get(index);
        }

        public void remove(int index) {
                words.remove(index);
        }

        public void set(int index, String string) {
                words.set(index, string);
        }
}


import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.util.StringTokenizer;


public class DocumentReader {
        private LabeledDocuments labeledDocuments;
        private Documents documents;

        public DocumentReader(String filePath, Boolean isLabeled) {
                Scanner documentScanner = null;

                try {
                        documentScanner = new Scanner(new File(filePath));
                }
                catch(FileNotFoundException e) {
                        System.out.println("File not found!");
                }

                if(isLabeled) {
                        labeledDocuments = new LabeledDocuments();
                        String line;
                        String document;
                        int polarity;

                        while(documentScanner.hasNextLine()) {
                                line = documentScanner.nextLine();
                                StringTokenizer token = new StringTokenizer(line, "\t");
                                document = token.nextToken();
                                polarity = Integer.parseInt(token.nextToken());
                                labeledDocuments.add(new LabeledDocument(document, polarity));
                        }
                }
                else {
                        documents = new Documents();

                        while(documentScanner.hasNextLine()) {
                                documents.add(documentScanner.nextLine());
                        }
                }
        }

        public LabeledDocuments getLabeledDocuments() {
                return labeledDocuments;
        }

        public Documents getDocuments() {
                return documents;
        }
}


import java.util.ArrayList;

public class Documents {

        private ArrayList<String> documents;

        public Documents(){
                documents = new ArrayList<String>();
        }

        public void add(String document){
                documents.add(document);
        }

        public int size(){
                return documents.size();
        }

        public String get(int index) {
                return documents.get(index);
        }
}
```

```
import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.IOException;

public class DocumentWriter {
        private String filePath;
        private boolean append = false;

        public DocumentWriter(String filePath) {
                this.filePath = filePath;
        }

        public void writeToFile(LabeledDocuments documents) throws IOException {
                FileWriter fileWriter = new FileWriter(filePath + ".txt", append);
                PrintWriter printWriter = new PrintWriter(fileWriter);

                for (int i = 0; i < documents.size(); i++) {
                        printWriter.println(documents.get(i).getDocument() + "\t" +
                                        documents.get(i).getPolarity());
                }

                printWriter.close();
        }

        public void writePositiveAndNegativeFiles(LabeledDocuments documents)
                        throws IOException {
                FileWriter fileWriterPositive = new FileWriter(filePath
                                + "(Positive).txt", append);
                FileWriter fileWriterNegative = new FileWriter(filePath
                                + "(Negative).txt", append);
                PrintWriter printWriterPositive = new PrintWriter(fileWriterPositive);
                PrintWriter printWriterNegative = new PrintWriter(fileWriterNegative);

                System.out.println("FILENAME: " + filePath + "(Positive).txt");
                System.out.println("FILENAME: " + filePath + "(Negative).txt");

                for (int i = 0; i < documents.size(); i++) {
                        if (documents.get(i).getPolarity() == 1) {
                                printWriterPositive.println(documents.get(i).getDocument());
                        }
                        else {
                                printWriterNegative.println(documents.get(i).getDocument());
                        }
                }

                printWriterPositive.close();
                printWriterNegative.close();
        }
}


public class LabeledDocument {
        private String document;
        private int polarity;

        public LabeledDocument(String document, int polarity) {
                this.document = document;
                this.polarity = polarity;
        }

        public String getDocument() {
                return document;
        }

        public int getPolarity() {
                return polarity;
        }

        public void setDocument(String document) {
                this.document = document;
        }

        public void setPolarity(int polarity) {
                this.polarity = polarity;
        }
}

import java.util.ArrayList;

public class LabeledDocuments {

        private ArrayList<LabeledDocument> labeledDocuments;

        public LabeledDocuments() {
                labeledDocuments = new ArrayList<LabeledDocument>();
        }

        public int size() {
                return labeledDocuments.size();
        }
```

```
        public LabeledDocument get(int index){
                return labeledDocuments.get(index);
        }

        public void add(LabeledDocument document) {
                labeledDocuments.add(document);
        }

        public ArrayList<LabeledDocument> getDocuments() {
                return labeledDocuments;
        }

        public Documents getPositiveDocuments(){
                Documents positiveDocuments = new Documents();

                for (int i = 0; i < labeledDocuments.size(); i++) {
                        if (labeledDocuments.get(i).getPolarity() == 1) {
                                positiveDocuments.add(labeledDocuments.get(i).getDocument());
                        }
                }
                return positiveDocuments;
        }

        public Documents getNegativeDocuments(){
                Documents negativeDocuments = new Documents();

                for (int i = 0; i < labeledDocuments.size(); i++) {
                        if (labeledDocuments.get(i).getPolarity() == 0) {
                                negativeDocuments.add(labeledDocuments.get(i).getDocument());
                        }
                }
                return negativeDocuments;
        }
}


public class Model {

        private Preprocessor preprocessor;
        private BagOfWords positiveWords;
        private BagOfWords negativeWords;

        public Model(LabeledDocuments labeledDocuments, Preprocessor preprocessor) {
                this.preprocessor = preprocessor;

                positiveWords = new BagOfWords(labeledDocuments.getPositiveDocuments());
                negativeWords = new BagOfWords(labeledDocuments.getNegativeDocuments());

                positiveWords = preprocessor.preprocess(positiveWords);
                negativeWords = preprocessor.preprocess(negativeWords);
        }

        public boolean isPositive(String document) {
                if (getProbability(true, document) >= getProbability(false, document)) {
                        return true;
                }
                else {
                        return false;
                }
        }

        // probability of a document being positive or negative
        private double getProbability(Boolean isPositive, String document) {
                BagOfWords documentWords = new BagOfWords(document);
                documentWords = preprocessor.preprocess(documentWords);
                documentWords.removeRepeatingWords();
                double probability = 1;

                for (int i = 0; i < documentWords.size(); i++) {
                        probability = (double)probability *
                                (double)getWordProbabilityInClass(isPositive,
                                 documentWords.get(i));
                }

                return (double)probability * (double)getClassProbability(isPositive);
        }

        // for review
        private double getClassProbability(Boolean isPositive) {
                if (isPositive) {
                        return (double)positiveWords.size()/
                                ((double)positiveWords.size() + negativeWords.size());
                }
                else {
                        return (double)negativeWords.size()/
                                ((double)positiveWords.size() + negativeWords.size());
                }
        }

        // probability of a word being in positive or negative class
        private double getWordProbabilityInClass(Boolean isPositive, String word) {
```

```
                int k = 1;

                if (isPositive) {
                        return (double)(positiveWords.getWordCount(word) + k)/
                                (double)((k+1)*positiveWords.size());
                }
                else {
                        return (double)(negativeWords.getWordCount(word) + k)/
                                (double)((k+1)*negativeWords.size());
                }
        }
}


public class NaiveBayesClassifier {
        private Model model;
        private Validator validator;

        public NaiveBayesClassifier(String trainingInputFilePath,
                        Preprocessor preprocessor) {
                DocumentReader documentReader = new DocumentReader(
                                trainingInputFilePath, true);
                LabeledDocuments documentsForTraining = documentReader.getLabeledDocuments();

                model = new Model(documentsForTraining, preprocessor);
                validator = new Validator(documentsForTraining, preprocessor);
        }

        public LabeledDocuments classifyDocuments(String inputFilePath) {
                DocumentReader documentReader = new DocumentReader(inputFilePath, false);
                Documents documents = documentReader.getDocuments();
                LabeledDocuments classifiedDocuments = new LabeledDocuments();

                for (int i = 0; i < documents.size(); i++) {
                        if (model.isPositive(documents.get(i))) {
                                classifiedDocuments.add(new LabeledDocument(documents.get(i), 1));
                        }
                        else {
                                classifiedDocuments.add(new LabeledDocument(documents.get(i), 0));
                        }
                }

                return classifiedDocuments;
        }

        public Validator getValidator() {
                return validator;
        }
}

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextArea;
import javax.swing.JTextPane;
import javax.swing.JButton;
import javax.swing.JRadioButton;
import javax.swing.ButtonGroup;
import javax.swing.JCheckBox;
import javax.swing.JSeparator;
import javax.swing.JFileChooser;
import javax.swing.ImageIcon;

import java.awt.Color;
import java.awt.Font;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.EventQueue;
import java.io.IOException;
import java.text.DecimalFormat;

import java.util.Timer;
import java.util.TimerTask;

public class NaiveBayesClassifierUI extends JFrame {

        private NaiveBayesClassifier nbc;
        private String outputFileDirectory;

        private JPanel contentPane;
        private JLabel training;
        private JLabel classifying;

        private ButtonGroup btnGrpOutputFile;
        private JRadioButton rdbtnTwoFiles;
        private JRadioButton rdbtnOneFile;

        private JButton btnBrowseTraining;
        private JButton btnTrain;
```

```
private JButton btnReset;

private JCheckBox chckbxTextNormalization;
private JCheckBox chckbxNegationHandling;
private JCheckBox chckbxNoiseWordsElimination;
private JCheckBox chckbxEmoticonConversion;
private JCheckBox chckbxInflatedOrDerived;

private JTextPane txtpnPrecision;
private JTextPane txtpnRecall;
private JTextPane txtpnAccuracy;
private JTextPane txtpnF1score;

private JButton btnBrowse;
private JButton btnClassify;
private JTextArea txtareaTrainingInput;
private JTextArea txtareaInput;
private JTextArea txtareaFileName;

/**
 * Launch the application.
 */
public static void main(String[] args) {
        SplashScreen splashScreen = new SplashScreen(3500);
        splashScreen.showSplash();

        EventQueue.invokeLater(new Runnable() {
                public void run() {
                        try {
                                NaiveBayesClassifierUI frame =
                                new NaiveBayesClassifierUI();
                                frame.setVisible(true);
                        } catch (Exception e) {
                                e.printStackTrace();
                        }
                }
        });
}

/**
 * Create the frame.
 */
public NaiveBayesClassifierUI() {
        ImageIcon img = new ImageIcon(this.getClass().getResource("/images/logo.png"));
        setIconImage(img.getImage());
        setResizable(false);
        setBackground(Color.WHITE);
        setForeground(Color.WHITE);
        setTitle("Sentiment Analysis Using Customizable Naive Bayes Classifier");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 500, 450);
        contentPane = new JPanel();
        contentPane.setBackground(Color.PINK);
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        JLabel lblInput = new JLabel("Input File");
        lblInput.setFont(new Font("Tahoma", Font.BOLD, 12));
        lblInput.setBounds(15, 35, 136, 28);
        contentPane.add(lblInput);

        txtareaTrainingInput = new JTextArea();
        txtareaTrainingInput.setBackground(Color.WHITE);
        txtareaTrainingInput.setBounds(20, 60, 350, 25);
        contentPane.add(txtareaTrainingInput);

        btnBrowseTraining = new JButton("Browse");
        btnBrowseTraining.addActionListener(new BrowseTrainingButtonHandler());
        btnBrowseTraining.setBackground(Color.LIGHT_GRAY);
        btnBrowseTraining.setBounds(375, 62, 110, 23);
        contentPane.add(btnBrowseTraining);

        JLabel lblPreprocessing = new JLabel("Preprocessing Methods");
        lblPreprocessing.setFont(new Font("Tahoma", Font.BOLD, 12));
        lblPreprocessing.setBounds(15, 90, 168, 28);
        contentPane.add(lblPreprocessing);

        chckbxTextNormalization = new JCheckBox("Text Normalization");
        chckbxTextNormalization.setBackground(Color.PINK);
        chckbxTextNormalization.setBounds(30, 115, 136, 23);
        contentPane.add(chckbxTextNormalization);

        chckbxNegationHandling = new JCheckBox("Negation Handling");
        chckbxNegationHandling.setBackground(Color.PINK);
        chckbxNegationHandling.setBounds(30, 135, 150, 23);
        contentPane.add(chckbxNegationHandling);

        chckbxNoiseWordsElimination = new JCheckBox("Noise Words Elimination");
        chckbxNoiseWordsElimination.setBackground(Color.PINK);
        chckbxNoiseWordsElimination.setBounds(30, 155, 180, 23);
        contentPane.add(chckbxNoiseWordsElimination);
```

```
chckbxEmoticonConversion = new JCheckBox("Emoticon Conversion");
chckbxEmoticonConversion.setBackground(Color.PINK);
chckbxEmoticonConversion.setBounds(30, 175, 168, 23);
contentPane.add(chckbxEmoticonConversion);

chckbxInflatedOrDerived = new JCheckBox("Inflated or"
                + " Derived Words Reduction");
chckbxInflatedOrDerived.setBackground(Color.PINK);
chckbxInflatedOrDerived.setBounds(30, 195, 240, 23);
contentPane.add(chckbxInflatedOrDerived);

btnTrain = new JButton("Train the Model");
btnTrain.addActionListener(new TrainButtonHandler());
btnTrain.setBackground(Color.LIGHT_GRAY);
btnTrain.setBounds(15, 225, 125, 23);
contentPane.add(btnTrain);

btnReset = new JButton("Reset");
btnReset.addActionListener(new ResetButtonHandler());
btnReset.setBackground(Color.LIGHT_GRAY);
btnReset.setBounds(145, 225, 125, 23);
contentPane.add(btnReset);

txtareaInput = new JTextArea();
txtareaInput.setBackground(Color.WHITE);
txtareaInput.setBounds(20, 320, 350, 25);
contentPane.add(txtareaInput);

JSeparator separator = new JSeparator();
separator.setBounds(275, 235, 210, 13);
contentPane.add(separator);

training = new JLabel(new ImageIcon(
this.getClass().getResource("/images/wait2.gif")));
training.setBounds(420, -5, 75, 75);
contentPane.add(training);

classifying = new JLabel(new ImageIcon(
this.getClass().getResource("/images/wait2.gif")));
classifying.setBounds(420, 255, 75, 75);
contentPane.add(classifying);

JLabel lblPerformance = new JLabel("Performance");
lblPerformance.setFont(new Font("Tahoma", Font.BOLD, 12));
lblPerformance.setBounds(260, 90, 128, 28);
contentPane.add(lblPerformance);

JLabel lblPrecision = new JLabel("Precision");
lblPrecision.setBounds(275, 115, 100, 23);
contentPane.add(lblPrecision);

JLabel lblAccuracy = new JLabel("Accuracy");
lblAccuracy.setBounds(275, 160, 100, 23);
contentPane.add(lblAccuracy);

JLabel lblRecall = new JLabel("Recall");
lblRecall.setBounds(390, 115, 100, 23);
contentPane.add(lblRecall);

JLabel lblFScore = new JLabel("F1 Score");
lblFScore.setBounds(390, 160, 100, 23);
contentPane.add(lblFScore);

txtpnPrecision = new JTextPane();
txtpnPrecision.setBounds(280, 135, 80, 25);
contentPane.add(txtpnPrecision);

txtpnRecall = new JTextPane();
txtpnRecall.setBounds(395, 135, 80, 25);
contentPane.add(txtpnRecall);

txtpnAccuracy = new JTextPane();
txtpnAccuracy.setBounds(280, 180, 80, 25);
contentPane.add(txtpnAccuracy);

txtpnF1score = new JTextPane();
txtpnF1score.setBounds(395, 180, 80, 25);
contentPane.add(txtpnF1score);

JLabel lblInputFile = new JLabel("Input File");
lblInputFile.setFont(new Font("Tahoma", Font.BOLD, 12));
lblInputFile.setBounds(15, 295, 100, 28);
contentPane.add(lblInputFile);

btnBrowse = new JButton("Browse");
btnBrowse.addActionListener(new BrowseButtonHandler());
btnBrowse.setBackground(Color.LIGHT_GRAY);
btnBrowse.setBounds(375, 322, 110, 23);
contentPane.add(btnBrowse);

JLabel lblOutputFormat = new JLabel("Output File Format");
```

```
        lblOutputFormat.setFont(new Font("Tahoma", Font.BOLD, 12));
        lblOutputFormat.setBounds(15, 350, 150, 28);
        contentPane.add(lblOutputFormat);

        rdbtnOneFile = new JRadioButton("filename.txt in format: "
                + "[statement]\\t[polarity]\\n");
        rdbtnOneFile.setBackground(Color.PINK);
        rdbtnOneFile.setBounds(30, 375, 320, 23);
        rdbtnOneFile.setSelected(true);
        contentPane.add(rdbtnOneFile);

        rdbtnTwoFiles = new JRadioButton("[filename](Positive).txt and "
                + "[filename](Negative).txt");
        rdbtnTwoFiles.setBackground(Color.PINK);
        rdbtnTwoFiles.setBounds(30, 395, 320, 23);
        contentPane.add(rdbtnTwoFiles);

        btnGrpOutputFile = new ButtonGroup();
        btnGrpOutputFile.add(rdbtnTwoFiles);
        btnGrpOutputFile.add(rdbtnOneFile);

        JLabel lblFileName = new JLabel("File Name");
        lblFileName.setFont(new Font("Tahoma", Font.BOLD, 12));
        lblFileName.setBounds(220, 350, 73, 28);
        contentPane.add(lblFileName);

        txtareaFileName = new JTextArea();
        txtareaFileName.setBounds(285, 350, 150, 25);
        contentPane.add(txtareaFileName);

        JLabel lblText = new JLabel(".txt");
        lblText.setBounds(440, 350, 25, 28);
        contentPane.add(lblText);

        btnClassify = new JButton("Classify");
        btnClassify.setFont(new Font("Tahoma", Font.BOLD, 11));
        btnClassify.addActionListener(new ClassifyButtonHandler());
        btnClassify.setBackground(Color.LIGHT_GRAY);
        btnClassify.setBounds(350, 390, 135, 23);
        contentPane.add(btnClassify);

        JSeparator separator_1 = new JSeparator();
        separator_1.setBounds(15, 260, 470, 5);
        contentPane.add(separator_1);

        JLabel lblClassifying = new JLabel("Classifying");
        lblClassifying.setFont(new Font("Tahoma", Font.BOLD, 18));
        lblClassifying.setBounds(15, 270, 125, 28);
        contentPane.add(lblClassifying);

        JLabel lblTraining = new JLabel("Training");
        lblTraining.setFont(new Font("Tahoma", Font.BOLD, 18));
        lblTraining.setBounds(15, 10, 125, 28);
        contentPane.add(lblTraining);

        training.setVisible(false);
        classifying.setVisible(false);
        txtareaInput.setEditable(false);
        txtareaTrainingInput.setEditable(false);
        setEnabledClassifyingButtons(false);
        setPerformanceEditable(false);
}

private class TrainButtonHandler implements ActionListener {
        public void actionPerformed(ActionEvent e) {
                if (txtareaTrainingInput.getText().length() == 0) {
                        JOptionPane.showMessageDialog(null, "Select a text file "
                                        + " training!");
                }
                else {
                        training.setVisible(true);
                        btnReset.setEnabled(false);
                        btnTrain.setEnabled(false);
                        btnBrowseTraining.setEnabled(false);
                        setEnabledPreprocessingMethods(false);
                        resetPerformanceTextPanes();

                        Timer timer = new Timer();
                        TimerTask task = new TimerTask() {
                        public void run() {
                                nbc =
                                new NaiveBayesClassifier(txtareaTrainingInput.getText(),
                                        new Preprocessor(
                                                chckbxTextNormalization.isSelected(),
                                                chckbxNegationHandling.isSelected(),
                                                chckbxNoiseWordsElimination.isSelected(),
                                                chckbxEmoticonConversion.isSelected(),
                                                chckbxInflatedOrDerived.isSelected()));

                                        DecimalFormat formatter =
                                        new DecimalFormat("#0.000000");
```

```
                                        txtpnPrecision.setText(formatter.format(
                                                nbc.getValidator().getPrecision()));
                                        txtpnRecall.setText(formatter.format(
                                                nbc.getValidator().getRecall()));
                                        txtpnAccuracy.setText(formatter.format(
                                                nbc.getValidator().getAccuracy()));
                                        txtpnF1score.setText(formatter.format(
                                                        nbc.getValidator().getF1Score()));

                                        training.setVisible(false);
                                        btnReset.setEnabled(true);
                                        setEnabledClassifyingButtons(true);
                                }
                        };

                        timer.schedule(task, 500);
                }
        }
}

private class ResetButtonHandler implements ActionListener {
        public void actionPerformed(ActionEvent e) {
                btnTrain.setEnabled(true);
                btnBrowseTraining.setEnabled(true);
                setEnabledPreprocessingMethods(true);
                resetPerformanceTextPanes();
                setEnabledClassifyingButtons(false);

                chckbxTextNormalization.setSelected(false);
                chckbxNegationHandling.setSelected(false);
                chckbxNoiseWordsElimination.setSelected(false);
                chckbxEmoticonConversion.setSelected(false);
                chckbxInflatedOrDerived.setSelected(false);
        }
}

private class BrowseTrainingButtonHandler implements ActionListener {
        public void actionPerformed(ActionEvent e) {
                JFileChooser fileChooser = new JFileChooser();
                fileChooser.setCurrentDirectory(new java.io.File("C:/"));
                fileChooser.setDialogTitle("Open a Text File");
                fileChooser.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);

                if (fileChooser.showOpenDialog(btnBrowseTraining) ==
                                JFileChooser.APPROVE_OPTION) {
                        txtareaTrainingInput.setText(
                                        fileChooser.getSelectedFile().getAbsolutePath());
                }
        }
}

private class BrowseButtonHandler implements ActionListener {
        public void actionPerformed(ActionEvent e) {
                JFileChooser fileChooser = new JFileChooser();
                fileChooser.setCurrentDirectory(new java.io.File("C:/"));
                fileChooser.setDialogTitle("Save File");
                fileChooser.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);

                if (fileChooser.showOpenDialog(btnBrowse) ==
                                JFileChooser.APPROVE_OPTION) {
                        txtareaInput.setText(
                                        fileChooser.getSelectedFile().getAbsolutePath());
                        outputFileDirectory =
                                fileChooser.getSelectedFile().getParentFile().getPath();
                }
        }
}

private class ClassifyButtonHandler implements ActionListener {
        public void actionPerformed(ActionEvent e) {
                if (txtareaInput.getText().length() == 0) {
                        JOptionPane.showMessageDialog(null,
                        "Select a text file for input!");
                }
                else if (txtareaFileName.getText().length() == 0) {
                        JOptionPane.showMessageDialog(null, "Enter file name!");
                }
                else {
                        classifying.setVisible(true);

                        Timer timer = new Timer();
                        TimerTask task = new TimerTask() {
                        public void run() {
                        LabeledDocuments classifiedDocuments =
                                nbc.classifyDocuments(txtareaInput.getText());

                        DocumentWriter documentWriter = new DocumentWriter(
                                outputFileDirectory + "\\" +
                                 txtareaFileName.getText());

                        try {
                                if (btnGrpOutputFile.getSelection().equals(
```

```
                                        rdbtnTwoFiles.getModel())) {
                                        documentWriter.writePositiveAndNegativeFiles(
                                                classifiedDocuments);
                                }
                                else if (btnGrpOutputFile.getSelection().equals(
                                        rdbtnOneFile.getModel())) {
                                        documentWriter.writeToFile(classifiedDocuments);
                                        }

                                classifying.setVisible(false);
                                JOptionPane.showMessageDialog(null,
                                "Positive Statements: "
                                + classifiedDocuments.getPositiveDocuments().size()
                                + "\n" + "Negative Statements: "
                                + classifiedDocuments.getNegativeDocuments().size());
                                }
                        catch (IOException ioe) {
                                        System.out.println("IOException!!!");
                                }
                        }
                        };

                        timer.schedule(task, 500);
                }
        }
    }

    private void setEnabledClassifyingButtons(boolean enabled) {
        btnClassify.setEnabled(enabled);
        btnBrowse.setEnabled(enabled);
        txtareaFileName.setEnabled(enabled);
        txtareaInput.setEnabled(enabled);
        rdbtnTwoFiles.setEnabled(enabled);
        rdbtnOneFile.setEnabled(enabled);
    }

    private void setEnabledPreprocessingMethods(boolean enabled) {
        chckbxTextNormalization.setEnabled(enabled);
        chckbxNegationHandling.setEnabled(enabled);
        chckbxNoiseWordsElimination.setEnabled(enabled);
        chckbxEmoticonConversion.setEnabled(enabled);
        chckbxInflatedOrDerived.setEnabled(enabled);
    }

    private void resetPerformanceTextPanes() {
        txtpnPrecision.setText("");
        txtpnRecall.setText("");
        txtpnAccuracy.setText("");
        txtpnF1score.setText("");
    }

    private void setPerformanceEditable(boolean editable) {
        txtpnPrecision.setEditable(editable);
        txtpnRecall.setEditable(editable);
        txtpnAccuracy.setEditable(editable);
        txtpnF1score.setEditable(editable);
    }
}

import java.util.ArrayList;

public class Preprocessor {

    boolean normalizeText;
    boolean handleNegations;
    boolean eliminateNoiseWords;
    boolean convertEmoticons;
    boolean reduceInflatedOrDerivedWords;

    private ArrayList<String> prepositions;
    private ArrayList<String> conjunctions;
    private ArrayList<String> determiners;
    private ArrayList<String> positiveEmoticons;
    private ArrayList<String> negativeEmoticons;
    private ArrayList<String> urlDomains;

    public Preprocessor(boolean normalizeText, boolean handleNegations,
        boolean eliminateNoiseWords, boolean convertEmoticons,
        boolean reduceInflatedOrDerivedWords) {

        this.normalizeText = normalizeText;
        this.handleNegations = handleNegations;
        this.eliminateNoiseWords = eliminateNoiseWords;
        this.convertEmoticons = convertEmoticons;
        this.reduceInflatedOrDerivedWords = reduceInflatedOrDerivedWords;
    }

    public BagOfWords preprocess(BagOfWords bow) {

        if (convertEmoticons) {
                loadPositiveEmoticons();
```

```
                loadNegativeEmoticons();

                bow = convertEmoticons(bow);
        }

        if (reduceInflatedOrDerivedWords) {
                bow = reduceInflatedOrDerivedWords(bow);
                bow.deleteBlankWords();
        }

        if (eliminateNoiseWords) {
                loadPrepositions();
                loadConjunctions();
                loadDeterminers();

                bow = eliminateNoiseWords(bow);
        }

        if (handleNegations) {
                bow = handleNegations(bow);
        }

        if (normalizeText) {
                loadUrlDomains();

                bow = normalizeText(bow);
                bow.deleteBlankWords();
        }

        return bow;
}

//remove hashtags and usernames
//convert to small letters
//remove symbols, punctuations
//remove repeating letters
//remove URLs
private BagOfWords normalizeText(BagOfWords bow){

        for (int i = 0; i < bow.size(); i++) {

                if (bow.get(i).charAt(0) == '#' || bow.get(i).charAt(0) == '@'
                        || containsURL(bow.get(i))) {

                        bow.remove(i);
                        i--;
                }
                else {
                        bow.set(i, bow.get(i).toLowerCase());
                        bow.set(i, removeSpecialCharacters(bow.get(i)));
                        bow.set(i, removeRepeatingCharacters(bow.get(i)));
                }
        }

        return bow;
}


//not_ + word
//prefix to not_word
private BagOfWords handleNegations(BagOfWords bow){

        for (int i = 0; i < bow.size(); i++) {
                if (bow.get(i).toLowerCase().equals("n't")) {
                        bow.set(i, "not");
                }
                else if (bow.get(i).toLowerCase().endsWith("n't")) {
                        bow.set(i, bow.get(i).substring(0, bow.get(i).length() - 3));
                        bow.add(i+1, "not");
                }
                else if (bow.get(i).toLowerCase().startsWith("de") ||
                                        bow.get(i).toLowerCase().startsWith("un") ||
                                        bow.get(i).toLowerCase().startsWith("im") ||
                                        bow.get(i).toLowerCase().startsWith("ir") ||
                                        bow.get(i).toLowerCase().startsWith("il") ||
                                        bow.get(i).toLowerCase().startsWith("in")) {
                        bow.set(i, "not_" + bow.get(i).substring(2));
                }
                else if (bow.get(i).toLowerCase().startsWith("mis") ||
                                bow.get(i).toLowerCase().startsWith("non") ||
                                bow.get(i).toLowerCase().startsWith("dis")) {
                        bow.set(i, "not_" + bow.get(i).substring(3));
                }

                if (bow.get(i).toLowerCase().equals("not") && i != bow.size() - 1) {
                        bow.set(i, "not_" + bow.get(i+1));
                        bow.remove(i+1);
                }
        }

        return bow;
}
```

```
//remove prepositions, conjunctions, determiners
private BagOfWords eliminateNoiseWords(BagOfWords bow) {

        for (int i = 0; i < bow.size(); i++) {
                if (isNoiseWord(bow.get(i).toLowerCase())) {
                        bow.remove(i);
                        i--;
                }
        }

        return bow;
}


private BagOfWords convertEmoticons(BagOfWords bow){
        for (int i = 0; i < bow.size(); i++) {
                if (positiveEmoticons.contains(bow.get(i))) {
                        bow.set(i, "positive_emoticon");
                }
                else if (negativeEmoticons.contains(bow.get(i))) {
                        bow.set(i, "negative_emoticon");
                }
        }

        return bow;
}

//connect, connected, connection, connecting
private BagOfWords reduceInflatedOrDerivedWords(BagOfWords bow) {
        for (int i = 0; i < bow.size(); i++) {
                if (bow.get(i).toLowerCase().endsWith("ed")) {
                        bow.set(i, bow.get(i).substring(0, bow.get(i).length() - 2));
                }
                else if (bow.get(i).toLowerCase().endsWith("ion") ||
                        (bow.get(i).toLowerCase().endsWith("ing"))) {

                        bow.set(i, bow.get(i).substring(0, bow.get(i).length() - 3));
                }
        }

        return bow;
}

private boolean containsURL(String string) {
        for (int i = 0; i < urlDomains.size(); i++) {
                if (string.contains(urlDomains.get(i))) {
                        return true;
                }
        }

        return false;
}

private String removeSpecialCharacters(String string) {
        StringBuilder sb = new StringBuilder(string);

        for (int i = 0; i < sb.length(); i++) {
                if(!Character.isAlphabetic(sb.charAt(i)) &&
                                !Character.isDigit(sb.charAt(i))
                        && !(sb.charAt(i) == '-')) {

                        sb.deleteCharAt(i);
                        i--;
                }
        }

        return sb.toString();
}

private String removeRepeatingCharacters(String string) {
        StringBuilder sb = new StringBuilder(string);

        for (int i = 0; i < sb.length() - 2; i++) {
                if(sb.charAt(i) == sb.charAt(i+1) && sb.charAt(i) == sb.charAt(i+2)) {
                        sb.deleteCharAt(i+2);
                        i--;
                }
        }

        return sb.toString();
}

private boolean isNoiseWord(String string) {
        if (prepositions.contains(string) || conjunctions.contains(string)
                || determiners.contains(string)) {

                return true;
        }
        else {
                return false;
        }
```

```
}

private void loadPrepositions() {
        prepositions = new ArrayList<String>();

        prepositions.add("aboard");
        prepositions.add("about");
        prepositions.add("above");
        prepositions.add("across");
        prepositions.add("after");
        prepositions.add("against");
        prepositions.add("ahead");
        prepositions.add("along");
        prepositions.add("alongside");
        prepositions.add("amid");
        prepositions.add("among");
        prepositions.add("apart");
        prepositions.add("around");
        prepositions.add("aside");
        prepositions.add("as");
        prepositions.add("at");
        prepositions.add("atop");
        prepositions.add("barring");
        prepositions.add("before");
        prepositions.add("behind");
        prepositions.add("below");
        prepositions.add("beneath");
        prepositions.add("beside");
        prepositions.add("besides");
        prepositions.add("between");
        prepositions.add("beyond");
        prepositions.add("but");
        prepositions.add("by");
        prepositions.add("concerning");
        prepositions.add("considering");
        prepositions.add("despite");
        prepositions.add("down");
        prepositions.add("during");
        prepositions.add("except");
        prepositions.add("for");
        prepositions.add("from");
        prepositions.add("in");
        prepositions.add("inside");
        prepositions.add("instead");
        prepositions.add("into");
        prepositions.add("like");
        prepositions.add("near");
        prepositions.add("nearby");
        prepositions.add("of");
        prepositions.add("off");
        prepositions.add("on");
        prepositions.add("onto");
        prepositions.add("opposite");
        prepositions.add("out");
        prepositions.add("outside");
        prepositions.add("over");
        prepositions.add("past");
        prepositions.add("regarding");
        prepositions.add("round");
        prepositions.add("since");
        prepositions.add("through");
        prepositions.add("throughout");
        prepositions.add("till");
        prepositions.add("to");
        prepositions.add("toward");
        prepositions.add("under");
        prepositions.add("underneath");
        prepositions.add("until");
        prepositions.add("unto");
        prepositions.add("up");
        prepositions.add("upon");
        prepositions.add("with");
        prepositions.add("within");
        prepositions.add("without");
}

private void loadConjunctions() {
        conjunctions = new ArrayList<String>();

        conjunctions.add("and");
        conjunctions.add("but");
        conjunctions.add("for");
        conjunctions.add("nor");
        conjunctions.add("or");
        conjunctions.add("so");
        conjunctions.add("yet");
        conjunctions.add("both");
        conjunctions.add("either");
        conjunctions.add("neither");
        conjunctions.add("whether");
        conjunctions.add("after");
        conjunctions.add("although");
```

```
        conjunctions.add("as");
        conjunctions.add("because");
        conjunctions.add("before");
        conjunctions.add("how");
        conjunctions.add("if");
        conjunctions.add("lest");
        conjunctions.add("provided");
        conjunctions.add("since");
        conjunctions.add("than");
        conjunctions.add("that");
        conjunctions.add("though");
        conjunctions.add("till");
        conjunctions.add("unless");
        conjunctions.add("until");
        conjunctions.add("when");
        conjunctions.add("whenever");
        conjunctions.add("where");
        conjunctions.add("wherever");
        conjunctions.add("while");
        conjunctions.add("accordingly");
        conjunctions.add("again");
        conjunctions.add("also");
        conjunctions.add("besides");
        conjunctions.add("consequently");
        conjunctions.add("finally");
        conjunctions.add("furthermore");
        conjunctions.add("however");
        conjunctions.add("indeed");
        conjunctions.add("moreover");
        conjunctions.add("nevertheless");
        conjunctions.add("otherwise");
        conjunctions.add("then");
        conjunctions.add("therefore");
        conjunctions.add("thus");
}

private void loadDeterminers() {
        determiners = new ArrayList<String>();

        determiners.add("the");
        determiners.add("my");
        determiners.add("your");
        determiners.add("his");
        determiners.add("her");
        determiners.add("its");
        determiners.add("our");
        determiners.add("their");
        determiners.add("whose");
        determiners.add("this");
        determiners.add("that");
        determiners.add("these");
        determiners.add("those");
        determiners.add("which");
        determiners.add("a");
        determiners.add("an");
        determiners.add("any");
        determiners.add("another");
        determiners.add("other");
        determiners.add("what");
}

private void loadPositiveEmoticons() {
        positiveEmoticons = new ArrayList<String>();

        positiveEmoticons.add(":-)");
        positiveEmoticons.add(":)");
        positiveEmoticons.add(":-]");
        positiveEmoticons.add(":]");
        positiveEmoticons.add(":-3");
        positiveEmoticons.add(":3");
        positiveEmoticons.add(":->");
        positiveEmoticons.add(":>");
        positiveEmoticons.add("8-)");
        positiveEmoticons.add("8)");
        positiveEmoticons.add(":-}");
        positiveEmoticons.add(":}");
        positiveEmoticons.add(":o)");
        positiveEmoticons.add(":c)");
        positiveEmoticons.add(":^)");
        positiveEmoticons.add("=]");
        positiveEmoticons.add("=)");
        positiveEmoticons.add(":-D");
        positiveEmoticons.add(":D");
        positiveEmoticons.add("8-D");
        positiveEmoticons.add("8D");
        positiveEmoticons.add("x-D");
        positiveEmoticons.add("xD");
        positiveEmoticons.add("X-D");
        positiveEmoticons.add("XD");
        positiveEmoticons.add("=D");
        positiveEmoticons.add("=3");
        positiveEmoticons.add("B^D");
```

```
                positiveEmoticons.add(":-))");
                positiveEmoticons.add(":'-)");
                positiveEmoticons.add(":')");
        }

        private void loadNegativeEmoticons() {
                negativeEmoticons = new ArrayList<String>();

                negativeEmoticons.add(":-(");
                negativeEmoticons.add(":(");
                negativeEmoticons.add(":-c");
                negativeEmoticons.add(":c");
                negativeEmoticons.add(":-<");
                negativeEmoticons.add(":<");
                negativeEmoticons.add(":-[");
                negativeEmoticons.add(":[");
                negativeEmoticons.add(":-||");
                negativeEmoticons.add(">:[");
                negativeEmoticons.add(":{");
                negativeEmoticons.add(":@");
                negativeEmoticons.add(">:(");
                negativeEmoticons.add(":'-(");
                negativeEmoticons.add(":'(");
                negativeEmoticons.add("D-':");
                negativeEmoticons.add("D:<");
                negativeEmoticons.add("D:");
                negativeEmoticons.add("D8");
                negativeEmoticons.add("D;");
                negativeEmoticons.add("D=");
                negativeEmoticons.add("DX");
        }

        private void loadUrlDomains() {
                urlDomains = new ArrayList<String>();

                urlDomains.add(".com");
                urlDomains.add(".org");
                urlDomains.add(".net");
                urlDomains.add(".int");
                urlDomains.add(".edu");
                urlDomains.add(".gov");
                urlDomains.add(".mil");
        }
}

import java.awt.BorderLayout;
import java.awt.Color;

import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JWindow;


public class SplashScreen extends JWindow {
        private int duration;

        public SplashScreen(int d) {
                duration = d;
        }

        public void showSplash() {
                JPanel content = (JPanel)getContentPane();
                content.setBackground(Color.white);
                setBounds(100, 100, 600, 300);
                setLocationRelativeTo(null);

                JLabel label = new JLabel(new ImageIcon(
                this.getClass().getResource("/images/splash2.PNG")));
                content.add(label, BorderLayout.CENTER);

                setVisible(true);

                try {
                        Thread.sleep(duration);
                }
                catch (Exception e) {}

                setVisible(false);
        }
}


public class Validator {
        private double precision;
        private double accuracy;
        private double recall;
        private double f1Score;
        private int k = 10;
        private LabeledDocuments trainingDocuments;
```

```
public Validator(LabeledDocuments trainingDocuments, Preprocessor preprocessor) {
        this.trainingDocuments = trainingDocuments;

        LabeledDocuments trainingSet = new LabeledDocuments();
        LabeledDocuments testingSet = new LabeledDocuments();

        double precisionValues[] = new double[k];
        double accuracyValues[] = new double[k];
        double recallValues[] = new double[k];
        double f1Scores[] = new double[k];

        int truePositive;
        int trueNegative;
        int falsePositive;
        int falseNegative;

        if(trainingDocuments.size() < 10) {
                k = trainingDocuments.size();
        }

        for (int i = 0; i < k; i++) {
                truePositive = 0;
                trueNegative = 0;
                falsePositive = 0;
                falseNegative = 0;

                trainingSet = getTrainingSet(i);
                testingSet = getTestingSet(i);

                Model model = new Model(trainingSet, preprocessor);

                for (int h = 0; h < testingSet.size(); h++) {
                        LabeledDocument document = testingSet.get(h);

                        if (model.isPositive(document.getDocument())) {
                                if (document.getPolarity() == 1) {
                                        truePositive++;
                                }
                                else {
                                        falsePositive++;
                                }
                        }
                        else {
                                if (document.getPolarity() == 0) {
                                        trueNegative++;
                                }
                                else {
                                        falseNegative++;
                                }
                        }
                }

                precisionValues[i] = computePrecision(truePositive, falsePositive);
                accuracyValues[i] = computeAccuracy(truePositive, trueNegative,
                        falsePositive, falseNegative);
                recallValues[i] = computeRecall(truePositive, falseNegative);
                f1Scores[i] = computeF1Score(precisionValues[i], recallValues[i]);
        }

        precision = getAverage(precisionValues);
        accuracy = getAverage(accuracyValues);
        recall = getAverage(recallValues);
        f1Score = getAverage(f1Scores);
}

private LabeledDocuments getTrainingSet(int index) {
        LabeledDocuments trainingSet = new LabeledDocuments();

        for (int i = 0; i < trainingDocuments.size(); i++) {
                if (i%10 != index) {
                        trainingSet.add(trainingDocuments.get(i));
                }
        }

        return trainingSet;
}

private LabeledDocuments getTestingSet(int index) {
        LabeledDocuments testingSet = new LabeledDocuments();

        for (int i = 0; i < trainingDocuments.size(); i++) {
                if (i%10 == index) {
                        testingSet.add(trainingDocuments.get(i));
                }
        }

        return testingSet;
}

private double getAverage(double[] values) {
        double sum = 0;
```

```
                for (int i = 0; i < k; i++) {
                        sum = sum + values[i];
                }

                return sum / k;
        }

        private double computePrecision(int truePositive, int falsePositive) {
                return (double)(truePositive /
                                (double)(truePositive + falsePositive));
        }

        private double computeAccuracy(int truePositive, int trueNegative,
                int falsePositive, int falseNegative) {

                return (double)(truePositive + trueNegative) /
                        (double)(truePositive + trueNegative + falsePositive + falseNegative);
        }

        private double computeRecall(int truePositive, int falseNegative) {
                return (double)(truePositive) /
                                (double)(truePositive + falseNegative);
        }

        private double computeF1Score(double precision, double recall) {
                return (2 * precision + recall) /
                                (precision + recall);
        }

        public double getPrecision() {
                return precision;
        }

        public double getAccuracy() {
                return accuracy;
        }

        public double getRecall() {
                return recall;
        }

        public double getF1Score() {
                return f1Score;
        }
}
```

XI. Acknowledgement

"Sa Diyos ang lahat ng Kapurihan!" (Roderos, 2017)