SMS Spam Detection using various Machine Learning Classifiers

Project Report

HARCOURT BUTLER TECHNICAL UNIVERSITY

Submitted to:
Ms. Ankita Gautam
Ms. Suman Lata

Submitted by:
Prateek Gupta (170108019)
Pratish Katiyar (170108020)
Ranjeet Singh (170108021)
Sapna Tomar (170108024)

INDEX OF CONTENTS

SUMMARY

Over recent years, as the popularity of mobile phone devices has increased, Short Message Service (SMS) has grown into a multi-billion dollars industry. At the same time, reduction in the cost of messaging services has resulted in growth in unsolicited commercial advertisements (spams) being sent to mobile phones. SMS spamming is an activity of sending ‘unwanted messages’ through text messaging or other communication services; normally using mobile phones. The SMS spam problem can be approached with legal, economic or technical measures. Nowadays there are many methods for SMS spam detection, ranging from the list-based, statistical algorithm, IP-based and using machine learning. However, an optimum method for SMS spam detection is difficult to find due to issues of SMS length, battery and memory performances. A database of real SMS Spams from UCI Machine Learning repository is used, and after preprocessing and feature extraction, different machine learning techniques are applied to the database. Among the wide range of technical measures, Bayesian filters are playing a key role in stopping sms spam. Here, we analyze to what extent Bayesian filtering techniques can be applied to the problem of detecting and stopping mobile spam. In particular, we have built SMS spam test collections of significant size in English. We have tested on them a number of messages representation techniques and Machine Learning algorithms, in terms of effectiveness. The effectiveness of the proposed features is empirically validated using multiple classification methods. The results demonstrate that the proposed features can improve the performance of SMS spam detection.

INTRODUCTION

Spams are unwanted messages which can be transmitted over a communication media such as SMS. According to TIME1 and Digital Trends2, 6 billion out of 7 billion people in the world have access to cellphones and it is going to increase to 7.3 billion by 2014. Thus, the number of cellphones will soon outgrow the world population. In 2012, there were more than 6 billion daily Short Message Service (SMS) exchanges over mobile phones just in the US3, and the rate of SMS spams increased by 400 percent. These spam messages not only waste network resources but also increase the cost for mobile phone users and even lead to cyber attacks such as phishing. Therefore, there is a strong need for SMS spam detection.

SPAM FILTERING IN TEXT MESSAGES VS EMAIL

A number of major differences exist between spam-filtering in text messages and emails. Unlike emails, which have a variety of large datasets available, real databases for SMS spams are very limited. Additionally, due to the small length of text messages, the number of features that can be used for their classification is far smaller than the corresponding number in emails. Here, no header exists as well. Additionally, text messages are full of abbreviations and have much less formal language that what one would expect from emails. All of these factors may result in serious degradation in performance of major email spam filtering algorithms applied to short text messages. There are two major types of methods for detecting SMS spam: collaborative based and content based methods. The first one is based on the feedbacks from users and shared user experience. The second one is focused on analyzing the textual content of messages. This research adopts the second approach, which is more popular due to the difficulty in getting access to the data about usage and user experience. Content spam detection can be further classified into dynamic and static approaches .However, studies have only considered words or tokens without looking into deep-level semantics. The choice of words and tokens can be easily manipulated by spammers. As a result, these detection methods have limited use because they are incapable to deal with constantly evolving spamming tactics.

To address the limitations of the state of research on SMS spam detection, we propose a content-based method that leverages lexical semantics. Instead of relying on individual words, our proposed method uses semantic categories of words as features, which allows us to handle variations in word choices by spammers. In addition, using categories of words as features also helps to reduce the feature space, which in turn improves the efficiency of spam detection that has significant implications for SMS users. An empirical evaluation of the proposed methods has shown promising results.In this project, the goal is to apply different machine learning algorithms to SMS spam classification problem, compare their performance to gain insight and further explore the problem, and design an application based on one of these algorithms that can filter SMS spams with high accuracy. We use a database of 5574 text messages from UCI Machine Learning repository gathered in 2012 . It contains a collection of 425 SMS spam messages manually extracted from the Grumbletext Web site (a UK forum in which cell phone users make public claims about SMS spam), a subset of 3,375 SMS randomly chosen non-spam (ham) messages of the NUS SMS Corpus (NSC), a list of 450 SMS non-spam messages collected from Caroline Tag's PhD Thesis. The dataset is a large text file, in which each line starts with the label of the message, followed by the text message string. After preprocessing of the data and extraction of features, machine learning techniques such as naive Bayes, SVM, and other methods are applied to the samples, and their performances are compared. Finally, the performance of best classifier from the project is compared against the performance of classifiers applied in the original paper citing this dataset

CHALLENGES

- Unlike emails, databases for spam SMS are very limited.
- SMS's are limited in length, number of features that can be used for classification is small.
- Text messages generally include abbreviations, informal language, text-speak, other languages written in english.

DATASET

The SMS Spam Collection is a set of SMS tagged messages that have been collected for SMS Spam research. It contains one set of SMS messages in English of 5,574 messages, tagged according being ham (legitimate) or spam. The files contain one message per line. Each line is composed by two columns: v1 contains the label (ham or spam) and v2 contains the raw text. This corpus has been collected from free or free for research sources at the Internet:

- A collection of 425 SMS spam messages was manually extracted from the Grumbletext Web site. This is a UK forum in which cell phone users make public claims about SMS spam messages, most of them without reporting the very spam message received. The identification of the text of spam messages in the claims is a very hard and time-consuming task, and it involved carefully scanning hundreds of web pages.
- A subset of 3,375 SMS randomly chosen ham messages of the NUS SMS Corpus (NSC), which is a dataset of about 10,000 legitimate messages collected for research at the Department of Computer Science at the National University of Singapore. The messages largely originate from Singaporeans and mostly from students attending the University. These messages were collected from volunteers who were made aware that their contributions were going to be made publicly available.
- A list of 450 SMS ham messages collected from Caroline Tag's PhD Thesis.
- Finally, we have incorporated the SMS Spam Corpus v.0.1 Big. It has 1,002 SMS ham messages and 322 spam messages and it is public available.

PROBLEM DEFINITION

Objective: **To identify text messages/sms as spam or ham(non-spam)**

Problem Formulation

In SMS, there are a set of k text messages TM = {tm1 , … , tmk}. Each message is limited to 160 characters consisting of words, number, etc. Messages can be about any topic.
The tasks of SMS spam detection is to predict whether tmi is a spam (A) or non-spam (B) by using a classifier c. The problem is formulated below:

$$c: tm_i \rightarrow \{spam, non - spam\}$$

To support the classification, we need to first extract a set of n features F = {f1 , … , fn} from TM.

A Framework

We propose a framework for detecting spam messages in SMS (see Figure). This framework is composed of two major components: feature extraction and classification. The goal of feature extraction is to transform the input data into a set of features. It is very important in text analysis because it has a direct effect on machine learning techniques to distinguish classes or clusters; moreover, it is hard to find good features in unstructured data. In feature extraction step, we extracted two categories of features which will be introduced in detail in the following sections. In addition, we explored a wide range of classification algorithms from Random Forest to Naive Bayes and different test options to evaluate our proposed framework.

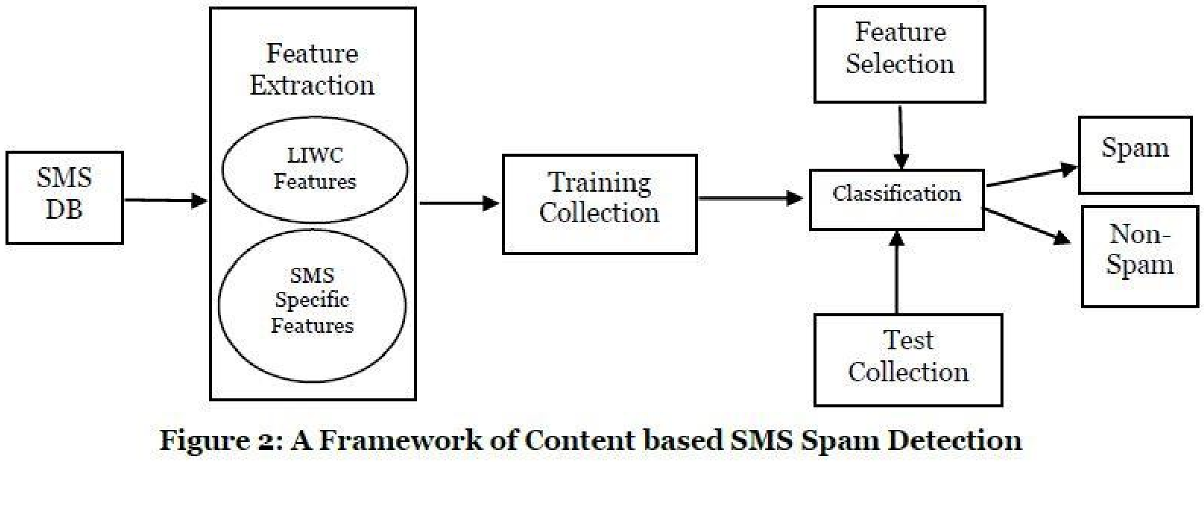


Figure 2: A Framework of Content based SMS Spam Detection

BACKGROUND STUDY

WHAT IS MACHINE LEARNING

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves. Machine learning is so pervasive today that you probably use it dozens of times a day without knowing it. In the past decade, machine learning has given us self-driving cars, practical speech recognition, effective web search, and a vastly improved understanding of the human genome.
The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly. Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly. Combining machine learning with AI and cognitive technologies can make it even more effective in processing large volumes of information.

MACHINE LEARNING METHODS

SUPERVISED LEARNING : Supervised learning algorithms are trained using labeled examples, such as an input where the desired output is known. For example, a piece of equipment could have data points labeled either “F” (failed) or “R” (runs). The learning algorithm receives a set of inputs along with the corresponding correct outputs, and the algorithm learns by comparing its actual output with correct outputs to find errors. It then modifies the model accordingly. Through methods like classification, regression, prediction and gradient boosting, supervised learning uses patterns to predict the values of the label on additional unlabeled data. Supervised learning is commonly used in applications where historical data predicts likely future events. For example, it can anticipate when credit card transactions are likely to be fraudulent or which insurance customer is likely to file a claim.

UNSUPERVISED LEARNING : Unsupervised machine learning algorithms are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn’t figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.

MACHINE LEARNING ALGORITHMS:

- Naive Bayes (NB): This classification technique is based on Bayes' theorem, assuming independence between predictors. The Bayesian classifier assumes that a particular feature in a class is not related to the presence of any other feature. Even if they do depend upon each other or upon the existence of the feature, the Bayesian classifier will consider all of the desired properties to independently contribute to the probability. The classifier holds well when the desired input's dimensionality is high. It is regarded as simple and sturdy. An advanced version of NB is Multinomial Naive Bayes (MNB). The main improvement is the independence between document length and class. It involves multinomial distribution which works well for countable type of data such as the words in a document or text. In simple terms, NB classifier involves conditional independence of each of the features in the model, whereas MNB classifier is a special case of a NB classifier which uses a multinomial distribution for each feature. an input layer, some hidden layers and an output layer, but it is not fully connected. Some layers are convolutional, that use a mathematical model to pass on the results to layers ahead in the network.
- SUPPORT VECTOR MACHINES (SVM): SVM is a discriminative classifier which is widely used for classification task. The algorithm plots each data item as a point in n-dimensional space assuming the value of each feature as the value of a particular coordinate. It then forms a line that splits the whole data into two differently classified groups of data. The closest points in the two groups will be the farthest away from this line.Linear kernel gains better performance compared to other mappings. Using the polynomial kernel and increasing the degree of the polynomial from two to three shows improvement in error rates, however the error rate does not improve when the degree is increased further. Finally, applying the sigmoid kernel results in all messages being classified as hams.
- Decision Tree (DT): DT is a supervised learning algorithm which is normally preferred for classification tasks. The algorithm works well for both types of variables i.e., categorical and continuous. It starts with splitting the population into multiple homogeneous sets which is done on the basis of most significant attributes or independent variables. DT is non-parametric and hence the need for checking outlier existence or data linearity separation is not required.
- Logistic Regression (LR): It is considered as the go-to method for classification involving binary results. It is mainly used in estimating discrete values which are based on set of variables which are independent. In more relative terms, LR outputs the probability of an event by fitting it into a logistic function which helps in prediction. The logistic function which is mostly used is sigmoid.
- Random Forest (RF): It is a term used for an ensemble of decision trees. The Random Forest classifier is a ensemble learning method which involves collection of decision trees. Voting is done to classify a new object which is performed by each tree i.e., the trees mark their votes for that class. The class having most number of votes decides the classification label.

- AdaBoost: AdaBoost or Adaptive Boosting is a meta- machine learning algorithm, used to increase the performance of a classifier by simply using the weak classifiers to combine them into a strong one. The final output of the boosted classifier depends upon the weighted sum of the output of all the weak classifiers. A drawback of this technique is that although it predicts more accurately, it takes more time for building the boosted model.
- K-nearest neighbors: K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification predictive problems in industry. The following two properties would define KNN well :− Lazy learning algorithm − KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification and Non-parametric learning algorithm − KNN is also a non-parametric learning algorithm because it doesn’t assume anything about the underlying data.
- Bagging Classifier: A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.Each base classifier is trained in parallel with a training set which is generated by randomly drawing, with replacement, N examples(or data) from the original training dataset – where N is the size of the original training set. Training set for each of the base classifiers is independent of each other. Many of the original data may be repeated in the resulting training set while others may be left out.
- Extra Tree Classifier: Extremely Randomized Trees Classifier(Extra Trees Classifier) is a type of ensemble learning technique which aggregates the results of multiple de-correlated decision trees collected in a “forest” to output it’s classification result. In concept, it is very similar to a Random Forest Classifier and only differs from it in the manner of construction of the decision trees in the forest. Each Decision Tree in the Extra Trees Forest is constructed from the original training sample. Then, at each test node, Each tree is provided with a random sample of k features from the feature-set from which each decision tree must select the best feature to split the data based on some mathematical criteria (typically the Gini Index). This random sample of features leads to the creation of multiple de-correlated decision trees.

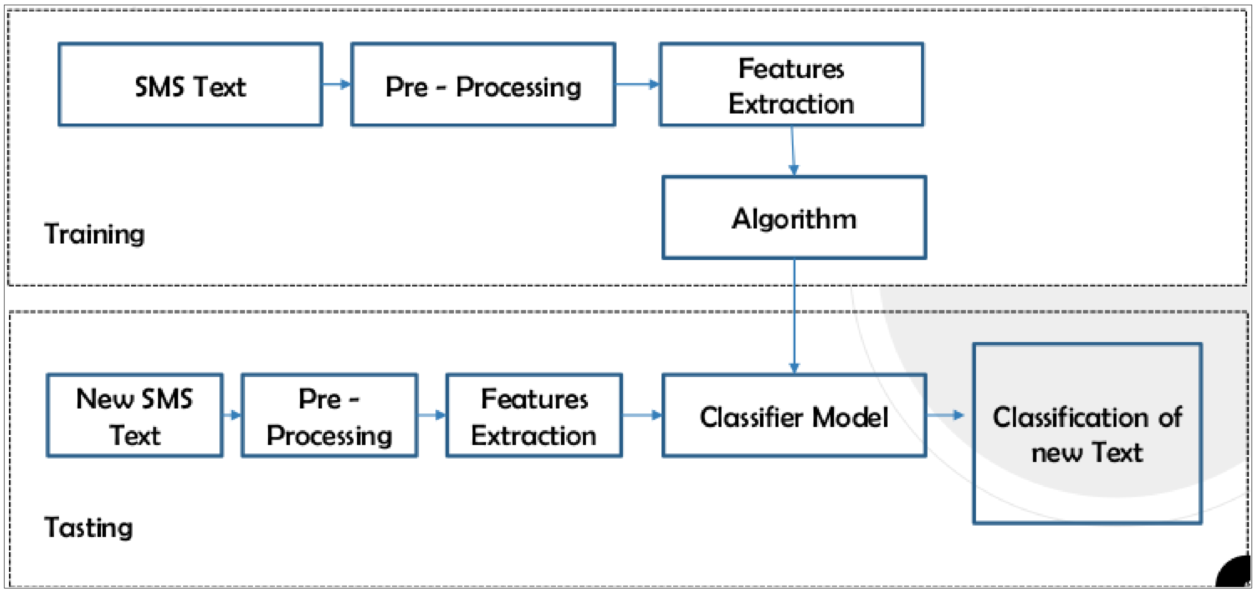
PROPOSED METHODOLGY

FRAMEWORK

We propose a framework for detecting spam messages in SMS. This framework is composed of two major components: feature extraction and classification.

Feature Extraction:
The goal of feature extraction is to transform the input data into a set of features. It is very important in text analysis because it has a direct effect on machine learning techniques to distinguish classes or clusters. For the initial analysis of the data, each message in dataset is split into tokens of alphabetic characters.

Classification:
After the feature extraction, classification is done initial analysis on the data is done using Logistic Regression, and based on the results, next steps are determined.
The quality of a classifier depends on the quality of features.



IMPLEMENTATION

Pre-processing:

- The dataset is split – 70% into training data, 30% to testing data.
- The training data is read line by line and the label and the message are separated.
- The message is tokenized.

Feature vectors:

The feature vectors for each message are computed with their frequency counts as the values.

Classification:

The Logistic Regression classifier is used here**.** For classification, I will use library Scikit-learn. It contains many methods for:

- ◆ classification, regression and anomaly detection.
- ◆ implements the k-nearest neighbors algorithm.
- ◆ decision tree-based models for classification and regression.
- ◆ also implements ensemble methods – random forest, adaboost, bagging classifier.
- ◆ implements Naive Bayes algorithms and other modules.

Each of this module is a black box with the same interface.

- **Fit**() — method for teaching classifier.
- **Score**() — method that returns the mean accuracy on the given test data and labels.
- **Predict**() — method for making prediction. For example: 'ham' or 'spam'.
- **Predict_score**() — method that returns probability estimates for each predictions. For example: 0.8 for 'ham' and 0.2 for 'spam'.

Vectorizers

Every module does not understand plain text, they need an array of features. How to build feature vectors from plain text?
For it, Scikit-learn has vectorizers: CountVectorizer, TfidfVectorizer.
CountVectorizeris a great tool provided by the scikit-learn library in Python. It is used to transform a given text into a vector on the basis of the frequency (count) of each word

that occurs in the entire text. This is helpful when we have multiple such texts, and we wish to convert each word in each text into vectors (for using in further text analysis). CountVectorizer creates a matrix in which each unique word is represented by a column of the matrix, and each text sample from the document is a row in the matrix. The value of each cell is nothing but the count of the word in that particular text sample.

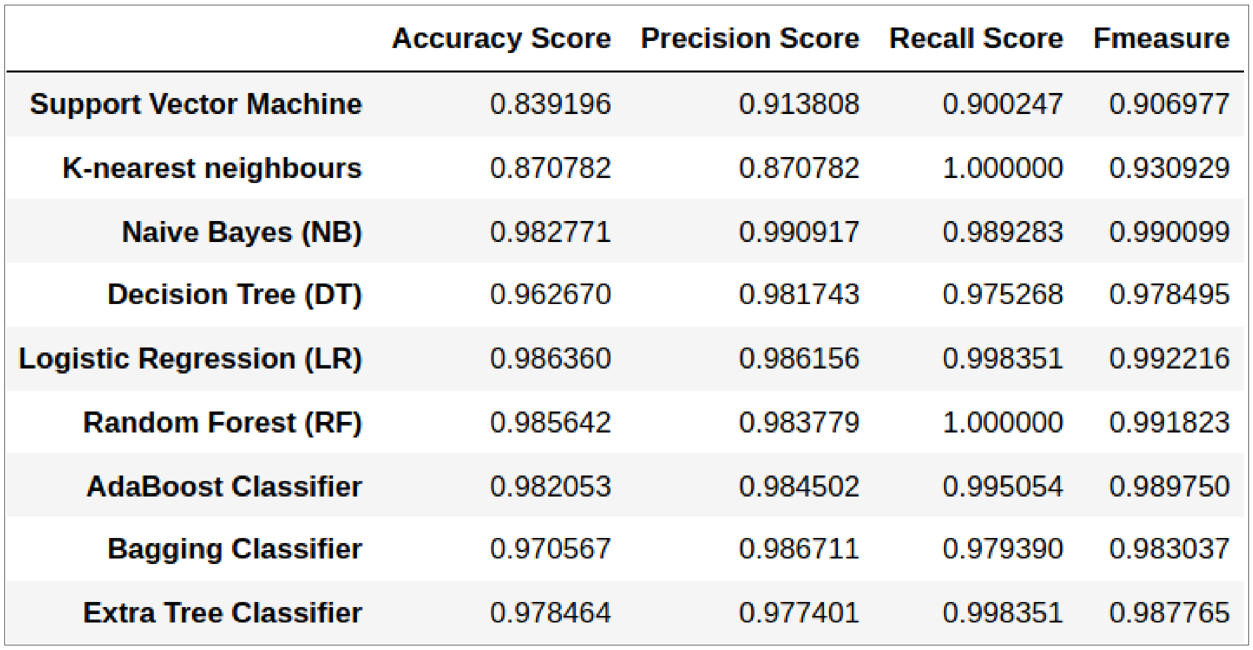
Testing

How to test score of classification? There is one way. We need to divide one data-set (spam.csv) to two data-sets (teach-set and test-set) with the ratio 80/20 or 70/30. We will use teach-set for teaching classifier and test-set for calculating accuracy score, precision score, recall score and f-measure.

Accuracy score measures how often the classifier makes the correct prediction. It’s the ratio of the number of correct predictions to the total number of predictions (the number of test data points).
Precision score tells us what proportion of messages we classified as spam, actually were spam. It is a ratio of true positives(words classified as spam, and which are actually spam) to all positives(all words classified as spam, irrespective of whether that was the correct classification), in other words it is the ratio of [True Positives/(True Positives + False Positives)]
Recall score(sensitivity) tells us what proportion of messages that actually were spam were classified by us as spam. It is a ratio of true positives(words classified as spam, and which are actually spam) to all the words that were actually spam, in other words it is the ratio of
[True Positives/(True Positives + False Negatives)]
For classification problems that are skewed in their classification distributions like in our case, for example if we had a 100 text messages and only 2 were spam and the rest 98 weren't, accuracy by itself is not a very good metric. We could classify 90 messages as not spam(including the 2 that were spam but we classify them as not spam, hence they would be false negatives) and 10 as spam(all 10 false positives) and still get a reasonably good accuracy score. For such cases, precision and recall come in very handy. These two metrics can be combined to get the F1 score or F-measure, which is weighted average of the precision and recall scores. This score can range from 0 to 1, with 1 being the best possible F1 score.
We will be using all 4 metrics to make sure our model does well. For all 4 metrics whose values can range from 0 to 1, having a score as close to 1 as possible is a good indicator of how well our model is doing.

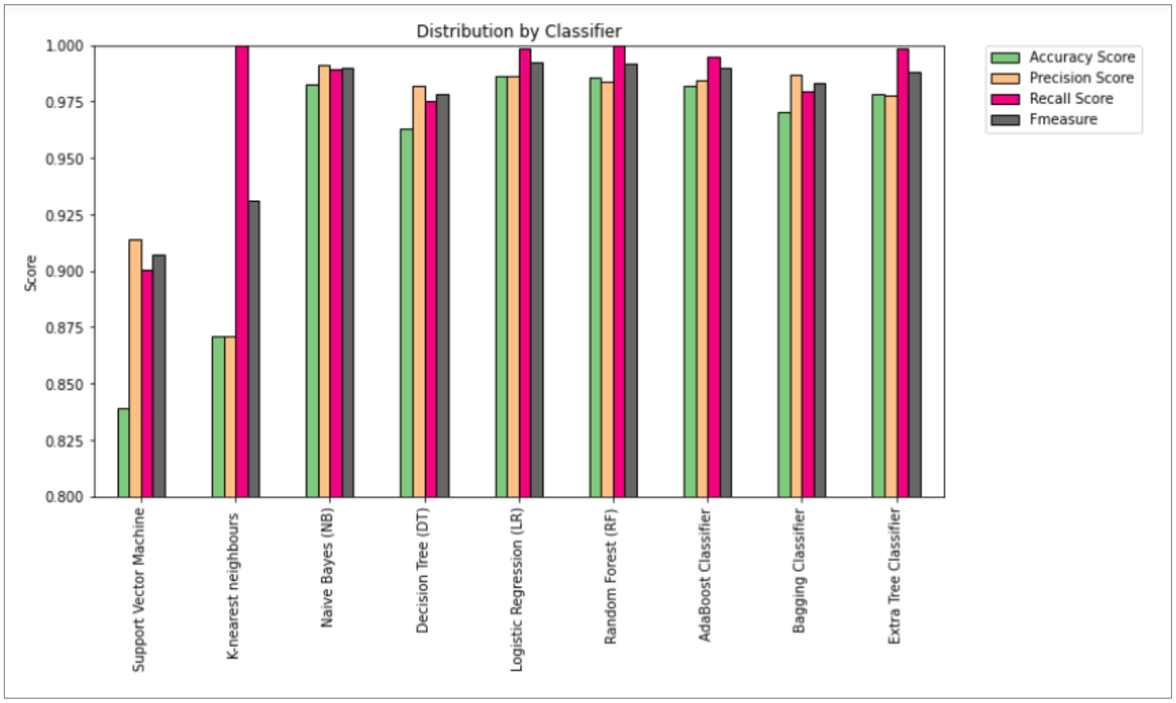
RESULTS

- The training of the classifier is very fast, < 1Second on a Core i3 with 8GB DDR3 RAM.
- Testing takes far lesser time.
- The accuracy score, precision score, recall score and f-measure computed by testing the classifier on the test data is shown in the following output of our implementation.



	Accuracy Score	Precision Score	Recall Score	Fmeasure
Support Vector Machine	0.839196	0.913808	0.900247	0.906977
K-nearest neighbours	0.870782	0.870782	1.000000	0.930929
Naive Bayes (NB)	0.982771	0.990917	0.989283	0.990099
Decision Tree (DT)	0.962670	0.981743	0.975268	0.978495
Logistic Regression (LR)	0.986360	0.986156	0.998351	0.992216
Random Forest (RF)	0.985642	0.983779	1.000000	0.991823
AdaBoost Classifier	0.982053	0.984502	0.995054	0.989750
Bagging Classifier	0.970567	0.986711	0.979390	0.983037
Extra Tree Classifier	0.978464	0.977401	0.998351	0.987765

Based on the experiment, CountVectorizer with Multinomial Naive Bayes is good but CountVectorizer with Logistic Regression classification algorithm outperforms good compare to other algorithms in terms of accuracy percentage. However, it is not enough to evaluate the performance based on the accuracy alone since the dataset is imbalanced; therefore, the precision, recall and f-measure of the algorithms must also be observed. After some examinations, Logistic Regression algorithm still manages to provide good precision and f-measure with 0.98 of precision while 0.99 for f-measure and accuracy(98.7% approx.) greater than all the various classifiers used.



A few things to note about the limitations of our code:

- Our code fails to understand the similarity of words like man and men, bed and beds etc. This can be solved using stemming lemmatization.
- The classification accuracy can be improved by using more sophisticated algorithms like neural networks.

DISCUSSION AND CONCLUSION

The recent surge of mobile phone use makes the emerging communication media such as SMS articularly attractive for spammers. The challenge of detection spams in SMS is due the small number of characters in short text messages and the common use of idioms and abbreviation. The research that does exist on SMS spam detection has only focused on word distribution but has yet to examine explicit semantic categories of text expressions.

Content based filtering suffers from challenges like short content, abbreviated words and user content safety. All of the studies tried to solve some challenges of SMS spam detection. Logistic Regression algorithm also suffers from traditional threshold selection problem, dataset dependency, assuming prior probability. Despite having those shortcomings, Logistic Regression classifier is declared as the most suitable algorithm for spam filtering.

In the current study, we proposed to employ categories of lexical semantic features in the detection of SMS spam. Our experiment results show that incorporating semantic categories improve the performance of SMS spam detection.The features identified in this study may be applied to improve spam detection in other types of communication media such as emails, social network systems, and online reviews. These features will also help to improve mobile users' aware of spam and their knowledge on how to detect spams in SMS.

There are some interesting future research issues such as incorporating dynamic features by tracking the usages of different words over time and testing the generality of the proposed features in other communication media such as online review and social networks. In addition, there is space for further improving the performance of spam detection.