.

Content Based SMS Spam Detection

REPORT FOR NLP

DELHI TECHNOLOGICAL UNIVERSITY

Submitted By
Ravi Kumar Verma (2K15/SE/055)
Shubham Gupta (2K15/SE/069)
Yogesh Saini (2K15/SE/095)
Aship Libang(2K15/SE/501)

Submitted to:
Ms. Minni Jain

.

<u>INDEX OF CONTENTS</u>

.

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.
• SMSes 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

UCI Machine Learning Repository has a collection of sms messages – SMS Spam Dataset. It contains:
• A collection of 425 spam messages from Grumbletext web site.
• A subset of 3375 ham messages from the NUS Sms Corpus
.• A list of 450 ham messages from Caroline Tag's Phd Thesis.
A total of 4827 ham and 747 spam = 5574 messages The dataset consists of one message per line. Each line is prefixed with a ham/spam label separated by a tabspace.

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 = $\{tm_1, \dots, tm_k\}$. 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 tm_i is a spam (A) or non-spam (B) by using a classifier c. The problem is formulated below:

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

To support the classification, we need to first extract a set of n features F = $\{f_1, \dots, f_n\}$ 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.

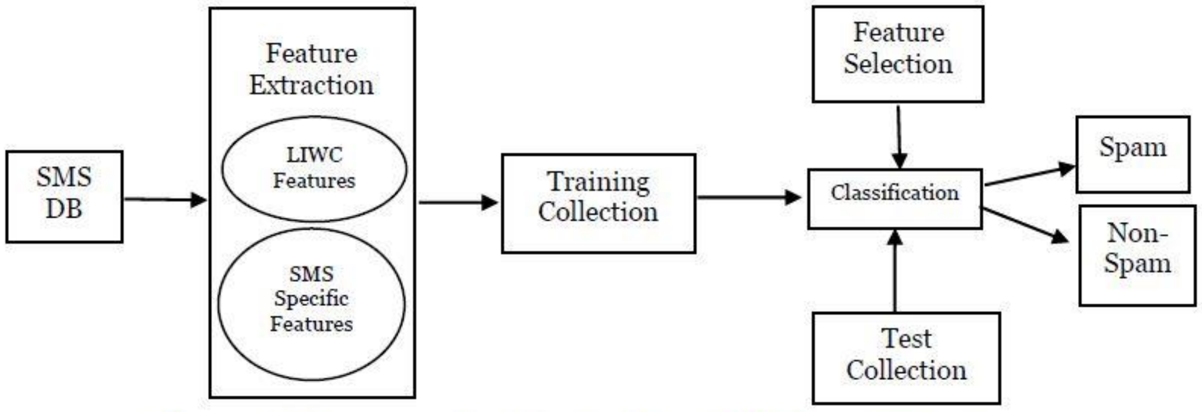


Figure 2: A Framework of Content based SMS Spam Detection

LITERATURE SURVEY/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:

Bayesian is a probabilistic approach that starts with a prior belief, observes some data and then updates that belief. The probability being spam and not spam of a word can be calculated with the frequency of that word in ham and spam messages using the Bayesian algorithm. A prior probability also needs to be assumed in this algorithm which is a shortcoming of this approach.
Support Vector Machines are supervised learning models with associated learning algorithms that analyse data used for classification and regression analysis. If a set of training example containing spam and legitimate SMS is given, then an SVM training algorithm builds a model that can assign new examples into spam and legitimate category. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.
The binary logistic model is used to estimate the probability of a binary response based on one or more predictor (or independent) variables (features). Logistic regression can be used in SMS spam detection on the basis of different feature variables.
A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance of event outcomes. A decision tree can be used to make decision that whether a new message is spam or ham.
The k-nearest neighbours algorithm (k-NN) is a nonparametric method used for classification and regression. The input consists of the k closest training examples in the feature space. The output is a class membership. An object is classified by a majority vote of its neighbours, with the object being assigned to the class most common among its k nearest neighbours.
Random Forests grows many classification trees. To classify a new SMS from an input vector, the algorithm puts the input vector down each of the trees in the forest. Each tree gives a classification, called "votes" for that class. The forest chooses the classification having the most votes

.

PROPOSED METHODOLGY

We describe (1) Feature Extraction ,

(2) Naive Bayes

(3)Support Vector Machines

(4) Ensemble Method

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.

Classification:

After the feature extraction, classification is done initial analysis on the data is done using naive Bayes (NB) algorithm with multinomial event model and laplace smoothing, and based on the results, next steps are determined.

The quality of a classifier depends on the quality of features.

1. FEATURE EXTRACTION

For the initial analysis of the data, each message in dataset is split into tokens of alphabetic characters. Any space, comma, dot, or any special characters are removed from feature space for now, and alphabetic strings are stored as a token as long as they do not have any non-alphabetic characters in between. The effect of abbreviations and misspellings in the messages are ignored, and no word stemming algorithm is used. Additionally, three more tokens are generated based on the number of dollar signs ($), the number of numeric strings, and the overall number of characters in the message. The intuition behind entering the length of message as a feature is that the cost of sending a text message is the same as long as it is contained below 160 characters, so marketers would prefer to use most of the space available to them as long as it doesn’t exceed the

.

limit. For the initial analysis of data, we have used the multinomial event model with laplace smoothing.

S C = false negative cases / no. of spams

BH = false positive cases / no. of spams

SMS SPECIFIC FEATURES

Some of the SMS specific features are as follows :

- Capital Words (CW)
- Spam Words (SW)
- SMS Segments (S)
- SMS Frequency
- Unique Words (UW)
- URL
- Using word "Call"

2. NAIVE BAYES

NB algorithm is applied to the final extracted features. The speed and simplicity along with high accuracy of this algorithm makes it a desirable classifier for spam detection problems. In the context of naive Bayes algorithm with multinomial event model, entering the feature of length of the message corresponds to assuming an independent Bernoulli variable for writing each character in the text message in spam or ham messages.
Applying naive Bayes with multinomial event model and laplace smoothing to the dataset and using 10-fold cross validation results in 1.12% overall error, 94.5% of SC, and 0.51% of BH. Using the data priors and applying Bayesian naive Bayes with same event model will decrease SC (93.7%) and BH (0.44%) by a small margin, but overall error will stay the same. Bayesian model improves the algorithm in case of high variance.
. The errors for different datasets are produced using cross validation with 70% of the samples as the training set.The test set error and training set error are close to each other and in the acceptable range, and it implies no overfitting in the model.

3. SUPPORT VECTOR MACHINES

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.While the overall training set error of the model is far less than error rate for naive Bayes, the test set error is well above that rate. This characteristic shows the model might be suffering from high variance or overfitting on the data. One option we can explore in this case is reducing the number of

.

features. However, the simulation results show degradation in performance after this reduction. For instance, choosing 800 best features based on MI with the labels and training SVM with linear kernel on the result yields to 1.53% overall error, 91.5% SC, and 0.53% BH. While applying SVM with different kernels increases the complexity of the model and subsequently the running time of training the model on data, the results show no benefit compared to the multinomial naive Bayes algorithm in terms of accuracy..

4. k- NEAREST NEIGHBOURS (KNN)

k-nearest neighbour can be applied to the classification problems as a simple instance-based learning algorithm. In this method, the label for a test sample is predicted based on the majority vote of its k nearest neighbours.

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique.

Distance functions

Euclidean

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan

$$\sum_{i=1}^{k}|x_i - y_i|$$

Minkowski

$$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$$

A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If K = 1, then the case is simply assigned to the class of its nearest neighbor. It should also be noted that all three distance measures are only valid for continuous variables. In the instance of categorical variables the Hamming distance must be used. It also brings up the issue of standardization of the numerical variables between 0 and 1 when there is a mixture of numerical and categorical variables in the dataset.
Choosing the optimal value for K is best done by first inspecting the data. In general, a large K value is more precise as it reduces the overall noise but there is no guarantee. Cross-validation is another way to retrospectively determine a good K value by using an independent dataset to validate the K value. Historically, the optimal K for most datasets has been between 3-10. That produces much better results than 1NN.

5. ENSEMBLE METHODS

In this section, an ensemble learning algorithm named random forests is applied to data. Ensemble learning methods combine several models trained with a given learning algorithm to improve robustness and generalization compared to single models. They can be separated into two subcategories, averaging methods and boosting methods. Averaging methods build multiple models independently, but the overall prediction is the average of single models trained. This helps in reducing the variance term in error. On the other hand, boosting methods build models

.

sequentially and generate a powerful ensemble, which is the combination of several weak models.

❖ RANDOM FOREST

Random forests is an averaging ensemble method for classification. The ensemble is a combination of decision trees built from a bootstrap sample from training set. Additionally, in building the decision tree, the split which is chosen when splitting a node is the best split only among a random set of features. This will increase the bias of a single model, but the averaging reduces the variance and can compensate for increase in bias too. Consequently, a better model is built. In this work, the implementation of random forests in scikitlearn python library is used, which averages the probabilistic predictions. Two number of estimators are simulated for this method. With 10 estimators, the overall error is 2.16%, SC is 87.7 %, and BH is 0.73%. Using 100 estimators will result in overall error of 1.41 %, SC of 92.2 %, and BH of 0.51 %. We observe that comparing to the naive Bayes algorithm, although the complexity of the model is increased, yet the performance does not show any improvement.

❖ ADABOOST

Adaboost is a boosting ensemble method which sequentially builds classifiers that are modified in favor of misclassified instances by previous classifiers . The classifiers it uses can be as weak as only slightly better than random guessing, and they will still improve the final model. This method can be used in conjunction with other methods to improve the final ensemble model. In each iteration of Adaboost, certain weights are applied to training samples. These weights are distributed uniformly before first iteration. Then after each iteration, weights for misclassified labels by current model are increased, and weights for correctly classified samples are decreased. This means the new predictor focuses on weaknesses of previous classifier. We tried the implementation of Adaboost with decision trees using scikit-learn library. Like Random Forests, although the complexity is much higher, naive Bayes algorithm still beats Aadaboost with decision trees in terms of performance.

.

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. Only the alphabetical tokens are stored. Tokens with punctuation marks, special characters are ignored.
- No word stemming is done here.
- Three extra attributes per message are also stored – the number of dollar signs, the number of numeric strings, the length of the message.(Message length is a great feature because, spam generators aim to make of the message length usable without wastage.)
- Now, we remove tokens which occur less than 5 times and more than 500 times. This removes noisy terms and helps normalize the features.
- This phase generates about 1502 + 3 = 1505 features.

Feature vectors:

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

Classification:

- The Naïve Bayes classifier is used here.
- Multinomial event model of NB is used as it is useful for feature vectors with word counts.
- Laplace Smoothing is used in NB to handle cases where the feature vectors word counts are zero.

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
- 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’.

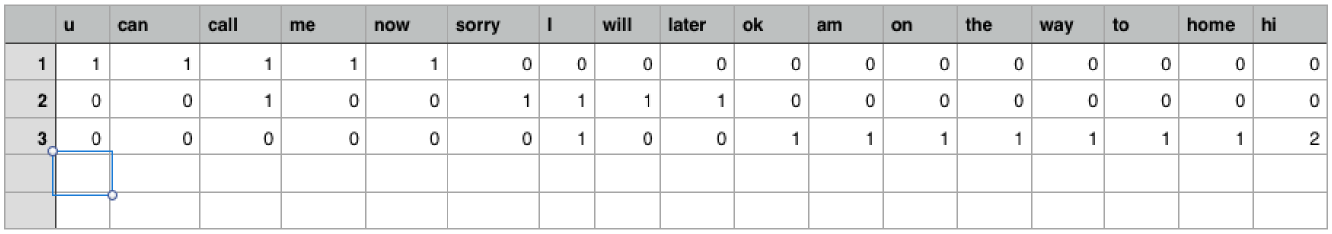
Vectorizers

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.

Example how to work CountVectorizer.
It converts a collection of text documents to a matrix of token of unique words counts. It finds all unique words in text-set and makes one vector. After, it converts each text to an array of unique words counts. And as result, we have one vector of unique words and many arrays with many count of zero. Example for data-set with 3 messages:

1. U can call me now
2. Sorry, I will call later
3. Ok i am on the way to home hi hi



	u	can	call	me	now	sorry	I	will	later	ok	am	on	the	way	to	home	hi
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	2

Another is **TfidfVectorizer**. It is an implementation of Term Frequency times Inverse Document Frequency algorithm. You can read about it in official documentation.

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.

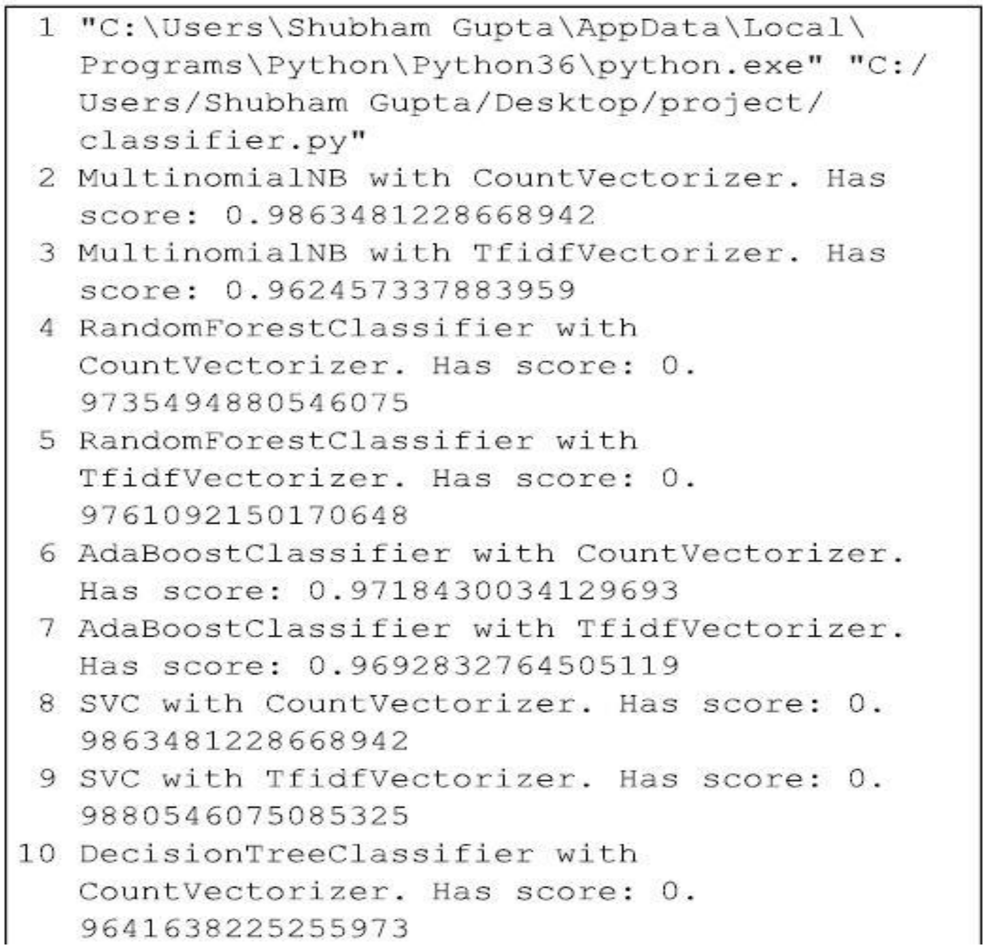
.

RESULTS

• The training of the classifier is very fast, < 1Second on a Core i5 with 8GB DDR3 RAM.

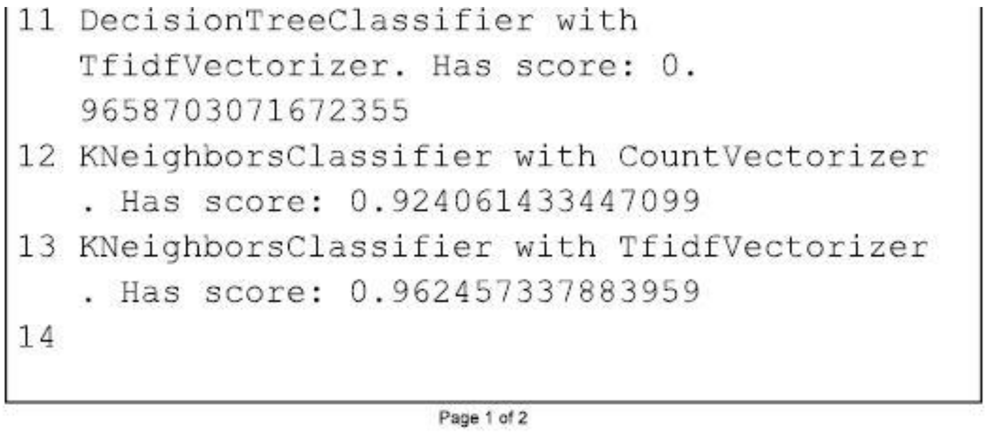
• Testing takes far lesser time.

• The accuracy computed by testing the classifier on the test data is shown in the following output of our implementation.



```
"C:\Users\Shubham Gupta\AppData\Local\
Programs\Python\Python36\python.exe" "C:/
Users/Shubham Gupta/Desktop/project/
classifier.py"
MultinomialNB with CountVectorizer. Has
score: 0.9863481228668942
MultinomialNB with TfidfVectorizer. Has
score: 0.962457337883959
RandomForestClassifier with
CountVectorizer. Has score: 0.
9735494880546075
RandomForestClassifier with
TfidfVectorizer. Has score: 0.
9761092150170648
AdaBoostClassifier with CountVectorizer.
Has score: 0.9718430034129693
AdaBoostClassifier with TfidfVectorizer.
Has score: 0.9692832764505119
SVC with CountVectorizer. Has score: 0.
9863481228668942
SVC with TfidfVectorizer. Has score: 0.
9880546075085325
DecisionTreeClassifier with
CountVectorizer. Has score: 0.
9641638225255973
```

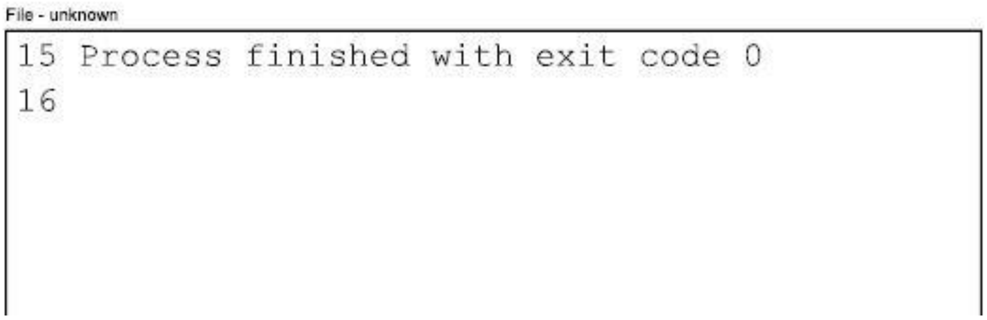
.



```
DecisionTreeClassifier with
TfidfVectorizer. Has score: 0.
9658703071672355
KNeighborsClassifier with CountVectorizer
. Has score: 0.924061433447099
KNeighborsClassifier with TfidfVectorizer
. Has score: 0.962457337883959
```

Page 1 of 2

File - unknown



```
Process finished with exit code 0
```

- Final results of different classifiers applied to SMS spam dataset

MODEL	Accuracy with Count vectorizer(in %)	Accuracy with TfidVectorizer(in %)
Multinomial NB	98.635%	96.246%
Random Forest	97.355%	97.611%
AdaBoost	97.184%	96.928%
SVM	98.635%	98.805%
k-Nearest Neighbour	92.406%	96.246%
Decision Tree	96.416%	96.587%

• Some of those messages were repeated in that corpus many times.

• There were false positives(ham labelled as spam) and false negatives(spam labelled as ham). But occurrence of false positives was greater than false negatives.

.

• False positives > False Negatives was because the classifier associated the label 'spam' with features – length of message, numeric terms and word features such as 'free', 'call', 'urgent' etc.

• This scenario can get improved accuracy if we introduce some features which use ratio of capital words to small words, ratio of spam words to normal words, consider more different currency symbols, number of URLs in the message.

Advantages and Disadvantages of Used Technique

The advantages and drawbacks of the approaches are mentioned in the table. From the table we can say that, content based filtering is more convenient than other non-content based and server side algorithms. Server side algorithms suffer from implementation complexity. Feature selection is also an important task for machine learning algorithms to work correctly. One important drawback is, some approaches do not use classification algorithm only focusing on user generated features. Classification algorithm is necessary for gaining better accuracy.

Sno.	Advantages	Disadvantages
1.	Combination of machine learning algorithms with user generated features	Users need to select features manually
2.	Classification based on two algorithms	No classification algorithm is used
3.	Combination of client and server side algorithms	Challenge- response technique suffers from server side traffic and user interaction problems
4.	Accurate as Naïve Bayesian with necessary feature extraction	Complex implementation
5.	Used a weighting Mechanism to reduce false negatives	Suffers from implementation complexity
6.	Demonstrates the need of spam filtering in spite of having established email spam filtering	Extensive feature engineering is needed for better accuracy
7.	Lightweight and focuses on runtime	Server side, complex and suffers from updating issues

.

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. Bayesian algorithm also suffers from traditional threshold selection problem, dataset dependency, assuming prior probability.* Despite having those shortcomings, Bayesian 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.

.

REFERENCES

1) SMS Spam Detection using Machine Learning Approach, Houshmand Shirani-Mehr, http://cs229.stanford.edu/proj2013/ShiraniMehr-SMSSpamDetectionUsingMachineLearningApproach.pdf

2) SMS Spam Collection Data Set from UCI Machine Learning Repository,

http://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection

3) Improving Static SMS Spam Detection by Using New Content-based Features , Amir Karami, Lina Zhou University of Maryland Baltimore County https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1205&context=amcis201

4) Almeida, T., Hidalgo, J.M.G.m., and Silva, T.P. "Towards SMS Spam Filtering: Results under a New Dataset," *International Journal of Information Security Science (2:1) 2013, pp 1-18.*

5) A Systematic Literature Review on SMS Spam Detection Techniques Lutfun Nahar Lota ,B M Mainul Hossain

http://www.mecs-press.org/ijitcs/ijitcs-v9-n7/IJITCS-V9-N7-5.pdf

6) SMS Spam Detection Using Simple Message Content Features Dr. Ghulam Mujtaba, Majid Yasin

https://pdfs.semanticscholar.org/ee96/4e647c07fcc5793d937a561847bac807c1cd.pdf

7)Ting, K.M. 1998. Inducing cost-sensitive trees via instance weighting. En Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery, 139-147.

8) Witten, I.H., E. Frank. 1999. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann.

9)Xiang,Y., Chowdhury, M., Ali, S. Filtering Mobile spam by Support Vector Machine. Proceedings of CSITeA-04 , ISCA Press, December 27-29, 2004.

10) Yang, Y. 1999. An evaluation of statistical approaches to text categorization. Information Retrieval, 1(1/2):69-90.

11)Yang, Y., J.O. Pedersen. 1997. A comparative study on feature selection in text categorization. En Proceedings of the 14th International Conference on Machine Learning.

12) Bratko, A, B. Filipic. Spam Filtering using Character-level Markov Models: Experiments for the TREC 2005 Spam Track. Proceedings of the 2005 Text Retrieval Conference,