SMS SPAM FILTER

Tushar Bhandari, Kosha Shah, Sankalpa Kulkarni (The Smart Miners)

Department of Computer Science

Northeastern University

Boston, MA 02115

{tushar27, kosha, sankalpa}@ccs.neu.edu

Yizhou Sun

Assistant Professor

Northeastern University

Boston, MA 02115

yzsun@ccs.neu.edu

Abstract

In the current age of mobile phones, Short Message Service (SMS) has become a popular form of text communication. This service is misused a lot by increasing number of spam messages. We are building a data mining system which will correctly classify spam messages.

It has been observed that Bayesian filters and Support Vector Machine are effective techniques in classifying spam messages. In this paper we have achieved text based spam classification using Bayesian and Support Vector Machine with n-grams as features. We are comparing these two supervised classification models. The dataset obtained has been divided to form the training and test dataset. The learned algorithm is then used to predict if a new incoming message is a spam or not. The comparison evaluation indicates that the Support Vector Machine performs better than the Bayesian Classifier.

Introduction

SMS Spam is an unwanted text message usually sent in bulk without the consent of the recipient. With increase in use of text messaging, there has been a growth in the number of unwanted advertisements, some of them requesting for personal information with intent of fraud.

In some cases, the recipient might be charged for incoming messages including spam which may be annoying for the recipient. Sometimes, these spam messages may also contain malware, spyware which may infect the user's device causing crucial data loss. To avoid this inconvenience caused to the recipient, it is important to filter these spam messages

The paper is organized as follows. The description of problem is summarized in next section. Information and preprocessing performed on data is described in the following section. The section after that explains the different approaches used for implementing the classifiers. Evaluation and comparison of the models are done in the last section.

Problem Definition and Formalization

The problem involves classifying a message containing stream of words into spam [1] or ham [2]. As problem deals with short messages, there are high frequency words which are prevalent in spam messages. The problem involved is to identify such high frequency words and then based on their frequency count, classify the message as a spam or ham. For example consider the messages: "Free prize in 2 a wkly comp to win FA Cup final tkts", "Winner for a prize…" in these two messages the frequency of words such as "prize", "winner" and "free" provide more weightage for the message to be classified as a spam. Thus we need to identify such patterns while performing classification.

The problem can be subdivided into the following tasks: Feature Selection on the data set, identification of efficient classification algorithms, training the classifier using training dataset, testing the learned classifier using test data and evaluation of the classifier model.

Data Description and Preprocessing

In any data mining related work, reliable and authentic dataset is an important factor in the success of data mining tasks. The dataset used is obtained from the UCI SMS Spam Collection Dataset [3]. This dataset consists of 5574 instances of which 86.6% are ham messages and 13.4% are spam messages. The dataset contains a collection of short messages including words, special characters and numbers.

Message	Amount	%
Hams	4827	86.6
Spams	747	13.4
Total	5574	100

Since the dataset is supervised it contains the class labels for each tuple. The dataset contains two attributes: The class label and the actual message. The class labels are of two types, i.e. spam and ham. The actual message consists of words i.e. letters and numbers, and non-word characters which include special characters and spaces.

The examples of dataset tuples are as follows:

Class-Label	Message
Ham	What you doing? how are you?
ham	Ok lar... Joking wif u oni...
spam	FreeMsg: Txt: CALL to No: 86888 & claim your reward of 3 hours talk time to use from your phone now! subscribe6GBP/ mnth inc 3hrs 16 stop?txtStop
spam	URGENT! Your Mobile No 07808726822 was awarded a L2,000 Bonus Caller Prize on 02/09/03! This is our 2nd attempt to contact YOU! Call 0871-872-9758 BOX95QU

```
@RELATION spam

@attribute message string
@attribute classname {ham, spam}

@data

'Ok lar Joking wif u oni ',ham
'U dun say so early hor U c already then say ',ham
'Nah I don t think he goes to usf he lives around here though ',ham
'Even my brother is not like to speak with me They treat me like aids patent ',ham
```

The available data is processed to a valid format that can be read and used to train the algorithm. For Naïve Bayesian Classification, we converted the dataset into a text file with each tuple occupying new line in the file and each line has a class label with the actual message. While reading the data from the text file we ignore the special characters and spaces and consider only the words also called as valid tokens for further classification.

For N-grams with Support Vector Machine, we converted the dataset into a valid .arff file format (Attribute-Relation File Format). This file is an ASCII text file format with two distinct sections – Header and Data. The header section contains the information about the meta data of the dataset, like name of the dataset, different attributes and their type shared between the instances, etc. The data section contains the actual data.

The token statistics of our dataset is shown in the following table

Hams	63,632
Spams	17,543
Total	81,175
Avg per Message	14.56
Avg in Hams	13.18
Avg in Spams	23.48

Methods Description

We have used two classification algorithms: Naïve Bayesian classification and Support Vector Machine.

Naïve Bayesian Classification:

It is a statistical technique of short message filtering. It works by classification of tokens based on their frequency of occurrence in spam or ham messages. It gives low false positive spam detection rates.

Bayes theorem [4] states that:

$$P(h|X) = P(X|h)\ P(h)/P(X)$$

where X is a data sample

Let h be a hypothesis that X belongs to class C

P(h): the initial probability

P(X|h): the probability of observing the sample X, given that the hypothesis holds

P(X): marginal probability that sample data is observed

P(h|X): the probability that hypothesis holds given the observed data sample X.

Learning Phase: First, we need to read the data from the dataset and preprocess it into valid tokens as mentioned in the data preprocessing step. These valid tokens are then processed into its respective hashmap, spam or ham hashmap. Both these hashmaps contain tokens and their respective frequency of occurrence in the ham or spam message. We have two more hashmaps that store unique words in the dataset and their probability of being in a spam or ham message. These conditional probabilities are calculated by applying Laplace smoothing.

$$P(W_i|S) = (\#(W_i|S) + 1) / (N + |V|)$$

Where N is total number of words in spam tuples in dataset and V is the unique words in the dataset. Similarly, conditional probability for hams is calculated.

Testing Phase: For each tuple in the test set, we take into account each word and its probability of occurrence in ham or spam message and depending on which is higher, we assign the resulting class label as ham or spam.

$$P(S|W) \alpha P(W|S)*P(S)$$

Where $P(W|S) = \prod_{i=1}^{n} \mathbf{P}(\boldsymbol{Wi}|\boldsymbol{S}) * \mathbf{P}(\boldsymbol{S})$

In the above equation, many conditional probabilities are multiplied which may result in a floating point underflow.

Hence, it is better to compute logarithms of probabilities. The class with the highest log probability is still the most probable. [5]

$$P(W|S) = \sum_{i=1}^{n} (\log P(W_i|S) * P(S))$$

Similarly probability of each word occurring in ham and its total conditional probability are calculated as above. Then the one with higher probability is the class that we assign to the message.

Support Vector Machine:

It is a supervised learning model which is used to analyze and recognize data. It is a linear or non-linear probabilistic model. SVM represents data in space and separates the data by assigning them to different categories which are formed by separating them with a plane which distinctly classifies them. SVM performs non-linear classification by mapping the data points into high dimensional space. The efficiency of SVM to classify text based content is high.

The approach we have used is to make a SVM learned classifier and then use the learned classifier on test dataset to predict the results. For training the classifier we have first extracted the words from messages and converted those words to a string vector. The string vectors formed are of single words and the frequency of the single word vectors is used to train the classifier. We have used WEKA [6] libraries to train the classifier. The implementation using WEKA involves importing the WEKA libraries into Java project and then using those libraries to train the classifier. WEKA requires the data to be present in .arff file format. We have dataset in string format, hence we have converted the dataset to "string to word vector" for reading purpose. Now the read dataset is applied to the filtered classifier. The classifier is tuned to work with string vector options. For training the classifier we have created an object of FilteredClassifier and applied it to the string vector object. The trained classifier is then applied to the test dataset. The classifier then predicts the class labels for each of the tuple in the test dataset. Once the prediction is done the trained classifier is used to evaluate the efficiency of the classifier.

Support Vector Machine with N-grams:

In support vector machine implementation we have considered just single word feature for creating string vectors. It has been observed that certain words when considered individually have higher frequency in spam messages, and hence affect the probability of the entire message to be classified as spam. However these same words when combined with other words may occur less frequently. Hence to accurately train the classifier it is preferable to extract n-gram features from the dataset and form the string vectors using these n-gram features.

In our implementation, we have extracted the n-gram features by using the NGramTokenizer and specifying the size of the tokens. These tokens extracted are then converted into the string to word vector. These string to word vectors are then used to train the support vector classifier. Then an object of the string to word vectors is applied to the filtered classifier object created using WEKA. The object is then applied over the training set to form a learned classifier. The learned classifier is then applied to test dataset which then predicts the label for each of the tuples in the test data set. Now based on the prediction we calculate the accuracy of the classifier. The detailed evaluation of both the methods is provided in the evaluation and comparison section.

WEKA

WEKA[6] is a tool which facilitates classification and clustering of data. It implements different classification algorithms like Bayesian, support vector machine, J48, LMT and many more classification algorithms. It contains a number of other functionalities like division of training and testing data based on different percentages.

Experiments design and Evaluation

For Naïve Bayesian, we have two input files namely "bayesian_train.txt" and "bayesian_test.txt" to train the model and to test it respectively. The output is an analysis of each tuple being classified as ham or spam and its comparison with actual label. For Support Vector Machine, we have used two input files namely "train.arff" and "test.arff". The output is shown in two different files one for the actual predictions of the test set and the other for the evaluation of the model.

Evaluation:

Some measures that we have considered while evaluating our model include Accuracy, Precision, Recall, etc. These measures prove to be very useful in evaluating the system.

Naïve Bayesian model:

Evaluation of our model for Naïve Bayesian classifier is as follows:

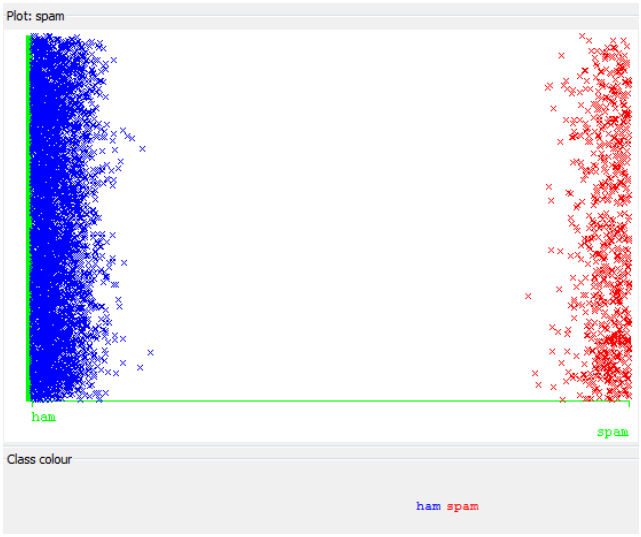
Accuracy	Sensitivity = Recall	Precision	Specificity
0.987	0.93	0.9612	0.994

Accuracy is the degree of closeness of the predicted value to its actual value. i.e. 98.7% accuracy indicates the percentage of tuples correctly classified by the trained classifier. Sensitivity (recall) indicates the percentage of positive tuples which are correctly classified as positive and our algorithm gives a true positive rate of 93%. Precision rate of 96.12% states that the classifier returned more relevant da-

ta than the irrelevant data. Specificity measures the proportion of negatives correctly classified, hence a 99.4% specificity indicates the same.

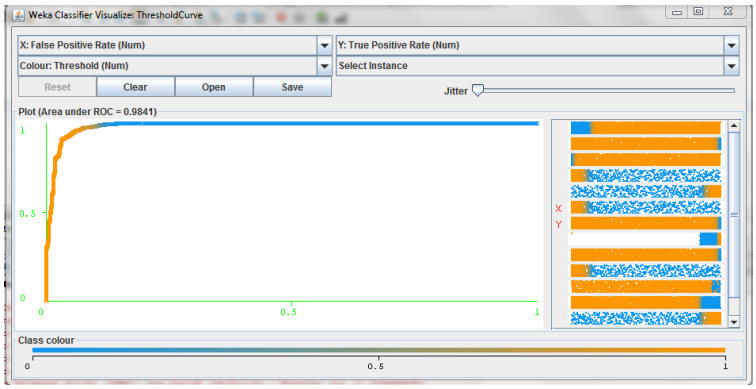
Observations: The Naïve Bayesian classifier can be trained very effectively in a supervised learning model. Bayesian model assumes that the probability of tokens taken for learning is independent of others tokens. Also it has been observed that because of these independent probabilities some words have higher frequency of occurrence in spam as well as ham e.g. common words like "the", "are" etc. are equally likely to occur in both ham and spam messages and hence affect the overall probability of the message. Whereas words like "free" and "call" have higher occurrences in spam messages and hence the messages containing these words are more likely to be predicted as spam. But these words when considered in combination with other words may not have a higher probability of occurrence.

Find below the results of the classification performed by Naïve Bayesian in graphical form:



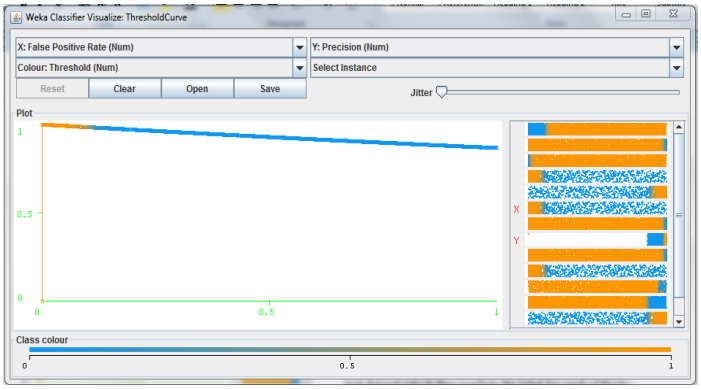
We have also observed that certain words such as "I" and "You" also have major impact on the result depending on the context they are used in. For example a message such as "You have free message" is correctly classified as spam whereas a similar message "I have free message" is correctly classified to be a ham by naïve bayes classifier.

The graphical evaluation of the Bayesian filter is given below:



The above graph displays the receiver operating characteristics; the classifier has an area under ROC of 0.9841. The area under ROC is the ability of the classifier to correctly classify messages that are spam and not spam. It can be seen that as the curve is away from the random guessing line the accuracy is found to be good.

The precision of Bayesian can be represented as below:



Support Vector Machine:

The evaluation of the support vector machine in terms of accuracy, TP rate, FP rate, Precision, Recall, ROC Area etc. are displayed in the table below:

	TP Rate	FP - Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area
Ham	0.99	0.074	0.989	0.994	0.992	0.933	0.96	0.989
Spam	0.926	0.006	0.958	0.926	0.942	0.933	0.96	0.896
Avg	0.985	0.065	0.985	0.985	0.98	0.933	0.96	0.977

The confusion matrix for SVM is as below:

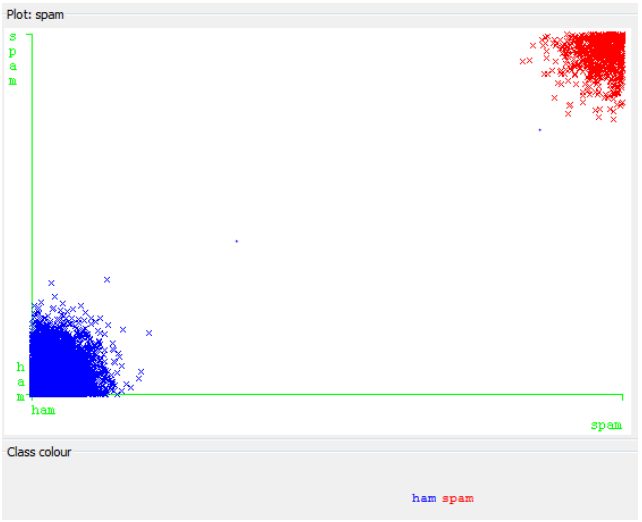
```
=== Confusion Matrix ===

   a   b   <-- classified as
 832   5 |   a = ham
   9 113 |   b = spam
```

Also the detailed analysis of other measures is as given below:

Correctly Classified Instances	945	98.54
Incorrectly Classified Instances	14	1.4599
Kappa statistic	0.933	
Mean absolute error	0.0146	
Root mean squared error	0.1208	
Relative absolute error	6.4054%	
Root relative squared error	36.25%	
Coverage of cases (0.95 level)	98.54%	
Mean rel. region size (0.95 level)	50%	
Total Number of Instances	959	

Observation: From the above observation it can be seen that accuracy of SVM is found to be 98.54% which are correctly classified instances. The difference in the actual value and the estimated value is the mean absolute error; it has been observed that for SVM it has a very low mean absolute error of 0.0011, also the relative absolute error for SVM is 0.18%.

Our F value is 0.99 and higher the F score means that feature is a good discriminator of ham and spam. While evaluating SVM without feature extraction it has been seen that SVM model projects each of the words in a high dimensional space and for each word generates a support vector number. Then based on a threshold determines the support vectors that need to be considered for further analysis. Then based on these support vectors identified, the model creates a separating plane and classifies the messages.



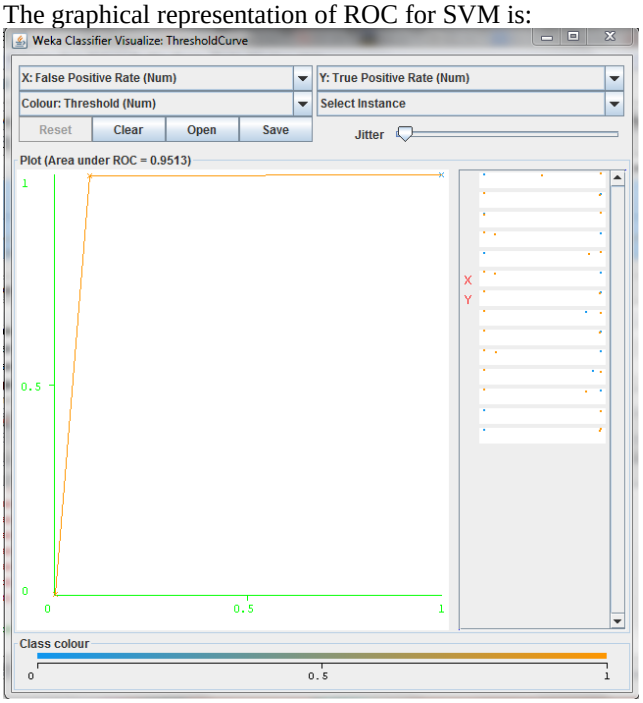
As shown in the above figure, SVM classifies the messages and for True positives and true negatives i.e. spams classified as spams and hams classified as hams it separates these tuples with a separating plane.

An example of the words converted to normalized support vectors by the algorithm with unigram features is displayed below:

+ -0.0443 * (normalized) BABE
+ -0.0725 * (normalized) BSLVYL
+ 0 * (normalized) BUT
+ 0.4831 * (normalized) Babe
+ -0.015 * (normalized) Be

As can be seen each word is given a weight which is used to determine the support vectors and hence used for classification.

The graphical representation of ROC for SVM is:



It can be seen that the ROC for SVM clearly shows that the area under ROC is 95%.

Support Vector Machine with N-grams:

Support vector machine does not assume the probabilities to be independent. Also to increase the accuracy of the algorithm, feature extraction can be done on the dataset and then the algorithm can be applied. Feature extraction involves grouping the words and then determining their frequency. Thus we can group the words for N-grams[7] i.e. continuous sequence of n items from a given sequence and then generate support vectors for these N-grams. The evaluation with bigram features is as below

	TP Rate	FP - Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area
ham	0.994	0.221	0.99	0.994	0.981	0.842	0.886	0.968
Spam	0.779	0.006	1	0.779	0.856	0.842	0.886	0.768
Avg	0.967	0.194	0.966	0.967	0.965	0.84	0.886	0.943

The confusion matrix for the same is as below:

```
=== Confusion Matrix ===

   a   b   <-- classified as
 832   5 |   a = ham
  27  95 |   b = spam
```

Also the evaluation based on other measured is as below:

Correctly Classified Instances	945	96.67%
Incorrectly Classified Instances	32	3.33%
Kappa statistic	0.837	
Mean absolute error	0.0334	
Root mean squared error	0.1827	
Relative absolute error	14.64%	
Root relative squared error	54.80%	
Coverage of cases (0.95 level)	96.66%	
Mean rel. region size (0.95 level)	50%	
Total Number of Instances	959	

As can be seen from the above statistics we find that accuracy of SVM with N-Grams is 96.67% i.e. 96.67% of the tuples were correctly classified whereas 0.9091% were incorrectly classified. It can be seen that the coverage obtained is 96.0909% and the absolute error is 0.0091% .
Observation: It can be seen that the accuracy of the system is approximately same as that for unigram but from the area under ROC it can be seen that bigram achieves better performance than the unigram. A further analysis of unigram, bigram and trigram features it can be observed that trigrams performance is the best. Also, if the number of grams is further increased, the accuracy decreases as the dependency between the words increase which becomes less probable in the fixed size of the dataset.

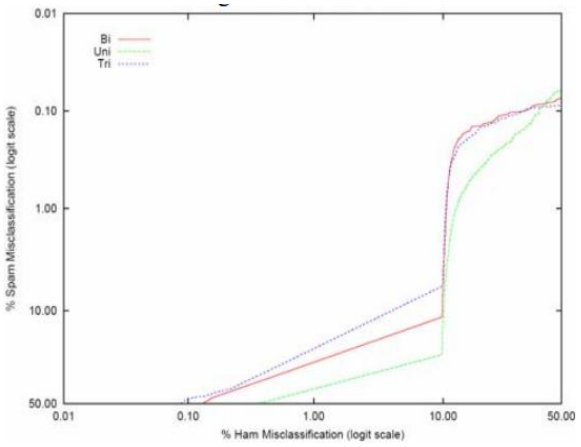


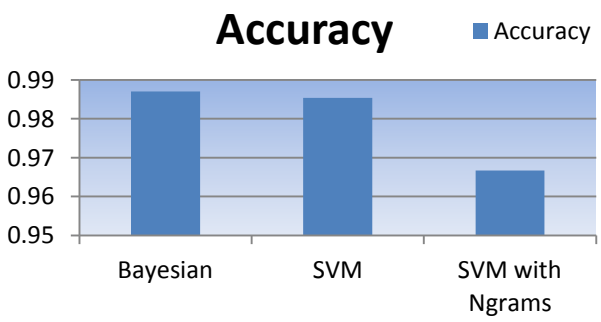
Figure for ROC of unigram, bigram and trigram

Comparison between Naïve Bayesian and SVM:

The comparison of the evaluation measures is done in the following table.

	Accuracy	Recall	Precision
Bayesian	0.987	0.93	0.9612
SVM	0.9854	0.92	0.999
SVM with N-grams	0.9667	0.994	0.991

The Bayesian classifier does not consider the dependencies among various features. This results in less computational cost but also results in less accuracy. This is because there exists dependencies among the features in real world scenarios. Whereas, SVM considers the distances between different features to classify the tuples. This distance is based on the hyper-plane it creates. On the other hand, SVM with n-grams considers the dependencies on different features.



As can be seen, the accuracy for Naive Bayesian is least. This is because a word i.e. our feature has more frequency in ham or spam by itself. But when combined with other features, it gives actual dependencies and calculates the frequency based on that. Hence the bigram algorithm gives better prediction than Naive Bayesian.

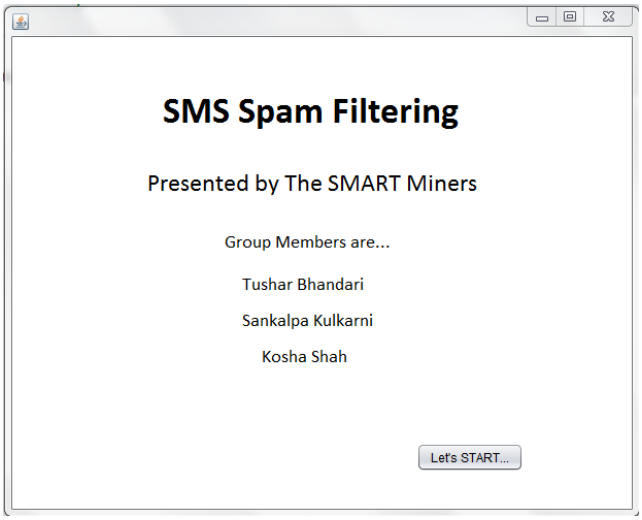
The other observation is that if the max value of n is 5 and minimum is 2 in n-grams algorithm then the accuracy falls down to 94%. This is because when the value of n is 5, it considers group of consecutive words of size 5 to classify the tuple, but finding the same group of consecutive words in the entire dataset is very less probable when the dataset remains the same. This affects the overall probability and hence it predicts most of the tuples as ham since the probability of ham in training set is more than that of spam.

Also it can be seen that these three algorithms do not differ significantly in their statistical measures.

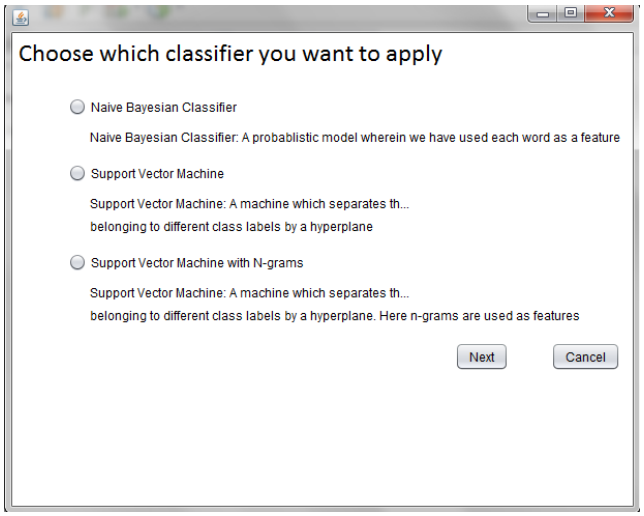
User Interface

JAVA Application:

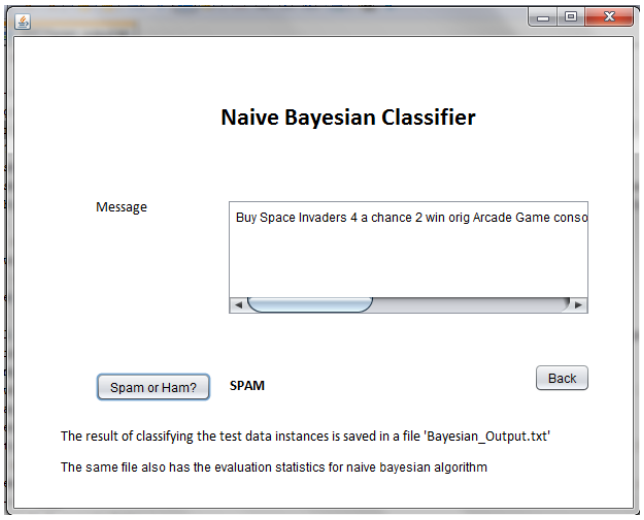
We have developed a user interface to make our data mining application more interactive. When the application is launched a new window appears and gives the introduction of the project and the team members.



After this when the button "Let's START" is clicked it will load a new panel wherein the user can choose the classifier desired. There are three options: Naive Bayesian, SVM and SVM with N-gram. On clicking the next button, the classifier chosen will start its learning process.

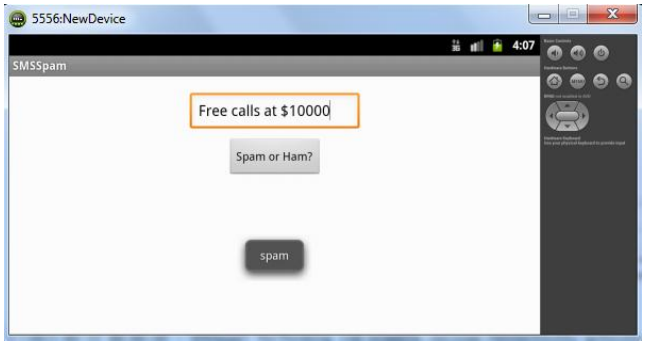


After the learning process completes you will be redirected to the classifier chosen. Here you can input the text message you want to classify.



Mobile Application:

Since we deal with SMS spam filtering, finally a mobile application is what is needed. But since the focus is more on the data mining task, we have prepared a basic android User Interface with a textbox for the user to enter the message. A button called "Spam or Ham" when clicked will display the output as ham or spam. The algorithm used for this demonstration is Naive Bayesian Classification. A screenshot of this user interface is given below



Developing a full fledged robust mobile application for the data mining task that we have performed successfully is left for future work.

Conclusion

Thus we have evaluated two classifiers Naïve Bayesian and SVM with and without feature extraction. Based on the comparison we can conclude that Naïve Bayesian and SVM differ by a very small amount statistically. It can be concluded that Bayesian does not take into consideration dependent tokens whereas SVM does consider dependencies. Also content-based filtering alone is not sufficient for an accuracy above 99% in detecting spam due to large number of false positives.

References

[1]Spam(electronic):http://en.wikipedia.org/wiki/Spam_(electronic)

[2] Ham(electronic):http://wiki.apache.org/spamassassin/Ham

[3]UCI SMS Spam Collection Dataset http://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection

[4] Data Mining Concepts and Techniques – Han and Kamber

[5] NLP Stanford: http://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html

[6] WEKA the University of Waikato:

http://weka.wikispaces.com/Use+WEKA+in+your+Java+code

[7]N-grams: http://en.wikipedia.org/wiki/N-gram