SMS Spam Detection using Machine Learning Approach

Houshmand Shirani-Mehr, hshirani@stanford.edu

***Abstract*—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. In parts of Asia, up to 30% of text messages were spam in 2012. Lack of real databases for SMS spams, short length of messages and limited features, and their informal language are the factors that may cause the established email filtering algorithms to underperform in their classification. In this project, 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. Finally, the results are compared and the best algorithm for spam filtering for text messaging is introduced. Final simulation results using 10-fold cross validation shows the best classifier in this work reduces the overall error rate of best model in original paper citing this dataset by more than half.**

I. Introduction

The mobile phone market has experienced a substantial growth over recent years. In second quarter of 2013, a total of 432.1 million mobile phones have shipped, which shows a 6.0% year over year increase [1]. As the utilization of mobile phone devices has become commonplace, Short Message Service (SMS) has grown into a multi-billion dollars commercial industry [2]. SMS is a text communication platform that allows mobile phone users to exchange short text messages (usually less than 160 seven-bit characters). It is the most widely used data application with an estimated 3.5 billion active users, or about 80% of all mobile phone subscribers at the end of 2010 [3]. As the popularity of the platform has increased, we have seen a surge in the number of unsolicited commercial advertisements sent to mobile phones using text messaging. SMS spam is still not as common as email spam, where in 2010 around 90% of emails was spam, and in North America it is still not a major problem, contributing to less than 1% of text messages exchanged as of December 2012 [4]. However, due to increased popularity in young demographics and the decrease in text messaging charges over the years (in China it now costs less than $0.001 to send a text message), SMS Spam is showing growth, and in 2012 in parts of Asia up to 30% of text messages was spam. In middle east, some of the carriers themselves are responsible for sending out marketing text messages. Additionally, SMS Spam is particularly more irritating than email spams, since in some countries they contribute to a cost for the receiver as well. These factors along with limited availability of mobile phone spam-filtering software makes spam detection for text messages an interesting problem to look into.

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.

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 [6] [9]. 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, and the SMS Spam Corpus v.0.1 Big (1,002 SMS non-spam and 322 spam messages publicly available). 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 [2]. Feature extraction and initial analysis of data is done in MATLAB, then applying different machine learning algorithms is done in python using scikit-learn library.

The project report is organized as follows: Section 2 explains the preprocessing of the data and extraction of features from the main dataset, and explores the result of initial analysis to gain insight. Section 3 explores the application of naive Bayes algorithm to the problem. In Section 4, application of Support Vector Machine algorithm to the classification problem is studied. Section 5 shows the performance of k-nearest neighbor classifier for the data. Section 6 explores application of two ensemble methods, random forests and

Label	Percentage in dataset
Spams	13.40
Hams	86.60

TABLE I
THE DISTRIBUTION OF SPAMS AND NON-SPAMS IN THE DATASET

Adaboost. Finally, Section 6 concludes the report.

II. Feature Extraction & Initial Analysis

As mentioned earlier, our dataset consists of one large text file in which each line corresponds to a text message. Therefore, preprocessing of the data, extraction of features, and tokenization of each message is required. After the feature extraction, an 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.

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. Extracting tokens for all messages in the dataset will result in 7,789 features. However, not all of these features are useful in the classification. Going through the extracted tokens, we removed the ones with less than five and more than 500 times frequency in the dataset, since those tokens are either too rare or too common, and do not contribute to the content of the messages. These two thresholds are set by exploring different values and checking the performance of NB classification algorithm on results. Finally, the remaining tokens result in 1,552 features.

Figure 1 shows the result of applying NB algorithm to the dataset using extracted features with different training set sizes. The performance in learning curve is evaluated by splitting the dataset into 70% training set and 30% test set. As shown in the figure, the NB algorithm shows good overall accuracy. The 10-fold cross validation for this algorithm on current data shows 1.5% overall error, 93% of spams caught (SC), and 0.74% of blocked hams (BH).

$$SC = \frac{False\ negative\ cases}{Number\ of\ Spams} \quad (1)$$

$$BH = \frac{False\ positive\ cases}{Number\ of\ Hams} \quad (2)$$

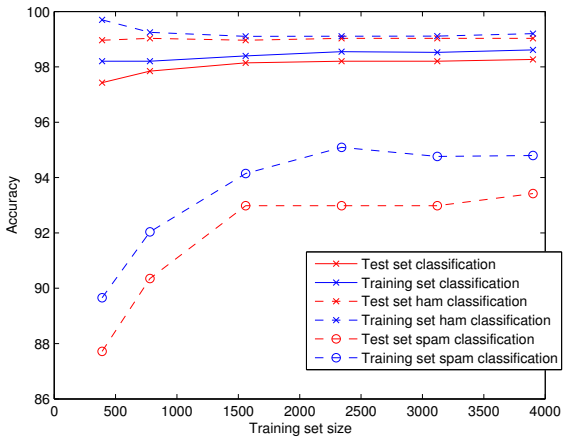

Fig. 1. Learning curve for naive Bayes algorithm applied to the dataset and evaluated using cross validation (30% of initial dataset is our test set

From the analysis of results, we notice that the length of the text message (number of characters used) is a very good feature for the classification of spams. Sorting features based on their mutual information (MI) criteria shows that this feature has the highest MI with target labels. Additionally, going through the misclassified samples, we notice that text messages with length below a certain threshold are usually hams, yet because of the tokens corresponding to the alphabetic words or numeric strings in the message they might be classified as spams.

By looking at the learning curve, we see that once the NB is trained on features extracted, the training set error and test set error are close to each other. Therefore, we do not have a problem of high variance, and gathering more data may not result in much improvement in the performance of the learning algorithm. As the result, we should try reducing bias to improve this classifier. This means adding more meaningful features to the list of tokens can decrease the error rate, and is the option that is explored next.

We update the features list by adding five different flags to gathered tokens. These flags determine if the length of the message in characters is $\leq$ 40, $\leq$ 60, $\leq$ 80, $\leq$ 120, and $\leq$ 160. Additionally, we add the string of non-alphabetic characters and symbols excluding dot, comma, question mark, and exclamation mark to our tokens. For instance, a string of characters such as "://" can imply the presence of a web address, or a character such as "@" can imply the presence of an email address in the message. The resulted features are again filtered if they are too rare or too common in the dataset. Finally, we end up with a list of 1582 features.

III. Naive Bayes

In this section, 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

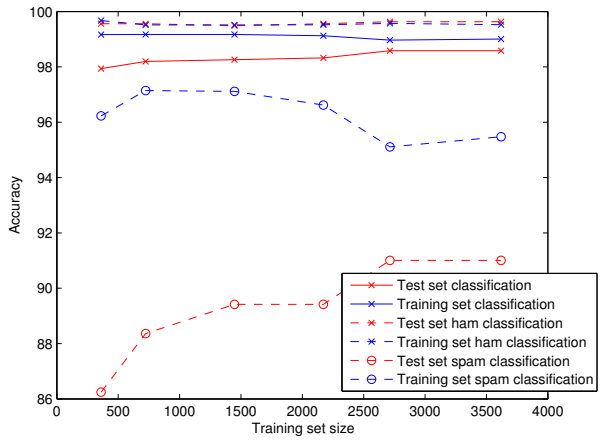

Fig. 2. Learning curve for multinomial NB algorithm applied to final features

with multinomial event model, entering the feature of length of the message corresponds to assuming an independent Bernouli 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. This is what we would expect, since Bayesian model improves the algorithm in case of high variance. Figure 2 shows the learning curve for mutinomial NB applied on the final features extracted from dataset. The errors for different datasets in this plot are produced using cross validation with 70% of the samples as the training set. As it is shown in the figure, 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. To reduce the bias and improve the accuracy of algorithm, we can explore other more sophisticated models in following sections.

IV. Support Vector Machines

In this section, support vector machine is applied to the dataset.

Table II shows the 10-fold cross validation results of SVM with different kernels applied to the dataset with extracted features. As it is shown in the table, 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. Radial basis function (RBF) is another kernel applied here to the dataset. RBF kernel on two samples x_1 and x_2 is expressed by following equation:

$$K(x_1, x_2) = \exp(-\frac{\|x_1 - x_2\|_2^2}{2\sigma^2}) \quad (3)$$

Kernel Function	Overall Error %	Spams Caught (SC) %	Blocked Hams (BH) %
Linear	1.18	93.8	0.47
Degree-2 Polynomial	2.03	85.7	0.27
Degree-3 Polynomial	1.64	89.7	0.40
Degree-4 Polynomial	1.70	90.5	0.60
Radial Basis Function	2.61	81.4	0.32
Sigmoid	13.4	0	0

TABLE II
10-fold cross validation error of SVM with different kernel functions on dataset

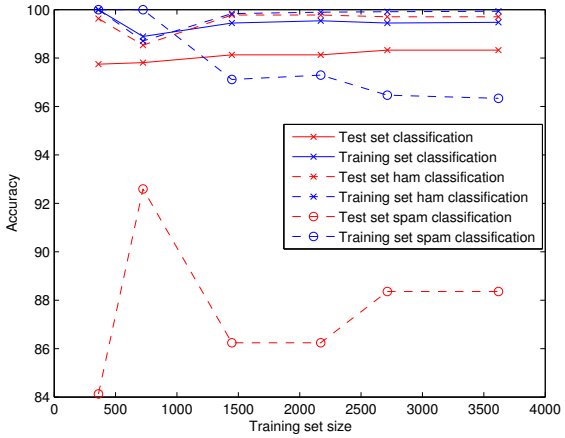

Fig. 3. Learning curve for SVM algorithm applied to final features

Finally, applying the sigmoid kernel results in all messages being classified as hams.

The learning curve for SVM with linear kernel validated using cross validation is shown in figure 3. From this figure, there is a meaningful distance between accuracy of trained model on training set and test set. 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.

V. k-nearest neighbor

k-nearest neighbor 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 neighbors.

k	Overall Error %	Spams Caught (SC) %	Blocked Hams (BH) %
2	2.78	81.3	0.46
10	2.53	82.6	0.40
20	2.98	78.8	0.35
50	3.4	74.8	0.24
100	4.14	68.4	0.16

TABLE III
10-FOLD CROSS VALIDATION ERROR OF k-NEAREST NEIGHBOR CLASSIFIER

Table III shows the 10-fold cross validation results of k-nearest neighbor classifier applied to the dataset.

VI. Ensemble Methods

In this section, two ensemble learning algorithms named random forests and Adaboost are applied to data. Ensemble learning methods combine several models trained with a given learning algorithm to improve robustness and generalization compared to single models [8]. 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.

A. Random Forests

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 scikit-learn 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.

B. Adaboost

Adaboost is a boosting ensemble method which sequentially builds classifiers that are modified in favor of misclassified instances by previous classifiers [5]. 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.

Model	SC %	BH %	Accuracy %
Multinomial NB	94.47	0.51	98.88
SVM	92.99	0.31	98.86
k-nearest neighbor	82.60	0.40	97.47
Random Forests	90.62	0.29	98.57
Adaboost with decision trees	92.17	0.51	98.59

TABLE IV
FINAL RESULTS OF DIFFERENT CLASSIFIERS APPLIED TO SMS SPAM DATASET

We tried the implementation of Adaboost with decision trees using scikit-learn library. Using 10 estimators, the simulation shows 2.1% overall error rate, 87.7% SC, and 0.74% BH. Increasing the number of estimators to 100 will change these values to 1.41%, 92.2%, and 0.51% respectively. Like Random Forests, although the complexity is much higher, naive Bayes algorithm still beats Aadaboost with decision trees in terms of performance.

VII. Conclusion

The results of multiple classification models applied to the SMS Spam dataset are shown in table IV. From simulation results, multinomial naive Bayes with laplace smoothing and SVM with linear kernel are among the best classifiers for SMS spam detection. The best classifier in the original paper citing this dataset is the one utilizing SVM as the learning algorithm, which yields overall accuracy of 97.64% . Next best classifier in their work is boosted naive Bayes with overall accuracy of 97.50%. Comparing to the result of previous work, our classifier reduces the overall error by more than half. Adding meaningful features such as the length of messages in number of characters, adding certain thresholds for the length, and analyzing the learning curves and misclassified data have been the factors that contributed to this improvement in results.

VIII. Acknowledgments

The author gratefully thanks Tiago A. Almeida and Jose Maria Gomez Hidalgo for making SMS Spam Collection v.1 available.

References

[1] Press Release, Growth Accelerates in the Worldwide Mobile Phone and Smartphone Markets in the Second Quarter, According to IDC, "http://www.idc.com/getdoc.jsp?containerId=prUS24239313"
[2] Tiago A. Almeida, Jos Mara G. Hidalgo, and Akebo Yamakami. 2011. Contributions to the study of SMS spam filtering: new collection and results. In Proceedings of the 11th ACM symposium on Document engineering (DocEng '11). ACM, New York, NY, USA, 259-262. DOI=10.1145/2034691.2034742 http://doi.acm.org/10.1145/2034691.2034742
[3] "http://en.wikipedia.org/wiki/Short_Message_Service"
[4] "http://en.wikipedia.org/wiki/Mobile_phone_spam"
[5] Adaboost, "http://en.wikipedia.org/wiki/AdaBoost"
[6] SMS Spam Collection Data Set from UCI Machine Learning Repository, "http://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection"
[7] Scikit-learn Ensemble Documentation, "http://scikit-learn.org/stable/modules/ensemble.html"
[8] T. G. Dietterich. Ensemble methods in machine learning. In J. Kittler and F. Roli, editors, Multiple Classifier Systems, pages 1-15. LNCS Vol. 1857, Springer, 2001.
[9] SMS Spam Collection v.1, "http://www.dt.fee.unicamp.br/~tiago/smsspamcollection"