SMS Spam Message Classification

Xinzhou Liu, Zhicheng Tang

***Abstract*—The explosive growth of mobile phone users has lead to a dramatic increasing of SMS spam messages. However, according to the paper we found, fighting mobile phone spam messages can be practically challenging because of several factors, including the easiness and low cost for spammers to spread spam messages and lack of public SMS spam datasets. In this report, we used the dataset provided by this paper to train different models we developed. We compare the performance achieved by our models with the results from that paper, using the same metrics. The results indicate that our best model performs at least comparable to, if not better than, the models used in the paper, and hence, it can be used to support future research in the spam filtering field.**

***Index Terms*—SMS spam, spam filtering, classification, SVM, Naive Bayes, bag of words, TF-IDF, tokenization**

I. Introduction

Short Message Service (SMS) is the text communication service component of phone, web or mobile communication systems, using standardized communications protocols that allow the exchange of short text messages between fixed line or mobile phone devices. SMS has become a massive commercial industry. As a consequence, the cell phones are becoming the largest target of SMS spam messages. SMS spam is a any junk message delivered to a mobile phone as text messaging. It is not only annoying, but it can also be expensive for cell phone users who pay to receive text messages, while the cost is very little for spammers to distribute the spam messages.

The paper we found aimed to develop spam classification methods using a large dataset they collected [1]. The main problem with studying SMS spam classification in the academic research field, as stated in the paper, is the lack of large enough public datasets for anti-spam filtering. Therefore, this paper combined spam datasets from different sources to make available a new real, public and non-encoded SMS spam corpus that was the largest one as the time of writing.

In our experiment, we used the data provided by the paper for our spam classification. We applied different train-test splitting ratios, tokenization strategies and feature extraction techniques (occurrence count, TF-IDF) to preprocess the data. Then we applied two different classification methods to the data, each of which consists one or an ensemble of machine learning algorithms. The testing data were used to evaluate the performances of those machine learning algorithms using multiple metrics.

II. Data and Methods

A. Data preprocessing

The data was downloaded from UCI Machine Learning Repository[link]. There are 5574 sentences in this data with some of the statistics shown in Table I table

TABLE I
Basic statistics

Msg	Amount	Percentage (%)
Hams	4827	86.60
Spams	747	13.40
Total	5574	100.00

According to the paper, the dataset was split into 30% training and 70% testing without stratification. In our project, we didn't only try that split ratio (Method 1), but also tried 80% training : 20% testing ratio with stratification(Method 2). Stratified splitting helps keep the ratio of different classes consistent across training and testing data.

To apply the prepossessing of the data, the paper provided two different ways of tokenizing the messages [2]: Token1: start with a printable character follow by any number of alphanumeric characters, excluding dots, commas, and colons from the middle of the pattern. Token2: any sequence of characters separated by blanks, tabs, returns, commas, colons and dashes are considered as tokens.

In our project however, we used two slightly different tokenizers:

1. We basically reproduce the second way of token the message of the paper, we split the sentences on whitespaces and kept all the lower/upper case of the words. Special symbols like exclamation mark were also kept in our word list.

2. A simple tonkenizer (regex pattern: \b\w\w+\b) that only recognizes words with more than one letter and no non-letter characters/punctuations. For example, ”I do not like U!”, after tokenization, will become only [”do”, ”not”, ”like”], leaving out the single-letter characters and punctuations.

B. Method 1: General approach

We first tried the traditional approaches applied through the paper. We named this method ”general approach”, because we used various machine learning models in one method.

With the help of the RapidMiner, we were able to fastly built the different models in one process. We split the data into 30% for training and 70% for testing. We used various models with default parameters in this method, such as SVM, Naive Bayes, Naive Bayes with Cross Validation, and a Vote model containing the previous three. We added cross validation to the Naive Bayes model to see if it can improve the results. The structure of the model is shown in Fig 1.

C. Method 2: Support Vector Machine

Support Vector Machine (SVM) is a machine learning algorithm that can be used for classification. It classifies the data by

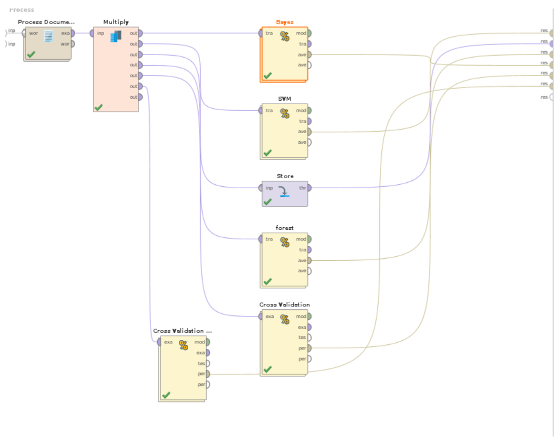

Fig. 1. Structure of the model in the general approach

finding a hyperplane that maximizes the margin between two classes. The inputs of SVM are the feature vectors extracted from the data.

The features extracted from each sentence are a vector of size of 7716, which is the size of the vocabulary. Each value in the vector is the TF-IDF value of each word in the vocabulary that may or may not be present in the sentence. Occurrence count used by "bag of words" has a main issue: longer sentences will have higher average count values than shorter sentences even they belong to the same class. To avoid such issue, TF-IDF, which is the product of term frequency and inverse document frequency, will normalize the occurrence by calculating the frequency (TF), and downscale weights for words that occur in many sentences in the corpus and are therefore less informative than those that occur only in a smaller portion of the corpus (IDF). For word i in sentence j:

$$w_{i,j} = tf_{i,j} \times \log(\frac{N}{df_i})$$

where $tf_{i,j}$ = number of occurrences of i in j, df_i = number of occurrences of sentences containing i, N = total number of sentences.[3]

To try SVM models with hyperparameters of different values to get the best model, we used the GridSearch API from Python machine learning package *scikit-learn* to implement the fine tuning using the training data. The tuned hyperparameters and their values are: kernel = [linear, rbf], C = [1, 10, 100, 1000], and gamma for rbf kernel = [0.001, 0.000.1].

The linear kernel simply assumes that the original dataset is linearly separable, while the rbf kernel maps the original dataset into a higher dimensional space to make it linearly separable. 'C' behaves as the regularization parameter for SVM. For larger values of C, a smaller margin will be accepted if the decision function is better at classifying all training points correctly. A lower C will encourage a larger margin, therefore a simpler decision function, at the cost of training accuracy. The gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'.[4]

5-fold cross-validation was performed during training, and scoring metrics that are used to evaluate and optimize the trained models are precision, recall and accuracy. When training was finished, the different models were evaluated using the testing data. For each scoring metrics, the model with the best performance was selected, so we ended up with 3 best models.

III. Results

A. Results from the paper

Based on the two different tokenization strategies described earlier in the "Data and Methods" section, the paper applied different models to the data set and obtained results shown in Table II

TABLE II
Results from the paper

Classifier	SC	BH	Acc	MCC
SVM + tok1	83.10	0.18	97.64	0.893
Boosted NB + tok2	84.48	0.53	97.50	0.887
Boosted C4.5 + tok2	82.91	0.29	97.50	0.887
PART + tok2	82.91	0.29	97.5	0.887
MDL+tok1	75.44	0.35	96.26	0.826
C4.5 +tok2	75.25	2.03	95.00	0.770
Bern NB+tok1	54.03	0.00	94.00	0.711
MN TF NB +tok1	52.06	0.00	93.74	0.697
MNBool NB + tok1	51.87	0.00	93.72	.0.695
Inn + tok2	43.81	0.00	92.70	0.636
Basic NB +tok1	48.53	1.42	92.05	0.600
Gauss NB + tok1	47.54	1.39	91.95	0.594
Flex NB + tok1	98.04	26.01	77.13	0.507
Boolean NB + tok1	98.04	26.01	77.13	0.507
3NN +tok2	23.77	0.00	90.10	0.462
EM + tok2	17.09	4.18	85.54	0.185
TR	0.00	0.00	86.95	-

There are several metrics used in the paper to measure the performance of the model. *SC* stands for the spam caught which indicates the proportion of the spam we detect from the total group of spams. *BH* stands for blocked ham which evaluates how many hams has been miss classified as spam.The formula of the two metrics are listed below.

$$SC = TP/(TP + FN) = Recall$$

$$BH = FP/(TP + FP) = 1 - Precision$$

Also the paper used another metric to evaluate the performance of binary class classification, called Matthews Correlation Coefficient (MCC):

$$MCC = \frac{TP * FN - TN * FP}{\sqrt{(TP + TN) * (TP + FP) * (FN + TN) * (FN + FP)}}$$

The MCC is in essence a correlation coefficient between the observed and predicted binary classifications; it returns a value between -1 and +1. A coefficient of +1 represents a perfect prediction, 0 no better than random prediction and -1 indicates total disagreement between prediction and observation. And the paper listed the results according to the MCC in decent order.[5]

To be consistent with the paper in order to cross compare its results with ours, we used the same performance metrics in our evaluation.

B. General approach results

Based on our general approach, we have come up with a result of each model based on the metrics mentioned in the paper, the results is listed as below in Table III Based on the

TABLE III
EVALUATION RESULTS OF EACH MODEL IN METHOD 1

classifier	SC%	BH%	Acc%	MCC
SVM	46.8	0	92.88	0.66
Bayes	79.7	5.0	92.88	0.71
Corss+Bayes	90.5	5.1	94.28	0.78
Vote	89.2	5.3	93.95	0.77

result of our general approach, we found that we should dig in more into the segmentation and method in order to improve the performance.

C. SVM results

Since there are 3 best models trained and optimized based on the 3 performance metrics (precision, recall, accuracy), they were evaluated on the testing data. The results were shown in Table IV.

TABLE IV
EVALUATION RESULTS OF EACH MODEL IN METHOD 2

Optimized model	Precision%	Recall%	Acc%	MCC
Precision	100	83.89	97.85	0.905
Recall	97.87	92.62	98.74	0.945
Accuracy	97.87	92.62	98.74	0.945

As shown in Table IV, the first model (kernel = rbf, C = 100, gamma = 0.001) optimized based on precision score gave best performance on precision (100%), and the next two models optimized based on recall and accuracy happened to be the same model (same hyperparameters, kernel = rbf, C = 1000, gamma = 0.001), and achieved the best performances on recall (92.62%) and accuracy (98.74%), respectively. MCC measures the quality of binary classification even if the classes in the dataset are imbalanced. Since our dataset is imbalanced (ham:spam = 87:13, Table I), the recall- and accuracy-optimized models showed a better performance on this imbalanced data (MCC: 0.945), almost near-perfect prediction.

TABLE V
EVALUATION RESULTS OF EACH MODEL IN METHOD 2 AFTER METRIC CONVERSION

Optimized model	SC%	BH%	Acc%	MCC
Precision	83.89	0	97.85	0.905
Recall	92.62	2.13	98.74	0.945
Accuracy	92.62	2.13	98.74	0.945

After converting the metrics to those used by the paper (Table V), we selected the precision-optimized model as our best model, because it gave the best overall performance considering not just 1 but all of the 4 metrics (SC, BH, Acc, and MCC).

IV. DISCUSSION

A good classification model needs to capture as many spam messages as possible (hight SC) while avoid misclassifying legitimate messages into spam (low BH). And it also needs to be able to classify messages with an imbalanced spam:ham ratio (high MCC).

The best model in the paper used SVM with tokenization that ignores punctuations and only generates tokens from words. Comparing our results (Table VI) with the best result (Table II) from the paper, we found that the performance of the best model from the general approach (Naive Bayes with cross-validation) in terms of the MCC score only scored 0.78 on MCC lower than the best model from the paper, but it achieved 90.5% spam caught rate, significantly higher than the paper's best model (83.10%). Note that our general approach (Method 1) applied the same data preprocessing techniques as in the paper, i.e., splitting dataset into 30% training and 70% testing, converting words separated by whitespaces to tokens, and using the occurrence counts as the features.

TABLE VI
BEST MODELS FROM THIS PROJECT AND THE PAPER

Model	SC%	BH%	Acc%	MCC
Corss+Bayes (Method 1)	90.5	5.1	94.28	0.78
SVM (Method 2)	83.89	0	97.85	0.905
SVM (Paper)	83.10	0.18	97.64	0.893

While the general approach achieved a better performance on certain metrics such as SC, the SVM approach (Method 2) achieved a slightly better overall performance, i.e., higher SC, Acc and MCC, but lower BH, compared to the paper's best model (Table VI). Note that the data preprocessing was different in Method 2 than Method 1 and the paper, using a different train-test split ratio (8:2), a different tokenization strategy, and more advanced feature extraction technique (TF-IDF).

If the size of training dataset is small, the model tends to underfit, so increasing the ratio to include more training data could help enhance the performance. Different tokenization stratgies can result in generating vocabularies with different sizes. The size of the vocabulary is related to the feature vector size in our case, so it is worth trying out different tokenization strategies. Note that, not all features are useful to the classification, and some might cause overfitting. That is likely why the tokenization used in the paper and our experiment to achieve the best performances, happens to yield a smaller vocabulary, therefore, smaller feature vector, and ignore some words and punctuations.

Furthermore, the bag of words are a good baseline feature extraction technique, but it doesn't capture the importance of each word in different sentences. The importance increases proportionally to the number of times a word appears in the sentence but is offset by the frequency of the word in the corpus. Therefore, TF-IDF can be a better measurement of the relevance of a document regarding each class (ham or spam).

V. CONCLUSION

We tried various classification methods and compared our results to those in the paper. Some of our models (Method 1) achieved similar, if not better performances to the best models in the paper. We analyzed the different data preprocessing techniques and hyperparameters used by the models. The best model used SVM with hyperparameters: kernel = rbf, C = 100, gamma = 0.001. After comparing our results to the paper, we concluded that the performance of a model depend on various factors such as dataset split ratio, stratification, tokenization strategies, feature extraction techniques, and the model's hyperparameters, although we couldn't directly examine the exact difference each factor can make. In the future, we would like to apply our methods to a larger and more balanced dataset. Our results can potentially contribute to the design and creation of highly effective spam filtering software.

References

[1] T.A. Almeida, J.M.G. Hidalgo, A. Yamakami, *Contributions to the Study of SMS Spam Filtering: New Collection and Results*. Proceedings of the 2011 ACM Symposium on Document Engineering, Mountain View, CA, USA.

[2] C. Siefkes, F. Assis, S. Chhabra, and W. Yerazunis. *Combining Winnow and Orthogonal Sparse Bigrams for Incremental Spam Filtering*. In Proc. of the 8th ECML PKDD, pages 410–421, Pisa, Italy, 2004.

[3] www.tfidf.com

[4] rbf kernel (scikit-learn.org)
rbf parameters (scikit-learn.org)

[5] Mathews correlation coefficient (scikit-learn.org)