SMS Text Message Spam Identification

Mary Yu, Nuo Tian, Sichang Tu

Objective

The objective of this project is to identify spam messages using Natural Language Processing (NLP) protocols. The goal is to fit and train models that will distinguish and filter spam SMS messages with a substantial accuracy. The data used contains 747 spam and 4,827 ham messages. In order to achieve the best results, the data were preprocessed and using different feature selection methods such as tf-idf, word count, length of message and number of punctuation. The models used in this project are Multinomial Naive Bayes, Random Forest, XGBoost, Support Vector Machine and Neural Network. The models are trained on various thresholds and the best results were generated from the Naive Bayes model.

1 Dataset

The dataset used in this project is called the SMS Spam Collection Data, and it comes from the University of California, Irvine's Machine Learning Repository. The data set contains 747 spam messages as well as 4,827 ham messages, and is one of the largest and most frequently used SMS spam identification dataset. Within this data set, a collection of 425 spam messages were manually extracted from the Grumbletext Website, a subset of 3,375 ham messages were randomly selected from the National University of Singapore (NUS) SMS Corpus (which contains about 10,000 ham messages), 450 ham messages from Caroline Tag's PhD Thesis, and finally, the SMS Spam Corpus v.0.1 Big, which contains 1,002 ham messages and 322 spam messages, was incorporated. The links to the websites of these data sets could be found in the appendix.

This SMS message collection is contained in one text file, and each line represents one observation and has two attributes. The beginning of each line has the correct class/label (ham or spam), followed by the raw text message. The label "ham" is a short synonym, representing "non-spam". These messages are not sorted in any way.

2 Background

Short Message Services, more widely known as SMS messages, was a popular form of communication. Nowadays, it is not only used as a form of communication between family and friends, but has become a convenient way to schedule appointments, receive notifications, manage security verifications and so on. However, because of the growing use of the SMS text messages, new ways of practicing fraud have been taking place using spam messages. These fraudulent acts involve sending user spam messages that contain a link, and attracts users to click these fraud links by claiming that they have won a prize or have a "important message" to view. This can potentially become a widespread problem if cellular-service carriers fail to come up with a solution to filter and block these messages, because it can cause their customers to lose money and also face a risk of leaking their personal information.

The spam-identification task, however, is not easy to achieve. There are several challenges in identifying and filtering spam text messages. The first challenge is the volume of the messages. It is not hard to imagine the amount of text messages sent per day, and running these messages through filters will require the filter to be as efficient as possible. On the other hand, it

is hard to build one model and utilize it for a long time, because spam messages can be changed easily, and it may not be classified as a spam message after its transformation. The above problems lead to the final problem, which is the expense of keeping and maintaining such a filter system. Therefore, it is essential to find out key factors that distinguish spam and ham messages, and develop a system that is both efficient and accurate.

In order to identify a correct track and extract experiences from previous researches, a number of literature reviews were done on the SMS spam filtering subject. The original authors of the dataset, Almeida et al. (2011) first collected and analyzed this dataset, and the team used a combination of tokenizers and machine learning classifiers. For the two tokenizers, the first tokenizes the text by excluding the dots, commas and colons, and the second uses blanks, tabs, returns, dots, commas, colons and dashes as tokens. The evaluation metrics used in this research includes Spam Caught (SC%), Blocked Hams (BH%), Accuracy (Acc%) and Matthews Correlation Coefficient (MCC). The final result showed that Support Vector Machine (SVM) and the first tokenizer performed best with 83.10 SC%, 0.18 BH%, 97.64 Acc% and 0.893 MCC. The same dataset was used by Delany et al. (2012), who summarized the current state of the art classifying and filtering methods for spam-identification. From this research, the SVM model with linear kernel also had the best performance that achieved high detection rates for individual spam message clusters.

Apart from SVM, some other classifiers as well as spam filters were recognized with good performances. Cormack et al.(2007) selected some high performing spam filters including Bogofilter(Bayesian spam filter), Dynamic Markov Compression, TR-IRLS(an open-source logistic regression classifier), OSBF-Lua(a Bayesian classifier) and SVMlight(a free-for-scientific use support vector machine classifier. The authors evaluated each classifier on the corpus using 10-fold cross validation and found that Bogofilter and OSFBF-Lua perform the worst on the raw messages but are fairly good when using textualized features. DMC is not feature based, so it performs well without any modifications. Logistic Regression and SVM perform well on the features. The results also indicate that the effect of shorter and sparser text since the first two filters perform poorly on the messages before tokenizations. Al Moubayed et al.(2016) presented a novel approach for SMS spam filtering by using probabilistic topic modelling and Stacked Denoising Autoencoder(SDA). To avoid the disadvantages of most common methods for SMS feature extraction, they adopted a text mining technique, which can model latent patterns in the data. Then the authors applied SDA, an unsupervised deep neural work to build the model, since the labelled training data are limited. In order to further separate the two labels, the authors used reconstruction errors of the SDA model as features and passed it to a Fisher's Linear discriminant analysis(FDA). This model reaches 99.52% for F-score, 99.34% for Precision, 99.91% for Recall, 99.43% for Ham accuracy and 99.28% for Spam accuracy.

Finally, Gomez et al.(2006) talked about building Bayesian filters using Machine Learning techniques applied to the pre-classified messages(SMS collection dataset is qualified). The authors explained in detail about the learning process, which includes five steps: Preprocessing, Tokenization, Representation, Selection and Learning. This process is adapted into this project. Preprocessing indicates the deletion of irrelevant elements(e.g. HTML), tokenization dividing the message into semantically relevant segments, representation converts the message into the attribute-value pairs' vector which values could be binary or frequencies, selection deletes the less predictive attributes(e.g. Stop words), and learning which means fit the data into different classifiers(eg. SVM, NB, NN).

3 Methodology

3.1 Preprocessing

We know that basic NLP text preprocessing methods are stemming, normalization, lemmatization, stop-words removal, and punctuation marks removal. In this project, we conducted stemming and stop-words removal only to all the cases, not even changing text into lowercase. This is because of the special nature of our tasks: Spam message identification. In order to identify spam messages, punctuation, capitalized words, and the length of the messages are all very useful. For those text processing approaches, like tf-idf, which depends on the occurrence frequencies of each

word, punctuations are removed and normalizations are conducted as default.

3.2 Feature Extraction

Feature extraction is the process of selecting features from data in order to use them as a classification indicator or criteria. In this project, we used four different types of feature selection method: commonly used word count vector and tf-idf representation, and two specifically designed length of message vector and count of punctuation vector. For word count vectors, each row corresponds to the documents in the corpus and each column corresponds to the tokens in the dictionary (which is created by taking a list of unique tokens in the corpus). It is a good way to classify text messages by looking at the frequency of each word in the message appearing in a large corpus of messages. For tf-idf, it is quite similar to the word count vector, the only difference is that it added a statistica weight to each word used to evaluate how important a word is to a message in a corpus. Two additional features that we added specifically for this project: one is the length of message. It is a useful indicator to see if the length of message could be useful in spam message identification since longer text messages may have a higher probability of being a spam message. The other feature is the number of punctuations in a text message. Unnecessary inclusion of a large group of punctuations could also serve as one criteria of distinguishing spam and ham messages. One of the important reasons why we would like to see how these two simple features could work is that most spam identification models are expensive to use, a simpler text data representation may have a not perfect but still reasonable model performance.

3.3 Machine Learning Models (SVM, RF, XGBoost, Naive Bayes)

Several learning approaches were examined for this binary classification task, including Support Vector Machine (SVM), tree based models, such as Random Forest (RF) and XGBoost, and Naive Bayes (NB) , mostly using the scikit-learn implementation. All selected models are widely employed in classification tasks.

As one of the most popular models in machine learning, SVM has always been used in spam detections tasks. We also included LinearSVC in this task since it is good at solving machine learning problems with small sized datasets and its high interpretability. Regulation has been applied through adjusting the C parameter in the models.

For tree-based classifiers, Random Forest and XGBoost have been implemented in this task. We chose RF for it can reduce the overall variance and error, can perform well on imbalanced datasets, and is less influenced by outliers. Additionally, it has good generalization capacity and less tendency to overfitting. Since the dataset is not huge, the complexity of the model is controlled by altering the number, depth of the tree and samples. GridSearch has been utilized in this part to figure out the best combination of all parameters. Different from RF, XGBoost is an end-to-end tree boosting system, which improves largely in regularization, handling missing values and effective tree pruning. To implement this model and avoid overfitting, we test out the combination of a series of parameters, such as learning rate and the numbers of trees and samples, through 10 runs, and retrain the model with best parameters to get the optimal result.

As a high-bias algorithm, Naive Bayes performs well when the dataset is small and is unlikely to overfit. And compared with tree-based methods, NB is way fast and simple to implement. In this task, Multinomial Naive Bayes (MNB) was used since it is suitable for tackling classification tasks with discrete features, such as word counts. Unlike other models, NB has fewer parameters to alter and the result would be largely impacted after adjusting the parameter, thus we adopted the default setting in MNB.

3.4 Neural Network Models (TextCNN, TextRNN)

For implementing neural networks approach, two models were tested including TextCNN and TextRNN, both using the implementation in TensorFlow Keras module.

To build the TextCNN model, we utilized the Sequential function to stack it. After being preprocessed and padded, the prepared texts were put into the embedding layer, followed by a dropout layer to avoid overfitting. Then a

Conv1D layer and a GlobalMaxPooling1D layer were added to extract the most significant feature. Another dropout layer was also used before the fully connected layer to deal with overfitting. As for the TextRNN model, the prepared texts were first go through the embedding layer and a bidirectional LSTM layer. Compared with other RNN models, Long Short-Term Memory (LSTM) can achieve a better performance when dealing with text data, since it is able to capture the dependencies and correlations of words in the context even if the distances are long. And bidirectional LSTM further tackle the incapacity of capturing the backward information in single direction LSTM. Similar to the TextCNN model, a dropout layer was applied after the texts were encoded by the BiLSTM layer. A last sigmoid layer was applied instead of the softmax layer due to the classification task being binary.

Both models were compiled with binary cross entropy loss function and using Adam as the optimizer. In order to ensure the generalization capacity of the models, a 20 percent validation split was introduced during the training process.

Model\Features	Word Count	TF-IDF	Length Vector	Num of Punc
SVM	0.9901	0.9901	0.8788	0.8698
Naive Bayes	0.9910	0.9685	0.8832	0.8833

Table 1.1: SVM and Naive Bayes model

4. Result

Table 1.1 indicates how each different feature perform on the Support Vector Machine model. SVM is the model with the highest performance on the tf-idf feature. It also shows that length could be a better indicator compared with the number of punctuations in the text messages. The Naive Bayes model on word count has the overall highest model accuracy, and it performs fairly well on the length vector and number of punctuation vectors, we recommend this model the most.

Model\Features	Word Count	TF-IDF	Length Vector	Num of Punc
Random Forest	0.9739	0.9721	0.8967	0.8833
XGBoost	0.9793	0.9820	0.9039	0.8833

Table 1.2: RF and XGBoost model Performance

Table 1.2 shows the Random Forest and XGBoost model performance. It is a strong model and has a very high accuracy. However, it is also a very expensive model to train. Thinking of the trade off between accuracy and model complexity, we do not recommend this model. It is interesting to see that though the accuracy of word counts and tf-idf on Random Forest are lower than that of SVM, the performance of Length Vector and Number of Punctuations is improved. With more modifications on these two features, or maybe a combination, may also improve the model accuracy.

Model\results	Loss	Accuracy
TextCNN	0.0566	0.9883
TextRNN	0.1023	0.9911

Table 1.3: TextCNN and TextRNN model Performance

Tabel 1.3 gives the loss and accuracy of TextCNN and TextRNN. For neural network models, loss and accuracy actually measure different things. Cross-entropy loss tends to be lower if the predictions are closer to the class label. And accuracy is a discrete value which shows the binary value for certain samples. As shown in the table, in terms of loss value, TextCNN performs better than TextRNN, which indicates that the predictions of TextCNN are much closer to the class label. However, if accuracy is used as the evaluation metric, the performance of TextRNN surpasses TextCNN. And the accuracy of TextRNN is the highest among all models applied to this task.

5. Discussion

In this project we were able to fit several models using several tokenizers. Apart from following the methods of the original authors, which is utilizing the combination of tokenizers and classifiers, we were also able to fit other models that weren't employed in previous research. However, in the case of conclusion and reflection, it seems that the tokenizers such as word count, message length can be further combined with models such as random forest, XGBoost and neural networks, maybe there will be surprising findings. Apart from reflecting on

the model and tokenizer choices, we were also trying to understand the relationship between the accuracy of the model and the complexity of the model. Although we all know that there is a bias-variance tradeoff between the complexity and the accuracy of the models, some of our simple models with simple tokenizers seemed to perform pretty well. The neural network models have high performances as well, but it has less interpretability. Therefore, it is essential to understand the demand in order to pick the right model for the task. Sometimes the most accurate model may not be the best option considering its training time and cost.

There are still some areas of feature selection that we haven't been able to touch in depth in this project. First, the abbreviation words could be a strong indicator of ham message. In future work, we would like to take a deep look at those and make it one of our additional features. Second, capitalized words are also interesting to look at, especially those messages which are all capitalized. Next step would be to select those capitalized words and put statistical weights over each of them to see if it would work better than these features that we have so far. Third, numbers and URLs could also work. In this study, we chose the length of message and number of punctuations as examples, but there are many other features that could also work.

A Appendix

Citations

Almeida, T.A., Gómez Hidalgo, J.M., Yamakami, A. Contributions to the Study of SMS Spam Filtering: New Collection and Results. *Proceedings of the 2011 ACM Symposium on Document Engineering (DOCENG'11)*, Mountain View, CA, USA, 2011.

Al Moubayed, Noura, et al. "SMS Spam Filtering Using Probabilistic Topic Modelling and Stacked Denoising Autoencoder." *Artificial Neural Networks and Machine Learning – ICANN 2016,* edited by Alessandro E.P. Villa et al., vol. 9887, Springer International Publishing, 2016, pp. 423–30.

Delany, Sarah Jane, Mark Buckley, and Derek Greene. "SMS spam filtering: Methods and data." *Expert Systems with Applications* 39.10 (2012): 9899-9908.

Cormack, G. V., GÃ³mez Hidalgo, J. M., and Puertas SÃ¡nz, E. Feature engineering for mobile (SMS) spam filtering. Proceedings of the 30th Annual international ACM Conference on Research and Development in information Retrieval (ACM SIGIR'07), New York, NY, 871-872, 2007.

Gomez Hidalgo, J. M., Cajigas Bringas, G., Puertas Sanz, E., and Carrero Garcia, F. 2006. Content based SMS SPAM filtering. In *Proceedings of the ACM Symposium on Document Engineering*. Amsterdam, Netherlands: Association for Computing Machinery.

Subset Data Sources

Grumbletext Website: http://www.grumbletext.co.uk/
NUS SMS Corpus: https://www.comp.nus.edu.sg/~rpnlpir/downloads/corpora/smsCorpus/
Caroline Tag's PhD Thesis: https://etheses.bham.ac.uk/id/eprint/253/1/Tagg09PhD.pdf
SMS Spam Corpus v.0.1 Big: http://www.esp.uem.es/jmgomez/smsspamcorpus/