SMS Spam Detection using Machine Learning Methods

Written by CS3244 Machine Learning Group 13
Guai Zi Wei A0155778L, Lin Ziwen Kelvin A0150057N, Loh Cai Jun A0160231Y,
Lee Chia Hong A0158119B, Lee Jin Yao A0168574N, Ng Kai Cong A0164556X
National University of Singapore
21 Lower Kent Ridge Road
Singapore 119077

Abstract

Mobile phone messaging apps such as Short Message Service (SMS) and WhatsApp to communicate are widely used and considered essential in our world today. In Singapore, it is not uncommon for phone users to receive unwanted messages which are also known as spam. To solve the issue of receiving spam messages, we proposed creating a spam filtering application using machine learning. We analyzed and show the experimental results of using 2 different machine learning models: K-Nearest Neighbour (k-NN), Support Vector Machine (SVM); in detecting spam and ham messages. Lastly, we discuss the unique features of Spam Filtering Application using these machine learning models.

Introduction

The use of mobile phone messaging apps such as Short Message Service (SMS) and WhatsApp to communicate is considered essential in our world today. For the purpose of the project, we consider the messages received in phone messaging services and SMS to be the same. While SMS is gradually being replaced by newer messaging systems, it is still widely used. The low cost of messaging services brought about a surge in spam and phishing texts being sent. In Singapore, it is not uncommon for phone users to receive unwanted messages from numbers of advertising money lending services or property launches.

These messages, which are known as spam, are an annoyance for people who receive it but can do little about it other than deleting them. Furthermore, as SMS scams and phishing are also prevalent, uninformed receivers will be exposed to possible cheating and money laundering schemes. Tactics such as SMS spoofing also make phishing attacks much harder to detect when the SMS is disguised as an official SMS. The problem of SMS spam remains prevalent as there is limited spam-filtering software available on smartphones that can effectively filter out spam, as such, we feel that it is a compelling problem to tackle using machine learning.

Since it is impossible to discern the random nature of spam itself, it is neither simple nor efficient to hard-code a program to filter spam as new types of spam messages are created all the time. Instead, we can use machine learning (ML) algorithms, which adapt well to continuous changes, as it allows us to filter messages faster and more efficiently. Current frameworks that are highly accessible also make ML models easier to program. To train these models, we will need a set of labelled spam and non-spam messages.

In this paper, we will write about 2 machine learning models: K-Nearest Neighbour (k-NN), Support Vector Machine (SVM). We will also show the experimental results of these machine learning models. Additionally, we will discuss how these machine learning algorithms can be used to help us filter spam messages. Our end goal is to design an algorithm that can be used in as a phone app to filter spam SMS messages.

Roles of members

Guai Zi Wei - SVM Implementation, Report Writing

Kelvin Lin - k-NN Implementation, Report Writing

Loh Cai Jun - SVM Implementation, Report Writing

Lee Chia Hong - SVM Implementation, Report Writing

Lee Jin Yao - k-NN Implementation, Report Writing

Ng Kai Cong - k-NN Implementation, Report Writing

Dataset Description

The dataset we have chosen contains 5,572 SMS messages in English. 4,825 of them are legitimate (ham) SMS while 747 of them are spam SMS. Out of the 4,825 ham SMS, 3,375 of them were randomly chosen from the NUS SMS Corpus (NSC), which is a dataset of about 10,000 legitimate messages with most of them were originating from Singapore. (Dua and Graff 2017) (Almeida 2011) (Almeida and Hidalgo 2019)

Due to the nature of the SMS in this dataset, we hope that the models will be able to learn from the unique lingo used in Singapore, Singlish and hence be able to classify spam and ham SMS that are being sent in Singlish.

	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
5	spam	FreeMsg Hey there darling it's been 3 week's n...

Figure 1: Snapshot of the chosen dataset

Overview of ML models used

Support Vector Machine model (SVM)

Introduction to SVM

The first machine learning algorithm used is the Support Vector Machine (SVM), which is able to define decision boundaries by finding the maximum margin between two classes. Firstly, the model will plot each data item as a point in n-dimensional space (where n is number of features in the data set) with the value of each feature being the value of a particular coordinate.

SVM will proceed to perform classification by finding the hyperplane that is best able to differentiate the two types of SMS. This is done by maximizing the distance between the nearest data point from both classes and the hyperplanes. The distance mentioned here is known as the margin while the nearest data point to the optimal hyperplane is known as support vectors.

One unique advantage of using SVM for SMS spam filtering is that we can implement a form of active learning that can be personalised to each individual. We start with a general set of spam and ham messages to train the initial model. This will be the base model for each user. The model will then train itself only on messages which fall between the margins of its hyperplanes by querying the user. These messages can be added in small batches, simulating the number of messages received a day or time period and at the same time, reducing memory requirements for the algorithm.

Training the SVM model this way is fast as only messages within these batches that the SVM is unsure about will be actively trained. Personalisation can be achieved through this because during the active learning phase, only spam and ham messages that the user receives will be used to further tune the model.

Experimental results of SVM Algorithm

Our SVM algorithm and code implementation for training model are available in Appendix A. The train size refers to percentage of training data fitted into the model before active learning. From Figure 2, we can see that the optimal training size is 0.2625 or 1,462 messages. We observed that when the training size becomes too small, the precision suffers greatly. This is because as the training size decreases, the parameters of the SVM model decrease as well and less parameters leads to less active learning due to lower ability to generalize. The end result is a model which is making pure guesses of what is ham and spam. For the models ran above, the batch size used was fixed at 10. We will then proceed to tune the batch size while fixing training size at 0.2625.

	Train_size	Time	Active Learned Data	Train Accuracy	Test Accuracy	Test Recall	Test Precision
0	0.3750	0.048869	26.0	0.968421	0.965542	0.827381	0.879747
1	0.3375	0.065824	45.0	0.976512	0.967696	0.791667	0.930070
2	0.3000	0.057844	40.0	0.973684	0.965542	0.821429	0.884615
3	0.2625	0.067819	54.0	0.974972	0.964106	0.851190	0.851190
4	0.2250	0.045877	0.0	0.919002	0.909548	0.898810	0.580769
5	0.1875	0.038854	0.0	0.945136	0.934673	0.791667	0.703704
6	0.1500	0.044906	0.0	0.952751	0.941134	0.839286	0.719388
7	0.1125	0.069814	0.0	0.951590	0.949031	0.791667	0.786982
8	0.0750	0.046478	0.0	0.944444	0.941134	0.845238	0.717172

Figure 2: Choosing optimal training size

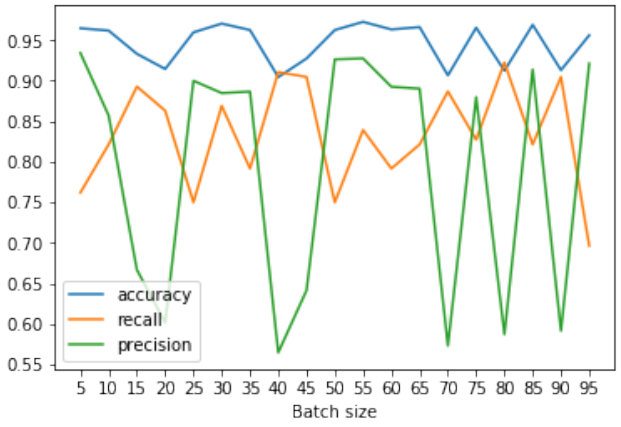

Figure 3: Choosing optimal batch size for active learning

Figure 3 shows that batch size actually matters during the active learning phase. The final accuracy only depends on the SVM's active learning. The high fluctuation in precision, observed is due to the noise present in the dataset. This means that sometimes, the model will label a ham message as spam. Hence we have to be careful with the batch size chosen since we do not want ham to be wrongly classified as spam. In particular we will choose batch sizes with about 0.9 precision. We believe this threshold is acceptable since the messages in SMS are generally less important, so misclassifying some ham messages as spam can be allowed.

In general, recall values can reach 0.9, which means that the model is able to accurately identify spam messages 90% of the time if the message is indeed spam. This is good news because we want to be able to identify spam messages as accurately as possible.

Based on the information above, we will pick batch size equal to 30, as the model was able to achieve an acceptable accuracy, precision and recall. Furthermore, we also feel that

30 is a good estimate for how many SMSes a person can receive in a time period of a day. Hence, we will proceed with running the SVM model with active learning on 3 different seeds and compare the results.

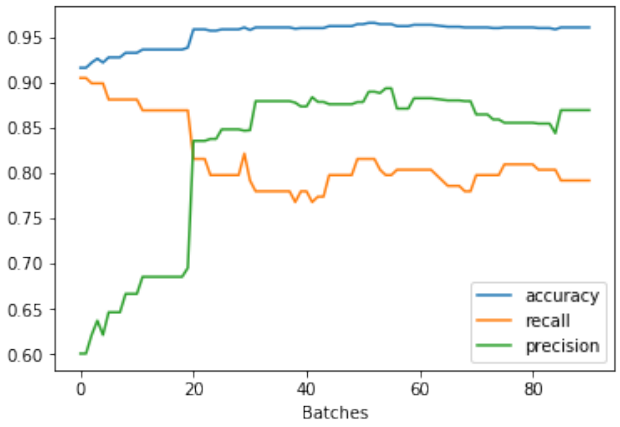

Figure 4: Model Results from first seed

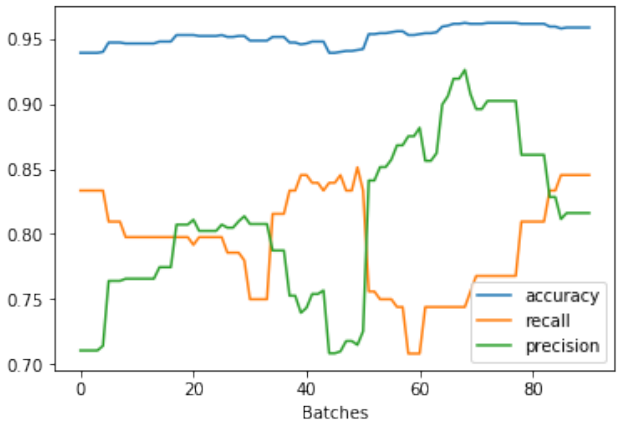

Figure 5: Model Results from second seed

Figures 4, 5 and 6 show the metrics for accuracy, recall and precision for the models as it does active learning. The model has been initially trained on a base set consisting of 1,462 messages. We then feed it batches of 30 messages until the training set is depleted.

From the graphs above, we can see that there are stretches where the metrics became a flat line. This means that no active learning occurred on some of the batches of messages. In other words, this means that the SVM model at that moment can confidently classify all data in the batch.

We observed that as more batches of SMS messages were passed through the models, accuracy and precision increases, while recall takes a slight dip. This is expected because in our dataset which contains more ham than spam, so as more batches are added, the algorithm will be less likely to misidentify ham as spam messages, but as a consequence it suffers slightly in its ability to identify spam when the message is indeed spam.

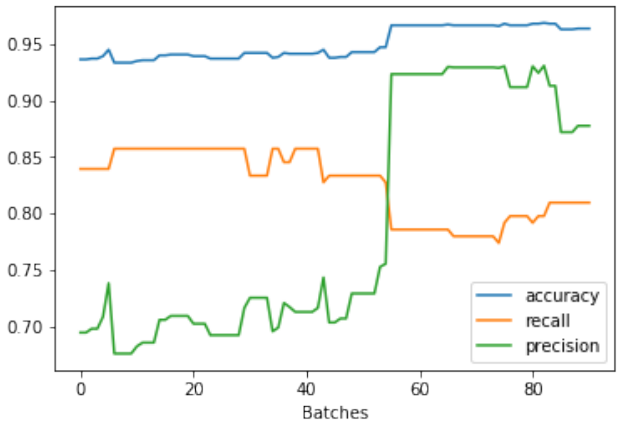

Figure 6: Model Results from third seed

We get these results because the parameters required to accurately make a classification are out of the current parameter space of the SVM and the noisy nature of SMS messages leads the algorithm to think that a spam message is ham. This means that there is a tradeoff during active learning: As accuracy increases, the SVM is less likely to mistake ham as spam, but it has reduced ability to identify spam. We think that this tradeoff is actually good as we want the SVM to correctly identify ham as mentioned earlier.

Based on the above graphs, we can see that the number of batches required for the SVM to reach 80% precision and recall varies quite a lot. This is expected since in the real world, different people will receive different types of messages. It can take 20 to 80 batches, so if each batch took a day, it can take from 3 weeks to 2.5 months for the tuning to be optimal. What we can guarantee is that precision in identifying spam messages will increase at the end of the training. Note that we are not looking for 90% precision and accuracy because of the random nature of the active learning process, which can affect the maximum precision and accuracy the model can attain.

Future Adjustments to SVM Model

Training Period We are aware of the uncertain time period required to fully train the SVM model. As a result, we propose an internal cut-off for the algorithm of about 20-30 batches (600-900 messages). If the misclassification rate is not acceptable e.g. user reports multiple misclassifications, the algorithm will restart for another 10 batches. This method uses the fact that the number of batches of messages required to optimally train the SVM will differ between individuals and doing this will allow the algorithm to guess when it has been optimally trained.

Memory Constraints The base model of the SVM decides what parameters will be in the final model after active training. While this reduces the memory requirements of the

algorithm, it also penalizes its generalization ability. An adjustment we could make is to retrain a new base model with the starting dataset and the actively learned data. However, this would increase the memory needed for the app which could be an undesirable tradeoff.

K-Nearest Neighbour model

Introduction to K-Nearest Neighbour

The second model we have used is the k-nearest neighbour (k-NN) classifier which is a straight forward supervised machine learning (ML) algorithm that excels in classifying simple recognition and regression problems. The k-NN algorithm handles classification by grouping data points and attaching class labels to them. This can be done by calculating the distance between each new training example and all the training examples already incorporated into the algorithm and then based on the labels of the nearest K-neighbours, classifies this new query example. Its ease of implementation and understanding makes it a good ML algorithm to use in classifying SMS spam.

When new data needs to be classified, k-NN will use the trained data set to compare and classify the new data into similar categories. As k-NN calculates the similarity between messages to be classified and the messages in the training data set, its performance will slow down when a large training data set is used. However, since SMS spam messages often have a similar structure, a huge data set may not be required for training; keeping a representative data set of ham and spam examples might be sufficient. The main challenge for k-NN models is the tuning of the number of closest neighbours, k allowed. A small k value can cause overfitting while a very large k value can reduce classification accuracy.

Experimental results of k-NN algorithm

Our k-NN algorithm and code implementation for training model are available in Appendix A. Firstly, we select a percentage of the sample dataset to be validation set. The k-NN algorithm runs much slower with larger dataset as it needs to check against all its training examples for each query. On the other hand, lowering the training data size provides for far less examples to check against, which would reduce accuracy.

Training Data Size	Time Taken (seconds)	Misclassification Percentage
1394 (25%)	58.02202s	3.6372%
2230 (40%)	72.70014s	4.0981%
3344 (60%)	83.17433s	3.4096%
4459 (80%)	81.92089s	2.6930%
5016 (90%)	78.8581	3.0521%

Figure 7: k-NN Dataset Size vs Runtime and Accuracy

Figure 7 illustrates the time taken for different sample sizes, taken over an average of 10 runs with a k-value of 10. We can see that the time taken to run the validation set increases, but peaks at 80% size. The misclassification rate also decreases as expected with a larger training data size.

Hence, for this experiment, we have randomly selected 20% of the 5,572 SMS messages to be the validation set. Before the data set is segregated into training and validation set, they are cleaned up first, such that any emojis and other special characters (excluding the standard punctuation marks) are removed in order to classify the messages more accurately.

Then, we run the experiment with a starting k-value of 1. We repeat the experiment by increasing the k-value by 2 up to k-value of 39. The reason to omit even k-values is that we do not want to have a case whereby the number of similar messages classified as ”ham” equals to the number of similar messages classified as ”spam”, resulting in the need for a tie-breaker function to make a classification.

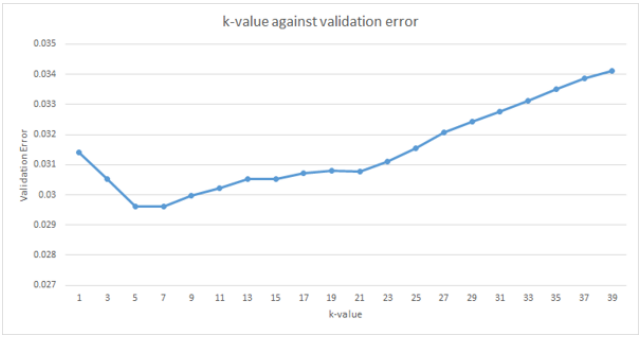

Figure 8: kNN Validation Error Graph

Figure 8 represents the validation error in classifying the messages for different k-values. It is clear that the k-values chosen have to be optimal. If the k-value is too small, overfitting will occur. If the k-value is too big, underfitting will occur, as the model will not be able to identify critical details when classifying new data.

In this scenario, we can see that the optimal k-value is 7. In other words, the model will take the seven most similar messages in the training set into account: the number of similar messages that are classified as ”spam”, and the number of similar messages that are classified as ”ham”; then classify the message by taking a majority voting (if the number of similar messages classified as ”spam” is bigger than the number of similar messages classified as ”ham”, the new message will be classified as ”spam”).

Discussion

Comparison between k-NN and SVM models

Both k-NN and SVM can be used to classify whether the message is a spam or ham at a high accuracy rate. Both models have their own strengths regarding the analysis of SMS messages.

In the case of the k-NN algorithm, SMS messages are short and spam messages are often structured similarly, hence, we can train an effective k-NN model without needing a large labelled dataset. This reduces the cost required to produce a good training dataset since obtaining labelled data requires verification of whether a message is spam or not and it can be costly to hire help to label a large dataset.

For the SVM, as we discussed earlier, the model's and active learning feature makes it more flexible due to its ability to be "customized" for each individual user. It also maintains a high speed and accuracy despite being modified to have less parameters. The ability to customize a spam filter to suit your own needs is a big selling point for the model since it panders to the users expectations of how specific they want their spam filters to be.

Between the 2 models, we would prefer implementing the SVM with active learning model in our spam filtering app. This is because we were able to provide a proof of concept that active learning maintains and can improve the classification power of the model.

Another advantage of using an SVM is that its classification speed does not suffer as much with a large training dataset. At its base, the SVM classifies an input instance based on its position on the SVMs hyperplanes and margins. This is definitely much faster than having to compare a new instance and its relative proximity to all other points in the training data as is the case for k-NN. This feature acts as a redundancy in case the training data requirements (size and thus complexity) for practical application are much bigger than the scope of this project.

Improvements

Fine-tuning

We can conclude that the machine learning algorithms used to classify SMS spams all have relatively high accuracy and using any of them will yield good results. However, if we want a higher accuracy, further fine-tuning needs to be done for both our models and our datasets.

For our project, we used word count vectors to form the training and test sets. This method is quick and fairly powerful for encoding short messages seen commonly in SMS, but may not capture word importance or intonation in the messages. We could consider using TF-IDF or other more complex word vectorisation methods in future to train our models.

We are also aware that spam message senders tend to also customize their wordings slightly in order to avoid detection. this is present in messages from unlicensed moneylenders or gambling syndicates which typically include emojis or other unusual characters that can impede the accurate classification of such messages. Such messages can be cleaned, by scanning through and removing such unusual characters from messages.

Business Applications

Spam Filtering Mobile App

On an individual level, the proposed algorithms aid the user in identifying spam messages. This can be applied by implementing said algorithms in a mobile application that allows spam messages to be pre-filtered. For example, the application initially only has a predefined set of training examples, and will filter subsequent incoming SMS using these training examples. Messages that are marked to be spam are then hidden from view and stored in another spam folder. Users can then improve the accuracy of the application by allowing the application to store messages, misclassified or not, thus providing the application with more training examples to filter SMS from.

This app will improve the quality of life for its users, as spam messages are automatically filtered by the app.

Personalisation Personalisation can be achieved by using SVM model for SMS filtering. The mobile app will query the user on messages which fall between the margins of its hyperplanes, in other words, borderline cases. A dialog overlay will prompt the user to classify such messages as ham or spam. This active learning process will further tune the model and is customized for each user. This is important as the definition of spam and ham messages are different for each user and we do not want to misclassify ham as spam (false negative).

Further tuning with user interaction Spam messages which are not successfully filtered can be notified by the user to the app. This allows the app to continuously train by the users as they use the app.

Telecom Markets

The algorithms can also be applied to a higher level, where Telecom markets can make use of such algorithms to catch spam and prevent them from being sent to the end-users terminals. Since all SMS must pass through the SMS control channel before reaching the end users terminal, a well-trained system can be deployed at a control channel that flags messages and prevents them from being sent out if it is detected as likely to be spam or phishing in nature.

Moreover, this app helps to fight the rising rate of scams, prematurely preventing suspicious SMS from being sent out will help in lower the number of people who fall for such scams. Since the spam messages are intercepted at the control channel, the end user does not need to receive such messages, leading to lower user frustration as they will not receive as many spam messages as before.

Conclusion

The high rate of smartphone possession and the widespread usage of SMS and instant messaging apps such as WhatsApp, makes it almost certain that users are bound to receive spam messages, whether from legitimate companies posting advertisements or illicit messages promoting money lending or gambling services. It would be a great service to help Singaporeans filter out such messages.

We have also covered some possible practical applications of the algorithms and believe that a skilled implementation can not only improve one's quality of life through receiving less annoying messages, but also lower the rate at which people fall prey to scams.

Acknowledgments

We would like to express our deep gratitude to Assistant Professor Bryan Low for his guidance for this project.

Appendix

A. Github repositories for SVM and k-NN

Our SVM model Github repository can be found in this following link: https://github.com/guaizw/CS3244_SMS_Spam_Project_svm

Our k-NN model Github repository can be found in this following link: https://github.com/jylee-git/CS3244-SMS-Spam-Project

References

[Almeida and Hidalgo 2019] Almeida, T. A., and Hidalgo, J. M. G. 2019. Sms spam collection. http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/. Accessed on 2019-04-02.

[Almeida 2011] Almeida, T.A., G. H. J. Y. A. 2011. Contributions to the study of sms spam filtering: New collection and results. In *Proceedings of the 2011 ACM Symposium on Document Engineering (DOCENG'11)*. Mountain View, CA, USA: University of California, Irvine, School of Information and Computer Sciences.

[Dua and Graff 2017] Dua, D., and Graff, C. 2017. UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences.