SPAM CLASSIFICATION USING TRANSFER LEARNING BY BERT PRE-TRAINED MODEL

Ehsan Mashhadi, Reyhaneh Jafari

Schulich School of Engineering, University of Calgary

ABSTRACT

It has been too many years that people get spam messages every day, so they may lose their money or time. Spam classification techniques are trying to prevent or label spam messages by different applications like emails or SMS. Many techniques and algorithms have been developed for this purpose, which has advantages and disadvantages, but in this work, we apply a pre-trained model called BERT by using transfer learning to detect spam SMS messages. We use the fine-tuning techniques in which the layers of a pre-trained model are frozen and some layers are added to the end of the mode. Then, those layers can be trained using a small dataset and limited resources to do the specific task. Our results show that our model is able to distinguish between spam and ham (not spam) messages.

Index Terms— Spam Classification, BERT, Transfer Learning, Transformers

1. INTRODUCTION

It is accepted that unwanted emails or text messages called spam have become a serious problem from years ago. Spam messages are preventing people from leveraging their time and may also distract many people. Spam threats have increased every year such that in Q3 2021, the share of spam in global mail traffic is 45.47% [1]. People find it very irritating to receive spams that they did not request, and they may lose their money because of being victims of scams and other attacks by receiving these messages. Spammers send messages containing interesting content from well-known companies which encourage users to click on a link or install an application to steal their sensitive information like their passwords and credit card numbers.

Since years ago different companies have used machine learning techniques to effectively handle this threat. These techniques analyze the content of messages to find if they contain any clues to mark them as spam. Since finding the spam message is counting as a classification technique there are many different classical and modern machine learning algorithms applicable to detect them. These algorithms include probabilistic, decision tree, support vector machine (SVM) [2], artificial neural networks (ANN), and case-based technique [3].

Mostly, these algorithms need a large dataset to perform well, but most of the existing datasets are not big enough to train a complex model from scratch. Also, it is sometimes required to train complex models to get the best result which requires many resources. Transfer learning technique is the reuse of a pre-trained model on a new problem which saves resources and since the model has been trained on a large dataset there is no need for such a large dataset on the new task [4].

BERT which stands for Bidirectional Encoder Representations from Transformers is a very popular language representation model published by Google AI Language [5]. It uses Transformers to learn contextual relations between words in a text. Its novel technique is using bidirectional transformers which reads the entire sequence of words at once as opposed to the directional models which read the text from left to right or right to left.

Bert used Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) as its training goals. The authors replaced 15% of the words in each sequence with [MASK] token and the model attempts to predict the original value of masked words for its first goal. For the second goal, pair of sentences are given to the model, and it should find if the second sentence in a pair is subsequent to the first sentence in the original document.

As the BERT model has been used for many tasks by researchers such as natural language inference, question answering, sentiment analysis, and news categorization, so we found it interesting to use this model for finding spam messages. We used the transfer learning technique by fine-tuning the BERT model by freezing the BERT model layers and adding some layers at the end so that it can detect if a message is spam or not.

2. RELATED WORK

Since finding spam messages has been investigated from many years ago there is much work done in this area which we will explore in this section.

Puniskis et al. [6] applied the neural network to find the spam by using attributes composed of descriptive characteristics of the evasive patterns that are being used by spammers without using the context or frequency of keywords in the message.

There is another work by Youn et al. [7] where the authors used different machine learning techniques such as Neural Network, SVM, Naïve Bayesian Classifier, and J48 classifiers and their results show that a simple J48 classifier could be efficient for the dataset which could be classified as a binary tree.

Kiritchenko et al. [8] reduced the classification error by finding temporal relations in an email sequence and embedding the discovered information into content-based learning techniques.

Also, Cui et al. [9] used Principal Component Analysis (PCA) as a preprocess of their NN model to reduce the data in terms of dimensionality and size.

3. MATERIALS AND METHODS

In this section, we will first explore our dataset and its statistics. Next, we will talk about the required preprocessing steps applied to the dataset. Then, we will discover the design of our technique for solving the problem.

3.1. Dataset

We used the dataset provided by [10] which contains 5,574 instances containing the SMS messages content and a label that shows if the SMS was labeled as spam or ham.

First, we deleted all empty columns and changed column names to ”text” and ”label” to make them more meaningful. Also, we replaced all the ”spam” values in the label column with ”1” and all the ”ham” values with ”0”. After doing these steps, we saved the result to a new file to have a neat dataset, so there is no need for doing these steps every time. There are Table 1. shows two instances of spam and ham messages.

In order to explore our data and get an idea about the most repeated words in the spam or ham messages, we plot the word cloud. The word cloud is displayed in Fig. 1, which shows that words like ”free”, ”call”, ”now”, ”stop”, and ”reply” are the most repeated words in spam messages and words like ”ok”, ”got”, ”u”, ”will”, ”call”, and ”go” are the most repeated words in the ham messages.

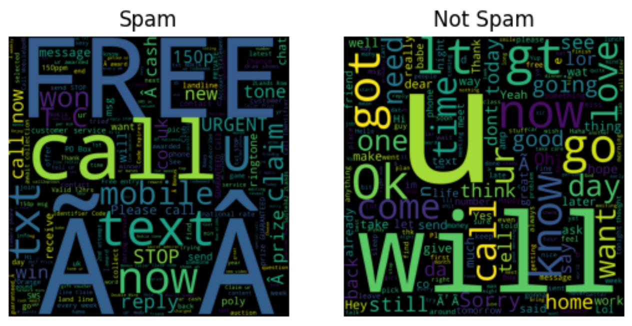


Fig. 1. Word Cloud of Spam and Ham Messages.

Text	**Label**
I’ll be late...	0
Ur cash-balance is currently 500 pounds	1
Was the farm open?	0
Dear Voucher Holder, 2 claim this weeks offer	1

Table 1. Sample Instances of Dataset.

3.2. BERT

BERT is a transfer model which has been trained on a large corpus of English data using the self-supervised method. It means that the model was trained using only raw texts and there was no human labeling process and the model has two goals of Masked Language Modeling (MLM) and Next Sentences Prediction (NSP). It helped the model to leverage an inner representation of the English language for extracting features useful for downstream tasks.

BERT has several variants, but the BERT base model which we use has 12 encoders with 12 bidirectional self-attention heads. It has 768 hidden states and 110M parameters. The original BERT model has been pre-trained on the concatenation of English Wikipedia (2,500M words) and the BookCorpus [11] (800M words).

3.3. Word Embedding

Representing the words for text analysis in the form of a real-valued vector which is called word embedding [12]. In this technique, vectors encode the meaning of words such that words that are closer in the vector spaces are expected to have a relationship like a similarity in their meaning. Since we want to use the BERT model it is important to see if this pre-trained model can understand the relationship between spam and ham messages.

First, we used the BERT tokenizer to encode the spam and ham messages, and then ask the pre-trained BERT model in the evaluation mode to predict the output of this encoded messages. Also, since this step is just for exploring the data and it does not affect the accuracy of the model we used fist 600 instances of our dataset to reduce the resource consumption. The vectors from BERT output have 768 dimension which is not possible to show them, so we use a popular dimension reduction technique to reduce the dimensions.

We used the t-Distributed Stochastic Neighbor Embedding (t-SNE) [13] which is an unsupervised and non-linear technique used for exploring and visualizing high-dimensional data. We reduced the dimension from 768 to two which makes it possible to show them in dimensional space. It calculates a similarity measure between pairs of instances in the high and low dimensional spaces. Then, it optimizes these two similarity measures using a cost function.

The first 600 sentences vectors from our dataset are displayed in Fig. 2 in which the spam messages have orange

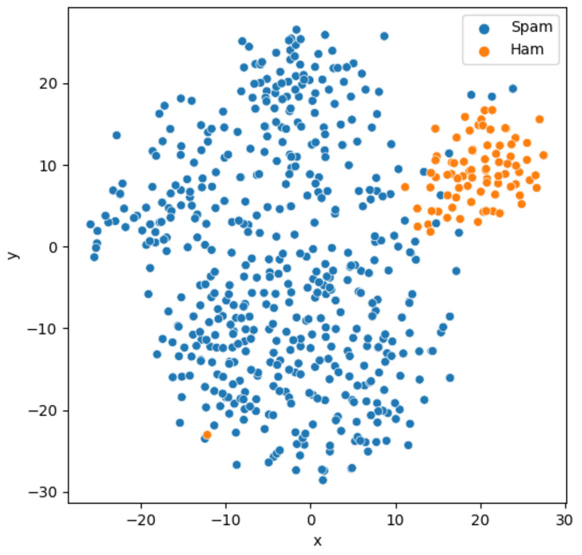


Fig. 2. Visualizing Spam and Ham Messages Using t-SNE.

colors and the ham messages have blue colors. The result shows that the original BERT model is able to somehow differentiate between the spam and ham messages because there are two clusters in which spam messages are closer together and ham messages are closer together.

3.4. Design

It is a common approach to convert all letters to lower case in natural language processing since there is no difference between uppercase and lowercase versions of one word like ”student”, ”STUDENT”, and ”STudenT” in many cases. Therefore, we decided to use the ”bert-base-uncased” version of the BERT model which does not make difference between the same words with lower or uppercase letters. We also count the maximum number of tokens in a sentence which is 238, so it is less than the BERT limitation which is 512 tokens.

We provided two versions by fine-tuning the BERT called simple and complex. In the simple version, we added a single layer feed-forward network with 768 inputs and 2 outputs. Also, we applied the LogSoftMax function which is log(Softmax(x)) to the output from the previously added layer.

In the complex version, we added several layers to help the model learn more complex relationships between data. We added a single layer feed-forward network with 768 inputs and 512 outputs, then we applied the rectified linear unit function to the output of previously added layers. After that, we added a drop-out layer to reduce the overfitting, and then we added a single layer feed-forward network with 512 inputs and 2 outputs. Finally, we applied the LogSoftMax function to the output from the latest added layer.

Before the training process, we freeze all the layers that exist in the BERT model and then add our defined layers to the end of the BERT architecture. We used the batch size of 32 since it is recommended in the BERT paper to use either use 16 or 32 as the batch size for the fine-tuning process. Also, we split our dataset into three splits, train, validation, and test by using 80%, 10%, 10% rules.

For each batch, we fed the training data into the model and use the negative log-likelihood loss value. The training loss is calculated by averaging the training loss of data in all batches. We also used the AdamW optimizer with the learning rate of 2e-5 since the authors of the BERT model suggested this value as one of the suitable values for the learning rate. The AdamW is a variant of the Adam optimizer that has improved the implementation of weight decay [14].

We use the validation set for computing the validation loss by averaging the validation loss of all batches. We did this phase to get an estimate of our model skill while tuning the model’s parameters, and we also save the weights of the model with the best validation loss.

For handling the class imbalance in our dataset we give different weights to the majority and minority classes. In this way, we influence the classification of classes during the training phase. The main idea behind this technique is penalizing misclassification by setting a higher class weight to the minority class and reducing the weight for the majority class. We used the balance heuristic [15] for calculating the weights and passing them to the loss function.

The process of training the model and validating using the training set and validation set respectively repeated for the 10 epochs, and the best model is saved in the disk. We also switched between the train and evaluation state of the model to enable and disable some specific layers/parts of the model that behave differently during training and evaluating time. We also turned off gradients computation during the evaluation phase.

For the model evaluation phase, we disabled gradients computation and used the test set by loading the weight of the saved model from the disk. We compare the predicted values of the test set with the original values by using different metrics. In an imbalanced dataset like our dataset, the accuracy is not a good metric to evaluate the result, but by using the precision and recall to calculate the F1-Score, it would be more meaningful. It is the weighted average of precision and recall, so it takes both false positives and false negatives into account.

We train and evaluate our model on a PC with 16G RAM and GeForce RTX 3070 GPU. The training time takes 4M and 6M for the simple and complex models respectively which seem rational.

4. RESULTS AND DISCUSSION

During the training phase, we calculate the training and validation loss and during the evaluation phase, we calculated different metrics such as precision, recall, and F1-score. The value of training loss and validation loss is displayed in Fig. 3. which shows that the model is learning data and the training loss and the validation loss are decreasing together which does not show any sign of overfitting.

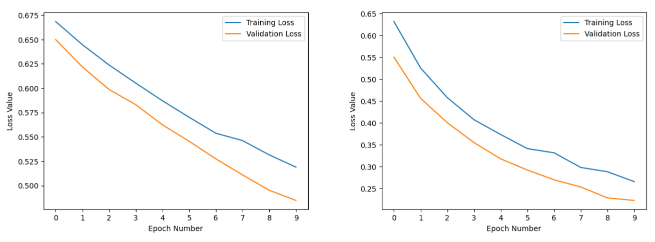
After training is finished we loaded the best saved model from the disk and calculated different metrics like precision, recall, F1-score, AUC, and ROC curve. The precision, recall, F1-score values are shown in Table 2.

It is a good practice to examine both precision and recall to fully evaluate effectiveness since precision and recall are often in tension. We used F1-score which is the harmonic mean of precision and recall. The results show that simple and complex models have good F1-Score however the complex version has a better F1-Score (0.96 & 0.75) than the simple model (0.91 & 0.68) which seems rational because of having more complex structures.

Also, Table 3 shows the accuracy and AUC score of model which compares the true positive rate vs false positive rate. It shows that if the model is able to distinguish between positive and negative classes. The higher the AUC, the better the performance of the model at this task. The AUC score of the complex model is 0.98 which is higher than the AUC score of the simple model (0.90) and this shows that the complex model is better in distinguishing between different classes.

By using this technique there is no need for a large dataset to train a model from scratch which requires many resources. Also, since the pre-trained model has been trained on a very large dataset it has a good knowledge of different contexts and domains, so the model could detect spam messages in different domains well. Also, it is capable of gaining context of words from both left to right and right to left simultaneously.

Since no technique or model is perfect always, there are some limitations with the BERT model and our technique too. The first limitation of the BERT is the lack of ability to handle long texts since it supports up to 512 tokens.



(a) Simple (b) Complex

Fig. 3. Training and Validation Loss of Simple and Complex Model.

	Class	Precision	Recall	F1-Score
Simple	0	0.99	0.84	0.91
	1	0.54	0.94	0.68
Complex	0	0.99	0.93	0.96
	1	0.60	0.99	0.75

Table 2. Evaluation Metrics of Fine-tuned BERT Model.

	Accuracy	AUC
Simple	85%	0.90
Complex	93.75%	0.98

Table 3. Accuracy and AUC of Fine-tuned BERT Model

But, there are some ways to handle this limitation like ignoring text after 512 tokens or breaking text into smaller parts and using them separately.

Another limitation of our work would be not using the k-fold validation to estimate the bias and variance of the results, but we assigned different weights to the majority and minority classes to overcome the imbalanced dataset problem.

Also, we did not run experiments with different values for hyper-parameters like learning rate and batch size to find the best possible values which may give us a better result. Furthermore, our model only works for messages in the English language, so it is another limitation that can be handled by using multi-lingual models.

Since we used the class weight to handle the imbalanced dataset, there are some other techniques like oversampling and undersampling which may lead to better results, so it is needed to run experiments with these techniques to find the best one.

5. CONCLUSION

Spam classification is one of the problems in different domains like emails and SMS messages, and many people lose their time or money because of receiving these messages. It is important to detect them in order to filter or mark them as spam. Many different techniques have been used to detect spam messages during past years which have their own challenges. We applied the BERT popular model and the transfer learning technique to leverage this pre-trained model. By using this technique we freeze all the layers and add several layers at the end of the pre-trained model. Because of using this technique, there is no need for a large dataset, and training the last few layers does not take much time. Our results show that our model is able to classify the spam and ham messages because of a good result in F1-score and the AUC score. For future work, it would be possible to tune the hyper-parameters to see if the results can be improved. Also, there are some other variants of the BERT model which have some differences in their architecture, so it is interesting to leverage those models to see if they can perform better than this model.

6. REFERENCES

[1] "Spam and phishing in q3 2021," https://securelist.com/spam-and-phishing-in-q3-2021/104741/, Accessed: 2021-12-05.

[2] Nizar Bouguila and Ola Amayri, "A discrete mixture-based kernel for svms: application to spam and image categorization," *Information processing & management*, vol. 45, no. 6, pp. 631–642, 2009.

[3] Florentino Fdez-Riverola, Eva Lorenzo Iglesias, Fernando Díaz, José Ramon Méndez, and Juan M Corchado, "Spamhunting: An instance-based reasoning system for spam labelling and filtering," *Decision Support Systems*, vol. 43, no. 3, pp. 722–736, 2007.

[4] Lisa Torrey and Jude Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pp. 242–264. IGI global, 2010.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[6] D Puniškis, R Laurutis, and R Dirmeikis, "An artificial neural nets for spam e-mail recognition," *Elektronika ir Elektrotechnika*, vol. 69, no. 5, pp. 73–76, 2006.

[7] Seongwook Youn and Dennis McLeod, "A comparative study for email classification," in *Advances and innovations in systems, computing sciences and software engineering*, pp. 387–391. Springer, 2007.

[8] Svetlana Kiritchenko, Stan Matwin, and Suhayya Abu-Hakima, "Email classification with temporal features," in *Intelligent Information Processing and Web Mining*, pp. 523–533. Springer, 2004.

[9] Bin Cui, Anirban Mondal, Jialie Shen, Gao Cong, and Kian-Lee Tan, "On effective e-mail classification via neural networks," in *International Conference on Database and Expert Systems Applications*. Springer, 2005, pp. 85–94.

[10] Tiago A Almeida, José María G Hidalgo, and Akebo Yamakami, "Contributions to the study of sms spam filtering: new collection and results," in *Proceedings of the 11th ACM symposium on Document engineering*, 2011, pp. 259–262.

[11] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 19–27.

[12] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[13] Laurens Van der Maaten and Geoffrey Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[14] Ilya Loshchilov and Frank Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[15] Gary King and Langche Zeng, "Logistic regression in rare events data," *Political analysis*, vol. 9, no. 2, pp. 137–163, 2001.

7. APPENDIX

In this section, we display the ROC curve of simple model in Fig. 4 and complex model in Fig. 5. ROC curve is a measurement for classification problems at various threshold settings. It summarizes the trade-off between the true positive rate and the false positive rate for a model using different probability thresholds. However, using the ROC curve for evaluating an imbalanced dataset is not a good idea since sometimes it would be misleading in some very imbalanced applications and it may seem that the model is doing very well.

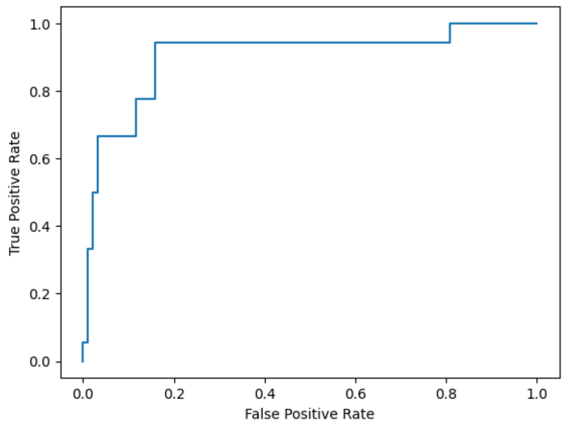


Fig. 4. ROC Curve of the Simple Model

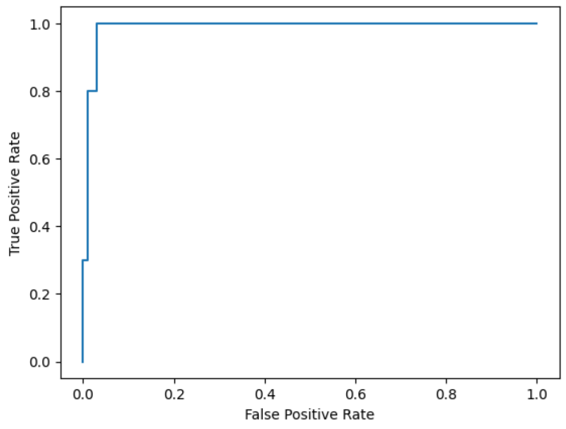


Fig. 5. ROC Curve of the Complex Model