Proceedings of the 3rd International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/6/20230571

Spam/Ham email classification using BERT

Siwei Zhang

School of Data Science, The Chinese University of Hong Kong, Shenzhen, Shenzhen, Guangdong Province, 518172, China

siweizhang@link.cuhk.edu.cn

Abstract. Email is a popular method for communicating with each other. However, as sending email is free of charge as long as an email server and a domain name are available, spam mail is becoming a critical problem in the email network. Conventionally, the industry uses spam filters based on rules and Bayesian inference to counteract spam mail, reaching an accuracy of 98.76%, which is far from satisfactory. Hence, to better protect email users from unsolicited messages containing advertisements, sensitive content, phishing content, and viruses, a new approach is proposed, in which email content is filtered by a spam detector using bidirectional encoder representations from transformers (BERT). BERT is a new language representation model published by Google that has achieved great success because of its powerful capabilities in understanding natural language. After the model is trained on a corpus from Kaggle, the spam detector equipped with the BERT model reaches a binary accuracy of 99.40% when classifying spam mail.

Keywords: machine learning, NLP, BERT, spam detector.

1. Introduction

Spam mail, which accounted for 45.37 percent of email traffic in December 2021, is a critical problem on the internet [1]. As everyone can set up an email server to deliver their email as long as they have internet access, the mailbox needs a spam filter to block spam email from their users. Conventional spam filters, such as SpamAssassin, are based on rules and Bayesian inference.

These conventional filters first apply the rules to check each incoming email to determine whether the sender is a creditable source. If an incoming email complies with the requirements, it will go to the Bayesian filter to further check the content. Combining these methods, spam email is unlikely to bypass the filters. Consider SpamAssassin as an example, as it is the basis for commercial anti-spam products built by first-class email providers [2]. According to research from the University of Waterloo, the supervised mode of SpamAssassin misclassifies 611 of 49,086 messages, reaching an accuracy of 98.76% [3]. However, John Graham-Cumming studied a vigorous statistical attack called Bayesian poisoning, against which the accuracy is even worse. By adding some carefully selected words, spammers cheat their way past the spam filter, which uses Bayesian inference [4]. A classical spam detector logic is summarized in figure 1.

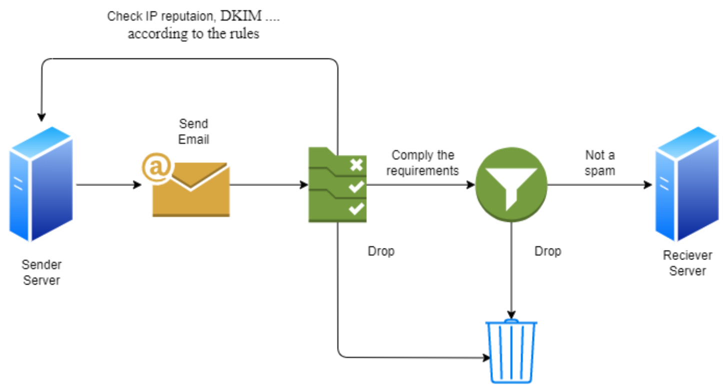

Figure 1. Spam detector logic.

(Photo credit: Original)

Even though the conventional filters perform well, chances are that the user will still receive some unsolicited emails. Hence, a few large companies simply blocklist whole internet protocol (IP) blocks without further processing, making email server self-hosting impossible. They do not follow the principles of net neutrality, making email delivery their proprietary service [5]. Measures should be taken, and a better approach to identifying spam will certainly encourage email providers to keep their antispam services online, making email an open protocol again.

Due to the limitations of the conventional approach and the complicated environment of the email network, a machine learning method for detecting spam emails is proposed. Recently, Google published a new language representation model named bidirectional encoder representations from transformers (BERT). Having an advantage over other natural language processing models, it achieves roaring success by realizing a 7.7% general language understanding evaluation (GLUE) score improvement [6]. Compared to the statistical method, machine learning aims to understand each message rather than calculating the number of times a word appears in the text. For the purpose of understanding language, BERT differs in two respects from previous machine learning methods: It trains a bidirectional language representation model by masking random words to avoid the problem of words seeing themselves vaguely, a nonnegligible interference on prediction. Moreover, training is conducted from both directions. Traditionally, models using long short-term memory (LSTM) and recurrent neural network (RNN) are trained from left to right, resulting in information loss on the word prediction when the right-side context impacts the meaning.

Nevertheless, a sentence cannot be fully understood without context; the model should dig much deeper. It adopts a transformer based solely on attention mechanisms [7]. Attention mechanisms successfully overcome the traditional algorithms' long-term dependence. The remainder of this paper is organized as follows: Section 2 demonstrates the methodology of this research, Section 3 presents the results of this research, Section 4 discusses the construction of the model in concrete detail, and finally, Section 5 presents the conclusions of this study.

2. Methodology

2.1. Dataset

The dataset [8], named UC Berkeley Data 100 Project 2, contains 8348 samples where 6208 emails are ham and 2140 emails are spam in a comma-separated values (CSV) format. It consists of four columns: observation ID, email subject, email body, and spam. As the email headers are stripped, the trained model will be independent of the email metadata, which means that the BERT filter will be fully content-based. Focusing on the content gives extra flexibility since the system admin can determine whether to combine it with additional meta rulesets. False-positive (good emails classified as spam) are highly undesirable. If good emails are sent from an IP with a bad history but a new owner, the content-based filter is more likely to let them through if no email meta rulesets are utilized. This dataset can be accessed from Kaggle, which was gathered by The University of California, Berkeley.

The dataset is randomly divided into two parts. One part is for training, which accounts for 80% of all cases. The other part is for testing, which accounts for the remaining 20%.

2.2. Model

Compared to RNN and LSTM, the recently proposed BERT model achieves better performance in real-world applications. BERT has an unparalleled benefit. That is, the model is pre-trained with a large-scale corpus. Based on an existing pre-trained model, which is adaptable to general natural language processing tasks, to make the model concentrate on a specific task, the only requirement is to add an extra output layer, necessitating the training of only a small set of parameters [6]. The classification model has several layers, which are the pre-processing layer, BERT encoder layer, concatenation layer, dropout layer, and classification layer.

Table 1. Pre-processing of an input sequence.

Input	Vector		
	Token (integer)	Segment	Position
[CLS]	Token $_{[CLS]}$	1	0
Money	Token $_{Money}$	1	1
maker	Token $_{maker}$	1	2
[SEP]	Token $_{[SEP]}$	1	3
Try	Token $_{Try}$	2	4
it	Token $_{it}$	2	5
for	Token $_{for}$	2	6
free	Token $_{free}$	2	7
[SEP]	Token $_{[SEP]}$	2	8

Pre-processing of an input sequence can be seen in table 1. The input text sequence is tokenized as an inner representation in the pre-processing layer. Different tokens, segments, and positions are extracted as features and put into vectors. These vectors are prepared as the input of the next layer. [CLS] denotes a special classification token, which is always the first token. [CLS] serves an important role in aggregating text sequences in classification tasks. [SEP] denotes a separator, which divides the inputs into several segments [6]. The previous output vectors are fed into the BERT encoder to generate pooled output. In this study, the program uses a small BERT pre-trained model (bert_en_uncased_L-4_H-512_A-8/2) [9] to balance the costs and benefits. It consists of 4 hidden layers with 512 hidden size and 8 attention heads. Compared to OpenAI Generative Pre-trained Transformer (GPT) [10], which has an unprecedentedly large number of parameters, this small BERT model is compact. Nevertheless, the performance is adequate for being embedded in an email system under restricted conditions, as the email system's memory is usually not large and a large latency is not acceptable. The concatenation layer is used to combine the pooled outputs of the email subject and body. When loading the input sequence, the email subjects and bodies are separately processed. Since the data are divided by the email's natural structure, it is obvious that the two different inputs will be assigned different weights in the BERT encoders, which will definitely improve the performance. After integrating these two outputs using the concatenation layer, a single output is obtained, making it easier for the follow-up layers to develop their parameters. The dropout layer aims to resolve the problem of overfitting. In the forward propagation process, the layer randomly masks some units in each iteration, preventing opposite units from canceling each other out by averaging. Moreover, it makes the model more robust, as the model will not rely on a specific feature but will take all features into consideration. The classification layer is an additional regular densely-connected NN layer. This is the main layer we are going to train in the email classification task.

2.3. *Fine-tuning*

The adjustment of a pre-trained model is called fine-tuning. With a pre-trained model trained on the large-scale corpus, the parameters inside the BERT encoder layer can approximately represent each word in a specific context. That is, the parameters are well trained for general-purpose natural language processing, resulting in a lower cost for fine-tuning. To take full advantage of the pre-trained model, a simple classifier is added to adjust the model to adapt to spam email detection. Additionally, the relatively small email dataset makes it difficult to support the entire model building process. The pre-trained model compensates for these disadvantages and provides better accuracy. The fine-tuned model used to classify email data with BERT can be seen in figure 2.

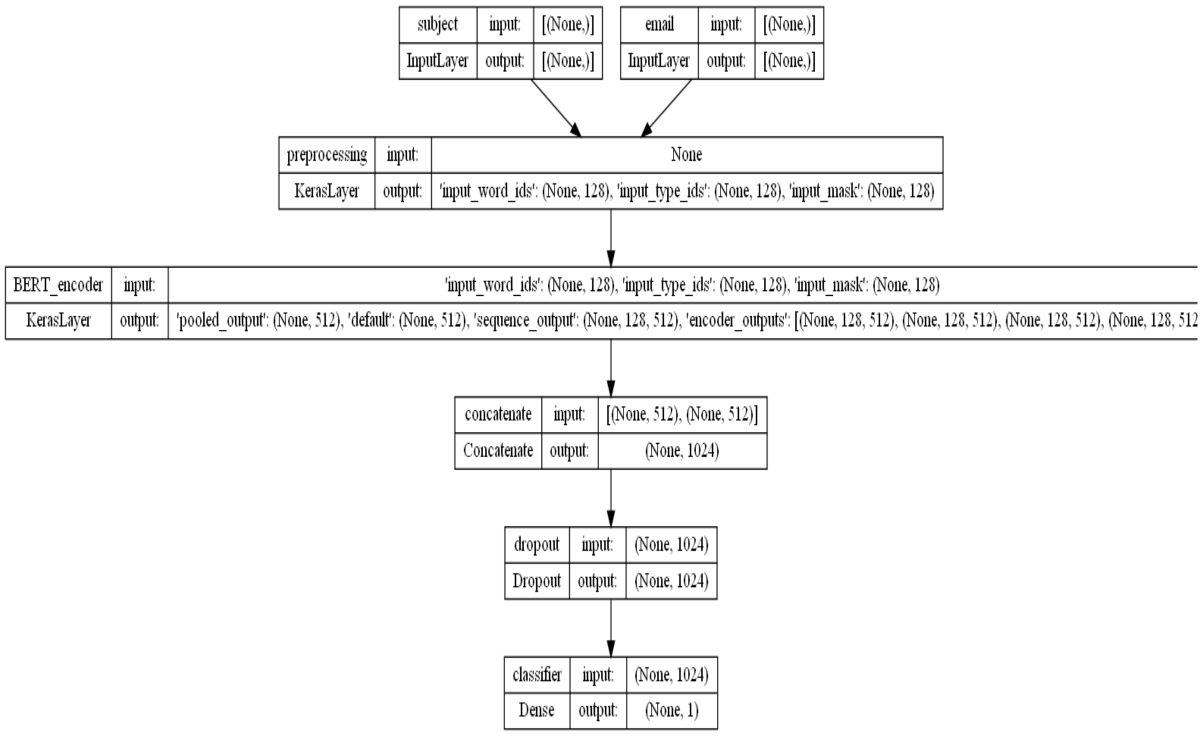
subject | input: | [(None,)]
InputLayer | output: | [(None,)]
email | input: | [(None,)]
InputLayer | output: | [(None,)]
preprocessing | input: | None
KerasLayer | output: | 'input_word_ids': (None, 128), 'input_type_ids': (None, 128), 'input_mask': (None, 128)
BERT_encoder | input: | 'input_word_ids': (None, 128), 'input_type_ids': (None, 128), 'input_mask': (None, 128)
KerasLayer | output: | 'pooled_output': (None, 512), 'default': (None, 512), 'sequence_output': (None, 128, 512), 'encoder_outputs': [(None, 128, 512), (None, 128, 512), (None, 128, 512), (None, 128, 512
concatenate | input: | [(None, 512), (None, 512)]
Concatenate | output: | (None, 1024)
dropout | input: | (None, 1024)
Dropout | output: | (None, 1024)
classifier | input: | (None, 1024)
Dense | output: | (None, 1)

Figure 2. Model used to classify email data with BERT.

(Photo credit: Original)

2.4. *Loss function*

A loss function is used to evaluate the performance during model training. The model tries to minimize the loss function to realize better performance. In this study, the training process uses the sigmoid cross-entropy function, which guides the model to be trained in a certain direction.

The function is defined as follows [11]:

$$\text{SigmoidCrossEntropy} = x - x * z + \log(1 + e^{-x}) \tag{1}$$

In this function, x = logits and z = target. Loss and accuracy during training can be seen in figure 3.

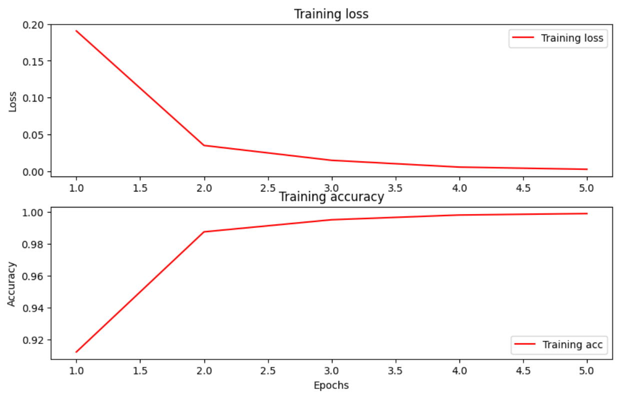

Figure 3. Loss and accuracy during training.

(Photo credit: Original)

3. Results

The model is trained with a batch size of 20, 5 epochs and an initial learning rate of 3e-5.

A 2x2 confusion matrix, table 2, is presented below to represent all four possible outcomes.

Table 2. Confusion matrix.

		Predicted	
		Ham	Spam
Real	Ham	TN	FP
	Spam	FN	TP

In this confusion matrix, the terms are defined as follows:

- TN: True-negative. The number of ham emails detected as ham.
- FP: False-positive. The number of ham emails detected as spam.
- FN: False-negative. The number of spam emails detected as ham.
- TP: True-positive. The number of spam emails detected as spam.

By applying the confusion matrix, the heatmap, figure 4, is plotted as follows:

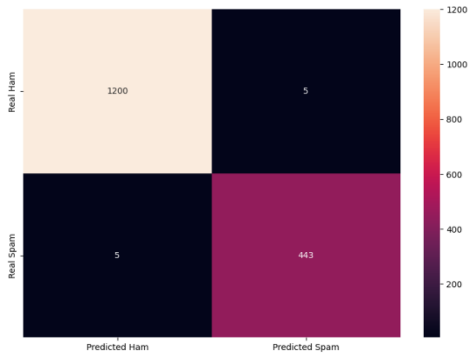

Figure 4. Spam/ham results on the UC Berkeley Data 100 Project 2 test data.

(Photo credit: Original)

3.1. *Evaluation functions*

An important metric for assessing a classification model is accuracy. In this research, the binary accuracy function, which classifies the confidence index into two classes according to a threshold is used. Accuracy is defined by the following formula:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

The calculated result is Accuracy $= 99.395\%$.

Recall indicates the proportion of spam emails that are identified correctly. A filter with a low recall rate will let spam email through to the inbox, causing inconvenience to users. Recall is defined by the following formula:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

The calculated result is Recall $= 98.884\%$.

Precision indicates the proportion of identified emails that are truly spam emails. A filter with a low precision rate will block ham emails, interfering with the normal work flow. Precision is defined by the following formula:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4}$$

The calculated result is Precision $= 98.884\%$.

F1 score can synthetically indicate how good a model is in a different aspect. It is the harmonic mean of precision and recall. The F1 score is defined by the following formula:

$$\text{F}_1\text{score} = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{5}$$

The calculated result is F_1 score $= 98.884\%$.

4. Discussion

The above data are acquired with a standard threshold. As the F_1 score result is sufficient under the standard circumstances, considering that users can accept small numbers of spam emails, the filter threshold is adjusted to push the precision to 100% because the users do not want to miss any ham emails.

When the threshold = 4, all false-positive results disappear at the cost of 14 spam emails passing through the BERT filter, with a recall rate of 96.875%. Receiving only 14 spam emails when there are 1219 emails in the inbox is totally acceptable. Compared to ham mail being lost due to the use of too restrictive of a threshold, letting the user report the spam emails and gathering them for use as training data in the next iteration is a more reasonable approach and follows the principles of net neutrality.

Due to the well-performing BERT model, some mails near the threshold are not totally spam. Even humans must make some effort to classify them. A better strategy is to let these mails directly post into a separate folder rather than recording a no-receipt log and removing them from the server. Additionally, for a different purpose, permitting users to adjust the threshold is a no-cost decision. As the BERT model already assigns a confidence index to each email, there is no need for the machine learning model to be rerun on all existing emails after the threshold adjustment.

4.1. *Selection of batch size, learning rate, and number of epochs*

Initially, the batch size is set to TensorFlow's default value of 32. After the model is trained for several iterations, the model's accuracy rate hovers at 99.20%. Batch sizes of 0.5 times and 2 times the original batch size are utilized. According to the accuracy results, a smaller batch size may result in better performance. Using the divide and conquer algorithm, the optimized batch size is located in the interval between 16 and 32. After multiple trials, the batch size is set to 20.

When the learning rate is set to 3e-7, to achieve better performance, the number of epochs needs to be increased to 50. Nevertheless, the accuracy rate is stuck at 98% when the number of epochs is increased, and no progress is observed. The learning rate is then set to 3e-6, and the number of epochs is set to 20. However, the performance is still disappointing. The learning rate is finally set to 3e-5, and the number of epochs is set to 5; then the result reaches an accuracy of 99.40%. The number of epochs is also increased with the learning rate unchanged. The result seems to be overfitting, which will not be satisfactory in real-world applications.

4.2. Combining with rulesets

Undoubtedly, the machine learning based model consumes more computational power than the traditional rulesets. To reduce the machine learning based filter's power usage, it is a smart move to apply the rulesets in front of the BERT filter. Additionally, backed up by the robust BERT filter, the ruleset penalty threshold can be reduced. With multiple guards involved, not only is the demand for computing resources lower, but also it is more difficult for spam emails to cheat their way into the inbox.

5. Conclusions

Although there are laws that regulate internet users, because of the difficulty of enforcement, spam is still a serious issue in the communication network. Measures should be taken to address spam, and consequently, developers should build a series of defensive products. From the sender policy framework (SPF) to Bayesian filters, thousands of programs have been developed to protect users from spam. There is no doubt that machine learning based filters will be an integral part of the email system. BERT, which is one of the machine learning models applied in this study, shows unparalleled performance in spam detection, reaching an accuracy of 99.40%.

In the future, the BERT filter will be combined with traditional rulesets to provide more accurate and synthetic results. The spam emails that slip through the net, after being identified by users, will also be added to the dataset for training to further improve the model's performance.

References

[1] Guo, Y., Mustafaoglu, Z., and Koundal, D 2022 *Journal of Computational and Cognitive Engineering*.
[2] Wang, F., Ko, R., and Mickens, J 2019 *In 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)* 615-630.
[3] Gordon C, and Thomas L 2007 Online Supervised Spam Filter Evaluation. *ACM Transactions on Information Systems* **25** 3 11 (*Preprint* https://doi.org/10.1145/1247715.1247717)
[4] John G 2004 How to beat an adaptive spam filter *MIT Spam Conference* Cambridge
[5] Cindy C, Annalee N 2011 Noncommercial Email Lists: Collateral Damage in the Fight against Spam *Electronic Frontier Foundation: White Paper* Electronic Frontier Foundation
[6] Jacob D, Ming-Wei C, Kenton L, and Kristina T 2019 *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* Minneapolis Association for Computational Linguistics **1** 4171–4186 (*Preprint* https://doi.org/10.18653/v1/n19-1423)
[7] Ashish V, Noam S, Niki P, Jakob U, Llion J, Aidan G, Lukasz K, and Illia P 2017 Attention is All You Need *Advances in Neural Information Processing Systems* **30** (*Preprint* https://doi.org/10.48550/arXiv.1706.03762)
[8] The University of California, Berkeley Data 100 Fall 19, Project 2 2019 Kaggle
[9] Turc I, Ming-Wei C, Kenton L, and Kristina T 2019 Well-Read Students Learn Better: On the I mportance of Pre-Training Compact Models *Preprint* https://doi.org/10.48550/arXiv.1908.08 962
[10] Alec R, Karthik N, Tim S, and Ilya S 2020 Improving Language Understanding with Unsupervised Learning OpenAI
[11] Martín Abadi *et al.* 2015 TensorFlow: Large-scale machine learning on heterogeneous systems *Preprint* https://doi.org/10.5281/zenodo.4724125